Deep learning methods in epidemiology and their applications to electronic health databases

Louis Falissard

Thèse de doctorat

NNT 2021UPASR019

université PARIS-SACLAY

Epidémiologie profonde : méthodes d'apprentissage profond et leurs applications sur des bases de données médico-administratives

Deep learning methods in epidemiology and their applications to electronic health databases

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 570, école doctorale de santé publique, EDSP
Spécialité de doctorat : santé publique - biostatistiques
Unité de recherche : CépiDc-INSERM
Référent : Faculté de médecine

Thèse présentée et soutenue à Paris-Saclay, le 23/09/2021, par

Louis FALISSARD

Composition du Jury

Emmanuel BACRY DR, CNRS	Président
Aline DESESQUELLES DR, Ined	Rapporteur & Examinatrice
Stéphane GAIFFAS PU, Université de Paris	Rapporteur & Examinateur
Mohamed NADIF PU, Université de Paris	Examinateur
Pascale TUBERT-BITTER DR, INSERM	Examinatrice
Aurélie NEVEOL CR, CNRS, LISN	Examinatrice

Direction de la thèse

Grégoire REY HDR, CépiDc-INSERM	Directeur de thèse
Karim BOUNEBACHE CépiDc-INSERM	Co-Directeur de thèse

Remerciements

Je tiens tout d'abord à remercier mon directeur de thèse, Grégoire Rey, sans qui la réalisation de cette thèse n'aurait jamais été possible. Déjà parce qu'en plus de s'être montré extrêmement à l'écoute et disponible pendant l'intégralité de ma thèse, il a mis à ma disponibilité tous les certificats de décès de France de 2000 à 2017 et que c'est quand même un sacré jeu de données. Et que sans données, en apprentissage profond on ne fait pas grand-chose. Mais surtout parce qu'à travers les nombreux échanges que l'on a pu avoir pendant ces trois ans, j'ai appris énormément sur les enjeux de l'épidémiologie et la santé publique. Sur à quel point les subtilités de la discipline existent en dehors des données, et non pas dans l'espace vectoriel qu'elles dessinent (même si ce dernier est très joli). Maintenant je peux presque faire semblant de m'y connaitre un peu en épidémio, et c'était exactement ce que je recherchais en postulant à l'école doctorale de santé publique. Donc encore merci.

Je tiens ensuite à remercier Karim Bounebache, mon co-directeur de thèse, pour ses remarques pertinentes et exigeantes sur les premières versions de mes articles, et pour les discussions sur les comics le midi à l'époque où on avait encore le droit de manger tous ensemble le midi (c'était vachement bien quand même cette époque).

Je tiens aussi à remercier l'EDSP, et tout particulièrement Jean Bouyer, pour avoir eu l'intelligence de me mettre en contact avec Grégoire, et pour avoir accordé suffisamment de confiance en ce projet de thèse pour nous accorder un contrat doctoral.

Ce travail n'aurait pas non plus été possible sans toute l'équipe du CépiDc. Tous les codeurs, y compris ceux que je n'ai pas connus, qui ont nourri tous les jours cette superbe base de données qui m'a permis de tant m'amuser ces 3 dernières années, et particulièrement Sylvie, que j'ai passé de nombreuses heures à embêter pour comprendre comment son cerveau fonctionnait quand elle regardait un certificat de décès. En vrai c'est super compliqué comme exercice, et tous les exemples que vous avez pu me donner étaient extrêmement instructifs. J'espère en retour que les outils que j'ai pu développer vous seront utiles dans le futur. Merci beaucoup à Claire Morgand également, qui est probablement la personne que j'ai le plus exploité au labo. Merci beaucoup d'avoir accepté de recoder totalement à la main quelques 200 certificats de décès, pour que je puisse m'amuser à faire un test de Turing de codage médical. Ça représente un travail de dingue. Je tiens aussi à remercier Walid et Claire Imbaud, qui m'ont beaucoup aidé à comprendre les obscurs méandres des bases du CépiDc ou du SNDS, allant parfois jusqu'à me faire des extractions eux même (pas du SNDS évidemment parce que ce serait très illégal). Je me dois également de remercier Pierre Etienne, qui a su supporter pendant de longues heures mes sifflotement de classiques Disney avec un stoïcisme digne du Buddha. Finalement, je tiens à remercier tous les gens du labo dans son intégralité, pour les moments qu'on a pu passer à parler de tout et n'importe quoi, et surtout de n'importe quoi. En particulier pour les blagues macabres, qui sont tout de même très à propos dans une institution pareille, et sans lesquelles travailler à longueur de journée sur l'histoire de la mort des gens pourrait devenir un peu pesant.

Pour ce qui est de l'extérieur du CépiDc (là où on parle plus de vie que de mort) je vais commencer par le commencement et remercier mes parents. Déjà parce que sans eux je ne serais littéralement pas là, et qu'être là, c'est quand même plutôt sympa. Au cours de ces presque 30 années d'existences, vous m'avez apporté énormément, poussé énormément, alors que ça devait pas forcément être facile tous

les jours, paresseux comme je suis. Si je suis en ce vendredi après-midi ensoleillé en train d'écrire les remerciements de ce travail de thèse qui m'a énormément plu, c'est en grosse partie grâce à vous. Alors merci beaucoup, et des bisous. Merci à mes sœurs aussi, parce qu'elles sont grave cools, et qu'elles se sont prises pas mal de discussions matheuses dans la tête les dimanches en famille alors qu'elles en ont quand même pas grand-chose à faire, il faut bien le dire.

Puis je tiens évidemment à remercier tous mes amis qui ont pu un jour subir un monologue mathématiques d'une à trois heures à des degrés d'alcoolémies pour lesquels ça devrait littéralement être interdit, ce qui si je ne m'abuse équivaut à remercier tous mes amis. Au moins ça ne fait pas de jaloux. Vu qu'il m'était impossible de décider de l'ordre dans lequel vous citer, je viens à l'instant de coder un petit échantillonneur aléatoire (distribution uniforme évidemment) en python. De gros bisous donc à Tox (comme quoi le hasard fait parfois bien les choses, après tout t'es mon plus vieil ami), Gomar, La Martina, Yoyo, David, Marie Canapé, Flogio, Charline, Mika, Marine G, Cutave, MC Tintamarre, Javière la rivière, STL, Marie LAM, Clairouche, Anne-Lolo, Mèxe, Valère, Marine Kitten, Idris, Anne, Tim'Tim', Cha et Jox. Vous êtes le sang vous-même vous savez (oui je me lâche un peu sur le vocab' là mais bon s'il y a un moment pour se faire plaisir). Je ne connais pas la moitié d'entre vous autant que je ne le voudrais, mais j'aime moins de la moitié d'entre vous à moitié moins que vous ne le méritez. J'ai vécu et appris des choses très belles avec chacun de vous, et pour ça je ne vous remercierai jamais assez. Pour finir, il semblerait que j'ai omis trois noms de cette (longue ça fait plaisir d'ailleurs) liste de personnes chères à mon cœur. Ça n'est certainement pas parce que je ne les remercie pas de tout ça, bien au contraire. Je tiens donc maintenant à remercier mon Guillaume, pour tous ces « business call » à base de ti 'punch qu'on a pu se faire, et qu'on va se faire dans le futur. Ça va être grave cool. Et pour tout le reste toi-même tu sais. Ça y est ma page de remerciement commence à ressembler à un skyblog de gamin de 4ème. Des bisous aussi au grand maître Lucus, merci pour tous ces moments qu'on a pu passer chez toi à siroter du whisky en discutant d'épistémologie et de causalité, avant de se mater tranquillement des mangas de cuisine. Tu m'as ouvert à tout un champ de pensée que je vais adorer explorer au cours des prochaines années. Finalement, un immense bisou au panda roux d'Aubervilliers. Pour tous les bons moments qu'on a passés quand c'était très compliqué. Pour m'avoir fait découvrir le langage sociologique, la matrice, et pour m'avoir inspiré ma théorie des langages. Merci aussi à son correspondant, le panda roux du zoo de Beauvais, tout simplement parce qu'il est trop mignon. Je sais que la distance vous pèse mais accrochez-vous, vous vous reverrez bien assez vite.

Un grand merci à la salle d'escalade Antrebloc, à Villejuif, pour être restée ouverte pendant les six derniers mois, malgré la situation sanitaire. Vous avez activement contribué à me garder saint d'esprit pendant l'écriture de cette thèse.

Peut-être aussi merci aux deuxièmes et troisièmes confinements. La tradition exige qu'un thésard s'enferme dans sa piaule durant de longs mois pour pondre sa thèse, et que c'est pas forcément le meilleur moment de l'existence. Au moins grâce à vous j'aurais pas eu le choix.

Valorisation scientifique

Articles

- A deep artificial neural network based model for underlying cause of death prediction from death certificates[1] (article publié dans JMIR Medical informatics)
- Neural translation and automated recognition of ICD-10 medical entities from natural language[2] (article accepté pour publication dans JMIR)
- Learning a binary search with a recurrent neural network: A novel approach to ordinal regression analysis[3] (article prépublié et en cours de soumission)
- On the comparability of international cause of death statistics: A deep artificial neural network based study (article finalisé et en cours de soumission)

Communications orales

- OMS (octobre 2019): A deep artificial neural network based model for underlying cause of death prediction from death certificates
- Santé publique France (septembre 2019) : Apprentissage profond et prédiction de la cause initiale de décès
- CESP B2PHI (janvier 2020) : Apprentissage profond et prédiction de la cause initiale de décès
- HAS (2021) : Traduction neuronale et reconnaissances d'entités médicales CIM-10 à partir du langage naturel
- Ined (2020) : Traduction neuronale et reconnaissance d'entités médicales CIM-10 à partir du langage naturel
- CESP Oncostat (2021) : Apprentissage profond et prédiction de la cause initiale de décès

Table des matières

1 Synthèse en français

1.1 Introduction

L'exploitation des bases de données médico administratives est récemment devenue un sujet d'importance en épidémiologie et santé publique. Ces recueils massifs de données de santé ont en effet le potentiel d'être source de quantité d'informations innovantes pouvant aider à l'élaboration de nouvelles politiques de santé publique. Cependant, l'analyse de ces jeux de données à la dimensionnalité souvent particulièrement élevée présente de nombreuses contraintes, autant au niveau de leur incompatibilité avec les méthodes statistiques traditionnellement utilisées en santé publique, que pour des raisons purement computationnelles. Parallèlement au gain d'intérêt croissant que portent les praticiens de santé publique à ces bases de données, la dernière décade a été témoin de la démocratisation des méthodes dites d'apprentissage machine, et tout particulièrement d'apprentissage profond, qui propose toute une famille de puissants modèles prédictifs tout particulièrement adaptés à l'analyse de complexes interactions non linéaires dans des jeux de données autant massif que non structurés, et dont certaines ont déjà été appliquées avec succès sur des bases médico administratives par le passé. En revanche, et ce de par leur différence fondamentale de culture, la communication entre épidémiologistes et biostatisticien d'un côté, et scientifique de la donnée de l'autre, a entrainé des complexités de communications et de coopérations entre les deux domaines. L'objectif de cette thèse est double. En premier lieu, cette thèse se veut être une introduction pour les épidémiologistes et biostatisticiens aux méthodes modernes d'apprentissage profond, à travers un travail de reformulation de ces méthodes dans une optique purement statistique, par opposition au langage cognitiviste auquel elles sont souvent associées. On exposera notamment que l'intégralité des méthodes modernes d'apprentissage profond, ce jusqu'aux modèles de séquence typiquement utilisés en traitement du langage naturel, peuvent s'exprimer en termes de modèles prédictifs, et notamment comme des extensions du concept de modèle linéaire généralisé. Un exemple innovant d'application de ces méthodes au problème de modélisation de variable ordinale sera également introduit pour montrer l'adaptabilité de ces méthodes à tout un spectre de problèmes régulièrement rencontrés dans des contextes de science des données médicales. Dans un second temps, ces concepts d'apprentissage profond seront directement appliqués à une base de données médico-administrative, la base du Centre d'Epidémiologie sur les Causes de Décès (CépiDc) pour illustrer leur potentiel, autant d'un point de vue purement méthodique que pratique, avec des résultats variant d'une accélération de la production des données de mortalité par cause ayant notamment permis la production en temps réel

de statistiques de comorbidités relatives aux décès liés à covid-19 pendant le premier confinement, à l'élaboration d'une étude sur la question de la comparabilité des statistiques de mortalité à l'échelle internationale.

1.2 Apprentissage profond et modèles prédictifs

L'apprentissage machine est un domaine d'étude qui se consacre à donner à un ordinateur la capacité d'effectuer une tache prédéfinie, mais sans instructions explicite pour ce faire. Plus formellement, on dit d'un programme informatique qu'il apprend d'une expérience E vis-à-vis d'une tache T et d'une mesure de performance P, si ses performances sur T, mesurées par P, s'améliore avec l'expérience E ». Cette définition particulièrement générale donne lieu à plusieurs interprétations mathématiques, mais peut tout particulièrement être rapprochée du concept de modèle prédictif en analyse statistique.

Les modèles de régression linéaires ou logistiques, par exemples, peuvent tout à fait être considérés comme des algorithmes d'apprentissage machine dits supervisés, un sous genre d'apprentissage machine focalisé sur la modélisation, à partir d'un jeu de données de la relation entre une ou plusieurs variables explicatives, et une variable à expliquer. On distingue au sein du domaine de l'apprentissage supervisé deux exercices distincts, en fonction de la nature des variables à expliquer :

- L'exercice de classification, où la variable à expliquer est qualitative. Dans ce cadre, le but de l'algorithme d'apprentissage est de trouver une surface fonctionnelle dépendant des variables explicatives qui sépare le mieux les états distincts de la variable à expliquer. Les méthodes de régressions logistiques et d'analyse linéaire discriminante peuvent être considérées comme des exemples d'algorithme de classification.
- L'exercice de régression, où la variable à expliquer est quantitative. Dans ce contexte, le but de l'algorithme d'apprentissage est de trouver une fonction dépendant des variables explicatives qui approxime le mieux la variable à expliquer. L'analyse de régression linéaire peut être considérée comme un exemple d'algorithme de régression.

1.2.1 Apprentissage profond

1.2.1.1 Régression linéaire

Un modèle de régression linéaire est un modèle destiné à modéliser la relation entre une variable à expliquer quantitative et une ou plusieurs variables explicatives par le biais d'une combinaison linéaire de ses dernières. Les coefficients de cette combinaison, appelés paramètres du modèle, sont obtenus par la minimisation d'une fonction objectif construite à partir d'un ensemble de pair d'observations des variables explicatives et de la variable à expliquer. La fonction objectif des moindres carrés constitue un exemple de fonction objectif couramment utilisé dans ce contexte, et son minimum peut s'écrire sous forme explicite.

1.2.1.2 Perceptron multicouche

Le perceptron multicouche, forme la plus élémentaire de modèle d'apprentissage profond, peut être considéré comme une évolution des modèles de régression logistique ou linéaire permettant de modaliser des relations non-linéaires complexes entre une variable à expliquer et un ensemble de variables explicatives. Ces modèles sont construits à partir d'une brique élémentaire appelée neurone artificiel, largement inspiré des neurones observés en biologie dans les cerveaux animaux, et dont la forme fonctionnelle est définie de manière similaire à celle d'un modèle généralisé, à savoir une combinaison linéaire de variables d'entrées injectée dans une fonction non linéaire (historiquement une sigmoïde dans les modèles d'apprentissage profond des années 90, par exemple). L'idée principale du perceptron multicouche consiste à définir la forme fonctionnelle du modèle prédictif utilisé pour la modélisation comme un réseau dirigé acyclique de ces neurones artificiels, comme peut être observé en figure 1.1.

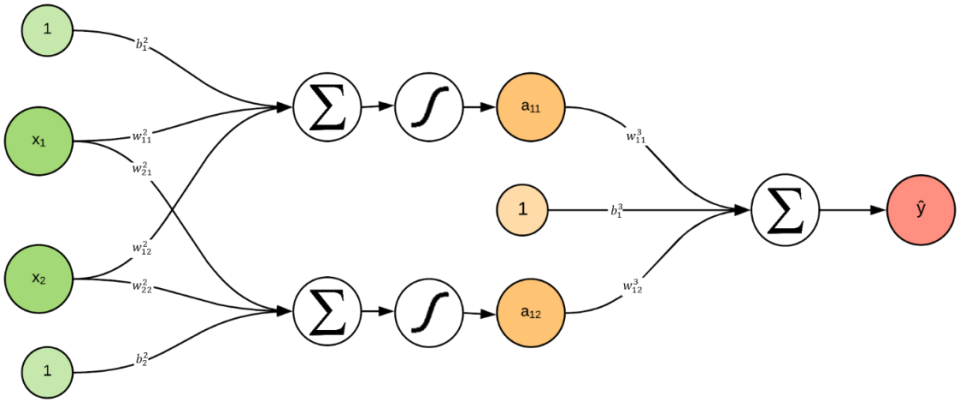


Fig. 1.1 Exemple de perceptron multicouche. Chaque « neurone », brique élémentaire du modèle, partage sa forme fonctionnelle avec celle des modèles linéaires généralisés

Le processus d'ajustement du modèle, en revanche, ne diffère pas fondamentalement de ceux observés dans les modèles statistiques standards, et passe par la minimisation d'une fonction objectif (typiquement un objectif de moindre carré ou d'entropie croisée dans le cas d'une variable à expliquer quantitative ou qualitative, respectivement), généralement par l'utilisation d'un algorithme d'optimisation différentielle. En revanche, certaine s0propriétés de la fonction objectif (explosion ou évaporation de gradient, non-convexité) rendent l'ajustement de ces modèles impossible sans un ensemble de techniques additionnelles, qui peuvent être résumées comme suit :

- Faciliter l'ajustement du modèle en modifiant la surface fonctionnelle à optimiser :
    - Choix de la non-linéarité (exemple : Utiliser des unités linéaires rectifiées en lieu des sigmoïdes historiquement utilisées)
    - Méthodes de normalisations (exemple : normalisation des variables d'entrées, normalisation par fournées ou normalisation par couches)
- Modifier l'algorithme de descente de gradient (exemple : algorithme d'optimisation Adam)

- Affiner les méthodes d'initialisation de l'algorithme de descente (exemple : méthode d'initialisation de Xavier)
- Utiliser des méthodes de régularisation pour éviter le sur-ajustement (exemple : régularisation par drop-out)

Le perceptron multicouche peut par la suite être modifié par le biais de deux idées fondamentales, celles de partage de paramètres et de neurone à connexions restreintes, pour s'adapter à différents cadres de modélisation où les modèles vectoriels classiques apparaissent souvent comme inadaptés, comme en analyse d'image ou de texte. Les deux variantes du perceptron les plus répandues dans la littérature sont obtenues comme suit :

- Adapter le perceptron multicouche à l'analyse de matrices ou tenseurs tridimensionnels, et tout particulièrement d'images, en imposant au modèle des contraintes de localités et d'invariance par translation. La localité est exprimée en limitant l'accès pour chaque neurone à seulement des sous matrices de tailles prédéterminées des variables d'entrées. L'invariance par translation est exprimée en copiant un neurone (au sens de ses paramètres) donné sur toutes les sous matrices des variables d'entrées. Cette famille de modèle est connue sous le nom de réseau de neurones à convolution.

- Adapter le perceptron multicouche à l'analyse de séquences vectorielles, en imposant au perceptron de prendre une observation de la séquence d'entrée à chacun de ces étages, de manière successive, et en partageant à chaque étage les mêmes paramètres. Cette famille de modèles est connue sous le nom de réseau de neurones récurrents.

1.2.2 Apprendre une recherche dichotomique avec un réseau de neurones récurrent, une approche originale à l'exercice de régression ordinale

L'exercice de régression ordinale est un problème d'analyse statistique qui consiste à modéliser une variable à expliquer qualitative mais dont les états présentent un caractère ordonné, et qui peut à ce titre être considérée comme une hybridation entre une variable qualitative et quantitative. Cette nature unique implique que les méthodes de régressions de variables quantitatives ou qualitatives ne lui sont pas parfaitement adaptées, en conséquence de quoi plusieurs méthodes spécialisées dans cet exercice ont été proposées au fil des années, typiquement inspirées soit des méthodes de régression de variables quantitatives, soit des méthodes de régression de variables qualitatives. Cependant, ces méthodes impliquent généralement des hypothèses sensiblement plus fortes que le simple caractère ordonné de la variable à expliquer. Dans cette partie, nous regarderons comment l'on peut s'inspirer d'un simple algorithme de recherche en table ordonnée, la recherche dichotomique, pour créer une nouvelle méthode de régression ordinale à base de réseaux de neurones récurrents potentiellement plus performante que les méthodes préexistantes.

1.2.2.1 Méthode

L'algorithme de recherche dichotomique est un algorithme de recherche dont la fonction est de retrouver au sein d'une table ordonnée une valeur donnée. Le fonctionnement de cet algorithme récursif est illustré en figure 1.2. L'idée fondamentale de la méthode proposée est de projeter la variable ordinale à expliquer sur un arbre binaire dichotomique, et de s'intéresser à l'adresse de chaque état sur cet arbre plutôt qu'à l'état même, cette adresse étant constituée d'une séquence de décision binaire (« au point où je me trouve dans l'arbre, dois-je aller à droite ou à gauche pour retrouver l'état de la variable que je recherche »).

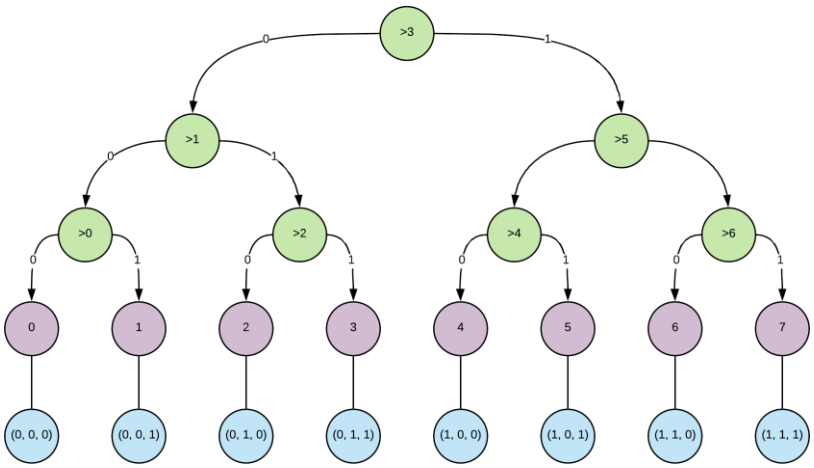


Fig. 1.2 Arbre de recherche dichotomique. Chaque état de la variable ordinale est associé à une suite de variables binaires représentant sa position dans l'arbre, autrement dit le résultat de la recherche dichotomique équivalente. On remarque que ces suites de variables binaires correspondent exactement à la décomposition binaire de l'état de la variable ordinale

Le problème de modélisation ordinale a donc ici été converti en un problème de modélisation d'une série de variables binaires, avec cependant une complexité additionnelle résidant dans l'interdépendance cumulative des variables à expliquer. Ce genre de problème, rarement rencontré en statistiques classique, et toutefois régulièrement rencontré en apprentissage profond, tout particulièrement en traduction automatique. A partir de cette constatation, sera proposé dans cette thèse pour régler ce problème de s'appuyer sur une architecture de traduction automatique appelée « architecture encodeur-décodeur basée sur des réseaux de neurones récurrents » pour développer une nouvelle méthode de régression ordinale. Ses performances prédictives seront évaluées sur un ensemble de jeu de données benchmark et comparées aux résultats obtenus par des méthodes plus traditionnelles proposées dans le package python mord.

Les performances de ces diverses méthodes seront évaluées par le biais de trois métriques distinctes, l'exactitude, le Cohen Kappa quadratique et l'erreur quadratique. La figure 1.3 présente un bref

résumé des résultats obtenus, où l'on peut constater que la méthode proposée semble se comporter de manière similaire ou significativement mieux que les méthodes traditionnelles.

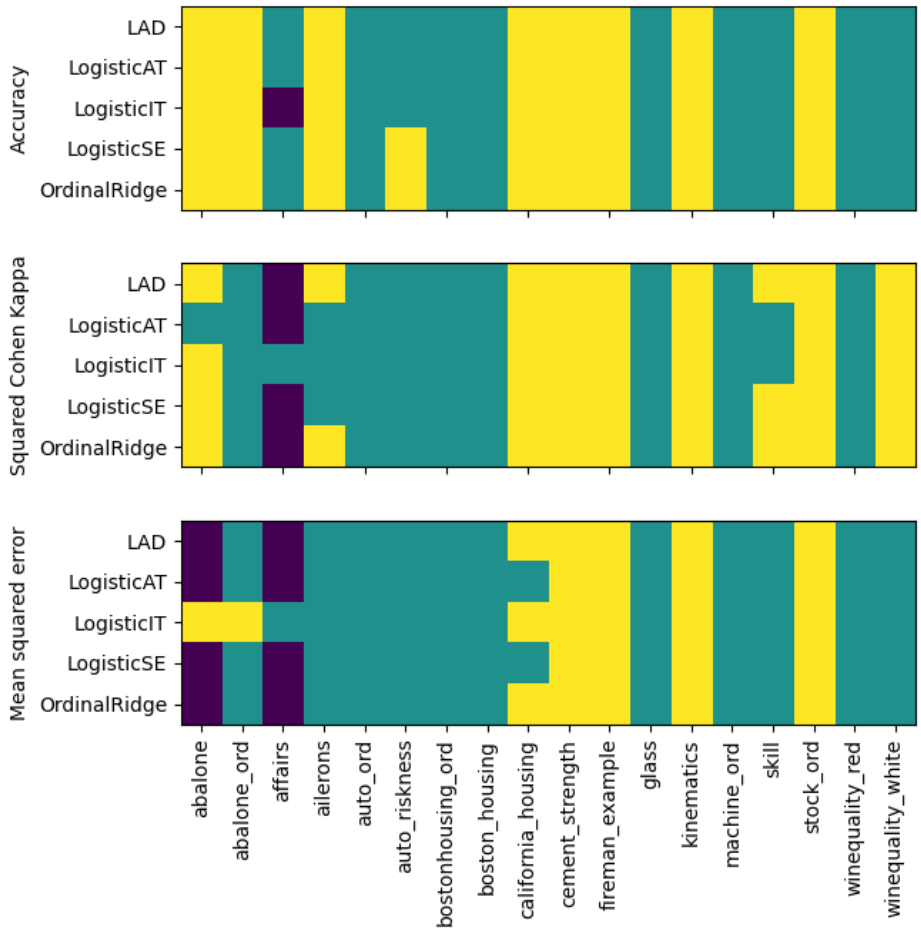


Fig. 1.3 Comparaison des résultats entre la méthode proposée et les méthodes baseline sur tous les jeux de données. Les cases jaunes, bleues et violettes correspondent à des méthodes baseline dont les performances sont significativement plus faibles, sans différence significative, ou significativement meilleures que la méthode proposée, respectivement

1.3 Méthodes d'apprentissages profond en épidémiologie et leurs applications en statistiques de mortalité

Les statistiques de mortalité par cause, publiée en France par le CépiDc, sont un outil précieux qui accompagne régulièrement la prise de décision dans les politiques de santé publique. Leur production est en revanche un procédé particulièrement long et couteux, pouvant parfois aller jusqu'à amoindrir leur pertinence, typiquement dans des cas où les délais de production deviennent prohibitivement longs. La production de ces statistiques commence par le renseignement par un praticien médical de l'enchainement d'évènements entraînant le décès du sujet sur un certificat, ceci en langage naturel (donc en français au CépiDc). Après réception du certificat par le CépiDc, celui-ci se voit appliqué deux étapes de traitement pour en extraire une cause initiale de décès, qui sera la statistique rapportée par le service annuellement :

- Convertir la chaine causale d'évènements menant au décès du sujet, renseignée par le médecin constatant le décès du langage naturel à une séquence de codes d'une classification médicale, la Classification statistique Internationale des Maladies (classification CIM-10)
- Identifier la cause initiale de décès, à partir de la chaine causale d'évènements menant au décès du sujet, exprimée en codes CIM-10

	Chaîne causale du décès du sujet (format textuel)	Format CIM-10	CI
Ligne 1	Cancer indifférencié de la glande thyroïde ayant entraîné une compression complète locale (sténose œsophagienne et paralysie bilatérale des cordes vocales)	C73 K222 J380	C73
Ligne 2	NA		
Ligne 3	NA		
Ligne 4	NA		
Partie 2	Diabète insulino-requérant, HTA, Dénutrition, Antécédent d'AVC, AOMI	E119 I10 E46 I696 I702	

Tableau 1.1: Exemple d'extraction de la cause initiale de décès à partir d'un certificat. Dans un premier temps, la chaine causale du décès du sujet est convertie en séquences de codes CIM-10. Dans un second temps, la cause initiale de décès est identifiée à partir de ces séquences de codes CIM-10 en appliquant un ensemble de règles définies par l'OMS

Les statistiques de mortalité étant exhaustives, il est nécessaire en France d'évaluer individuellement plus de 600 000 certificats par ans. Au fil des ans, des tentatives d'automatisation de ce procédé ont été proposées, la plupart basées sur des systèmes experts. Cependant, leurs performances ne permettent pas leur utilisation en autonomie complète, et ils sont donc utilisés en tant qu'outil d'assistance au codeur humain, rendant le processus de production de ces données long et onéreux. Les méthodes d'apprentissage profond étant réputées comme significativement plus performantes que les systèmes experts, on peut cependant se demander si leur application à ce problème ne pourrait pas mener à des gains de performance intéressants. Deux méthodes seront présentées pour adresser les deux tâches nécessaires à la production des statistiques de mortalité par cause. La première étape, d'identification des codes CIM-10 à partir du langage naturel, sera traitée comme une variante d'un exercice de traduction automatique qu'on choisira d'adresser avec un modèle de type Transformer. La seconde sera traitée comme un problème de classification classique et sera adressée par le biais d'un réseau de neurones à convolution.

1.3.1 Traduction neuronale et identification des entités médicales CIM-10 à partir du langage naturel

1.3.1.1 Matériel et méthode

Le jeu de données utilisé pour développer un modèle permettant de prédire les entités médicales CIM-10 à partir du langage naturel présent sur le certificat consiste en tous les certificats de décès présents dans les bases de données du CépiDc de 2011 à 2016, pour un total d'environ 3 millions de certificats. On dispose pour ces 3 millions de certificats de la chaine d'évènements menant au décès du sujet, à la fois en format textuel et encodée en CIM-10, ainsi que de diverses variables qui seront également inclues dans le modèle, à savoir l'année de décès, l'âge du sujet, son genre et la provenance du certificat (qui indique si le certificat a été remplis sous format papier, ce qui nécessite par la suite une

saisie par un sous-traitant, ou si le certificat a été rempli sous format électronique et directement envoyé au CépiDc). La variable à prédire étant une séquence de codes CIM-10, on considérera ce problème comme un exercice de traduction neuronale, auquel on appliquera le modèle le plus performant dans ce domaine connu à ce jour, l'architecture Transformer.

1.3.1.2 Résultats

Après une recherche d'hyper-paramètres et la constitution d'un ensemble, les performances du modèle ont été évaluées sur un jeu de données de test préalablement exclu de l'ajustement du modèle et réservé à cet effet. Ces performances peuvent être observées en table 1.2 et comparées à l'état de l'art actuel, obtenu par le LIMSI à partir d'une approche hybride entre un système expert et un modèle de type machine à vecteur de support. Le LIMSI n'ayant évalué leurs performances que sur les certificats électroniques, les performances du modèle sont rapportées à la fois pour les certificats électroniques uniquement, pour les certificats papier, et pour tout certificat. On constate une nette amélioration en termes de pouvoir prédictif du modèle.

Méthode	F-mesure	Précision	Rappel
Etat de l'art (LIMSI)	.825	.872	.784
Approche proposée (certificats électroniques)	**.952 [.946, .957]**	**.955 [.95, .96]**	**.948 [.943, .954]**
Approche proposée (certificats papier)	.942 [.941, .944]	.949 [.947, .95]	.936 [.934, .937]
Approche proposée (tous certificats)	.943 [.941, .944]	.949 [.948, .951]	.937 [.935, .938]

Tableau 1.2 F-mesure de l'état de l'art et de la méthode proposée, avec leurs intervalles de confiance à 95% obtenus par bootstrap.

1.3.2 Apprentissage profond et identification de la cause initiale de décès à partir du certificat de décès

1.3.2.1 Matériel et méthode

Le jeu de données utilisé pour développer un modèle permettant de prédire les entités médicales CIM-10 à partir du langage naturel présent sur le certificat consiste en tous les certificats de décès présents dans les bases de données du CépiDc de 2000 à 2016, pour un total d'environ 9 millions de certificats. On dispose pour ces 9 millions de certificats de la chaine d'évènements menant au décès du sujet, encodée en entités CIM-10, de la cause initiale de décès retenue soit par un codeur humain soit par le logiciel Iris, logiciel international de choix de la cause initiale dont l'utilisation est préconisée par l'OMS, et de diverses variables qui seront également inclues dans le modèle, à savoir l'année de décès, l'âge du sujet et son genre. La variable à prédire dans cet exercice, la cause initiale de décès, est considérée comme une variable qualitative munie d'autant d'états qu'il existe de codes CIM-10 correspondant à une cause initiale dans le jeu de données, à savoir environ 8000. Ce problème peut donc être considéré comme un problème de classification multinomiale, qu'on choisira d'adresser par le biais d'un réseau de neurones à convolution inspiré de l'architecture Inception v2.

1.3.2.2 Résultats

Après une recherche d'hyper-paramètres et la constitution d'un ensemble, les performances du modèle ont été évaluées sur un jeu de données de test préalablement exclu de l'ajustement du modèle et réservé à cet effet. Ces performances peuvent être observées en table 1.3 et comparées à l'état de l'art actuel, obtenu par le biais du logiciel Iris sur le même jeu de données. On constate une amélioration significative des performances en utilisant la méthode proposée.

Méthode	Exactitude
Etat de l'art (Iris)	$0.925\ [0.921, 0.928]$
Approche proposée	$\mathbf{0.978}\ [\mathbf{0.977}, \mathbf{0.979}]$

Tableau 1.3: Exactitude du logiciel Iris et du meilleur modèle prédictif obtenu, avec leurs intervalles de confiances à 95% obtenus par bootstrap

1.3.3 Sur la comparabilité des statistiques de mortalité par cause à l'international

1.3.3.1 Matériel et méthode

Dans une démarche de vérification de la comparabilité des statistiques de mortalité par cause à l'international, le modèle de la partie précédente a été réajusté sur un jeu de données international incluant 4 pays :

- L'Angleterre et le pays de Galles, dont tous les certificats de décès de 2005 à 2018 ont été inclus dans l'expérience, à l'exception des décès certifiés par un médecin légiste, totalisant 5,1 millions d'observations
- L'Italie, dont tous les certificats de décès de 2014 à 2016 ont été inclus dans l'expérience, à l'exception des certificats liés à des causes externes de décès, totalisant 1,8 millions d'observations
- Les U.S.A. dont tous les certificats de décès de 2000 à 2017 ont été inclus dans l'expérience, totalisant 45,5 millions d'observations
- La France, avec l'exact même jeu de données que dans l'expérience précédente

L'architecture du modèle et les modalités d'ajustement sont les mêmes que pour l'expérience de la partie précédente, avec toutefois l'inclusion d'une variable explicative en plus, le pays d'appartenance du certificat, considérée comme une variable qualitative à quatre états (un par pays). L'inclusion de cette variable constitue la clé de l'expérience, en postulant que si les statistiques de mortalités étaient

comparables à l'international, elle n'aurait aucun pouvoir prédictif. En particulier, il serait dans ce cas raisonnable de s'attendre à ce que changer la valeur de cette variable pour un certificat donné ne change pas la prédiction du modèle. On peut donc après ajustement du modèle obtenir différents jeux de prédictions pour un pays donné, interprétés comme s'ils étaient « codés comme un autre pays ». Observer les variations de pouvoir prédictif entre ces différentes prédictions permet de donner un aperçu exploratoire de la comparabilité des statistiques entre chaque pays.

1.3.3.2 Résultats

Après ajustement, le modèle fut appliqué aux jeux de tests de chaque pays en faisant varier l'état de la variable de pays d'origine, et les performances du modèle obtenu, par pays et par variation, peuvent être observées en table 1.4.

	Comme en France	Comme en Italie	Comme aux U.S.A.	Comme en Angleterre
France	97.3 [97.2, 97.4]	90.6 [90.4, 90.7]	94.1 [94.0, 94.2]	94.4 [94.3, 94.5]
Italie	90.0 [89.8, 90.3]	97.6 [97.5, 97.7]	91.1 [90.9, 91.3]	89.7 [89.4, 90.0]
U.S.A.	94.8 [94.7, 94.9]	90.5 [90.4, 90.6]	98.6 [98.6, 98.6]	96.8 [96.7, 96.8]
Angleterre	96.9 [96.8, 97.0]	96.3 [96.2, 96.4]	97.0 [96.9, 97.1]	99.1 [99.1, 99.2]

Tableau 1.4: Performances par pays (en ligne), en terme d'exactitude, en fonction de l'état de la variable "pays de codage" (en colonne). Les diagonales correspondent à la vraie exactitude du modèle pour chaque pays

On constate dans l'ensemble que la méthode utilisée pour prédire les certificats français s'exporte bien sur les jeux de données d'autres pays (avec des performances encore supérieures pour les pays anglo-saxons) et que les pertes en pouvoir prédictif du modèle en perturbant l'état de la variable origine du pays, quoique présentes, sont relativement faibles, le seul pays présentant une différence prédictive non négligeable étant l'Italie. En revanche, ce comportement anormal peut être expliqué par l'absence de cause externes de décès dans les certificats italiens, et présents dans tous les autres, ce que

semblent confirmer les variations de pouvoir prédictif du modèle évaluées en excluant les certificats liés à des causes externes de décès, qui sont beaucoup plus homogènes.

1.4 Conclusion

Au cours de cette thèse ont été présentés plusieurs exemples d'applications de méthodes d'apprentissage profond sur des bases de données médico-administratives, avec des résultats présentant un gain de performance significatif sur les précédents états de l'art. Ces modèles présentent d'immédiates applications réelles, de l'accélération de la production de données (avec par exemple l'automatisation du codage médical d'acte), dont l'intérêt a déjà été démontré avec notamment l'utilisation d'un modèle de Transformer dédié à la reconnaissance d'entités médicales à partir du langage naturel pour produire en temps réel des statistiques de comorbidités sur les décès COVID-19 pendant le premier confinement. De plus, les méthodes proposées dans cette thèse n'étant pas spécifiques aux données de mortalités ni aux problèmes de modélisations sur lesquelles elles ont été appliquées, elles constituent des pistes prometteuses pour des applications sur d'autres bases de données administratives, tout comme par exemple le Système National des Données de Santé.

2 Introduction

The exploitation of electronic health databases has gained a significant amount of interest in fields such as epidemiology or public health. Indeed, as repositories of numerous, oftentimes almost exhaustive, detailed patient's medical histories, they have the potential to provide public health decision makers with powerful insights on population health. However, the sheer size and complexity of these databases introduces specific difficulties, both technical and methodological, for their proper exploitation. On another side, the raise of machine learning, and most specifically deep learning in the past decade has led to the development of entire families of powerful models fit to analyse complex, non-linear interactions on massive and unstructured datasets, some of whom have already been successfully applied to electronic health database analysis. However, the difference in language between epidemiologists and biostatisticians on one side, and the newly arrived data scientist on the other, has made cooperation between these two complementary fields difficult[4]. The objective of this thesis is dual. First, this thesis is meant as an introduction to modern deep artificial neural network based models to epidemiologists and biostatisticians, by showing how these methods sometimes denoted as "algorithms" actually share a profound relationship with more traditional statistical predictive models commonly used in health sciences, such a linear or logistic regression. Finally, this thesis will showcase two practical applications of these methods on electronic health databases, with a focus on mortality statistics and epidemiology.

3 Predictive modelling and deep artificial neural networks

3.1 Predictive modelling and machine learning

Machine learning is a field of study focusing on giving computers the ability to perform predefined tasks without giving them explicit instructions on how to do so [5]. Formally, in a well-defined machine learning problem, "A computer program is said to *learn* from experience E with respect to some task T and some performance measurement P, if its experience on T, as measured by P, improves with experience E" [6].

This broad definition leads to several possible mathematical interpretations, and, consequently, to an equal number of subfields of machine learning, usually overlapping with several other disciplines from computer sciences, artificial intelligence, mathematics, and specifically statistical analysis. Indeed, a profound relationship seems to link the concepts of experience as defined from a machine learning point of view, and of data analysis, which is ubiquitous in statistics.

The statistical analysis of real dataset being at the core of modern biomedical sciences, it only seems natural to focus on machine learning subfamilies that blend elements of statistical analysis. Some of the most advanced deep neural networks can for instance be thought of as advanced expansion of fairly classical tools used in statistical analysis[7].

Supervised learning is among the most common machine learning disciplines. It shares deep roots with statistical analysis. For instance, linear regression and linear discriminant analysis, two widely known statistical modelling techniques, can be considered as supervised machine learning techniques. The goal of supervised learning techniques is to, given a dataset, model the relationship between one or several independent variables (the input or endogenous variables) and a predefined dependent variable [8] (the output or exogenous variables).

Two main distinct types of supervised machine learning can be identified, namely given the output's variable nature:

- Classification for categorical output variables: The learning algorithm is asked to find a surface function of the input variables that best separates the output variable's different states.
- Regression for quantitative variables: The learning algorithm is asked to find a function of the input variables that best fits the output variables

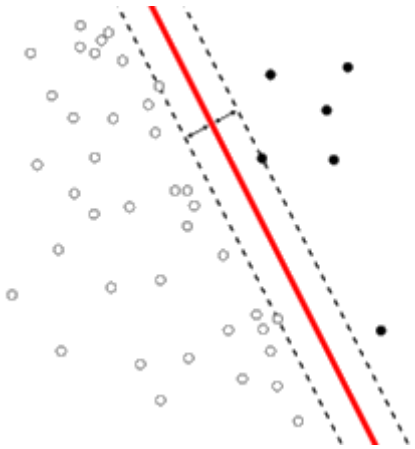


Fig. 3.1.1 Example of supervised learning method: Margin classifier [9]. The learning algorithm determines the straight line that separates the black and white dots that maximize the distance from the lines to the data points

These methods can easily be linked to the previously seen definition of machine learning:

- Experience E: The gathered dataset
- Task T: Correctly estimate the output variable given the input variables
- Performance P: model accuracy on the data

3.2 Neural networks

Artificial neural networks are a family of machine learning algorithms loosely inspired by biological neural networks. They are nowadays crucial components in most artificial intelligence related technologies such as speech or image recognition, self-driving cars or machine translation [10]. Their current popularity comes from their unprecedented ability to learn highly complex and nonlinear concepts by composing together a multitude of much simpler functions [7].

Artificial neural networks are quite versatile and can be adapted to a number of learning paradigm. Indeed, although they are mostly used for supervised learning, where they share a number of common characteristics with well-known statistical methods such as linear or logistic regression, they can be adapted to perform in supervised or unsupervised settings.

3.2.1 Linear regression

Linear regression is a widely used modelling technique consisting of modelling datasets with a linear, multivariate approximation of the real, studied phenomenon [11]. This simple approach also shares similarities with regressive feedforward neural networks and make as such a good introduction to understand them.

Let $\{(X_i, y_i)\}_{0<i<N+1}$ be a set of n observations (typically a multivariate dataset collected for an experimental study), with X_i and y_i being the i^{th} sample's input and output variables, respectively.

The objective of linear regression modelling is to find $\hat{y}$, a linear function of the input variable parameterized in (W, b) that best predicts the output variables, the concept of best predictive approximation being formalized by a cost function L, that quantifies how close to the observed dataset the approximation is (for instance the least square function in the case of linear regression).

$$\hat{y}_{W,b}(X) = (W)^T \cdot X + b \qquad (3.2.1),$$

$$L(W,b) = \frac{1}{2N} \cdot \sum_{i=1}^{N} \left(\hat{y}_{W,b}(X_i) - y_i\right)^2) = \frac{1}{2N} \cdot \sum_{i=1}^{N} ((W)^T \cdot X_i + b - y_i)^2) \quad (3.2.2).$$

The regression model is extracted by the data by finding $\hat{y}_e$ such that:

$$\hat{y}_e(X) = (W_e)^T \cdot X + b_e \quad (3.2.3),$$

$$with \quad (W_e, b_e) = \underset{(W,b)}{\operatorname{argmin}}(L(W,b)) \quad (3.2.4).$$

In the case of linear regression, the parametric solution (W_e, b_e) can be determined either analytically, or through the use of an optimization algorithm such as gradient descent. Finding the parameters that minimize a specific cost function is common to a number of machine learning algorithms, for which it is called the training process. Linear models are known both for being be easy to fit, as well as for their good generalization behavior. As is, however, they fail to capture any non-linearity relationship between input and target variables, as well as input variables interactions that would typically require additional treatment such as nonlinear basis expansion, for instance.

3.2.2 Logistic and softmax regressions

Logistic and softmax regression models are regressive models dedicated to the analysis of a binary, categorical output variables (two states variables such as 0/1, ill/healthy, pass/fail) and multimodal categorical variables (number of states >2), respectively [12]. They are widely used in statistical analysis as part of the family of generalized linear models and can be used from a machine learning perspective to perform binary linear classification.

The process of extracting a logistic regression model from a dataset is fairly similar to what is done in linear regression models. The objective of a linear regression model is to find a parametric function linking the input and output variables by minimizing a cost function of the model's parameters. However, the output variable's categorical nature in logistic regression requires a subtle change in the

problem's formalization. The objective is to link the input variables to the probability of the output variable being in one state or another, implicitly obtaining the output variable by taking the most probable option.

Let $\{(X_i, y_i)\}_{0<i<N+1}$ be a set of n observations (typically a multivariate dataset collected for an experimental study), with X_i and y_i being the i^{th} sample's input and categorical output variables, respectively.

The objective of logistic regression is to find a parametric function of the input variables that outputs an estimation function h of the output variable's probability to be in one state or the other, such that the output variable is best predicted, the concept being quantified through the formalization of a cost function L called cross entropy.

$$h_{W,b}(X) = \frac{1}{1 + e^{-((W)^T \cdot X + b)}} \quad (3.2.5),$$

$$\hat{y}_{W,b}(X) = \begin{cases} 1 & if \ \ h_{W,b}(X) > 0.5 \\ 0 & if \ \ h_{W,b}(X) < 0.5 \end{cases} \quad (3.2.6),$$

$$L(W,b) = -\frac{1}{N}\sum_{i=1}^{N} [\, y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \,] \quad (3.2.7).$$

The regression model is extracted by the data by finding h_e, and consequently $\hat{y}_e$, such that:

$$h_e(X) = \frac{1}{1 + e^{-((W_e)^T \cdot X + b_e)}} \quad (3.2.8)$$

$$with \ \ (W_e\,, b_e) = \underset{(W,b)}{\operatorname{argmin}}(\, L(W,b)) \quad (3.2.9)$$

The model's parameters $(W_e\,, b_e)$ can be determined through the use of an optimization algorithm such as gradient descent.

The process of fitting a softmax regression to a dataset with a M-states multimodal categorical variable is essentially the same as for a logistic regression, with a slight adaptation of the parametric function to account for the different variable states:

$$\forall\, j \in [\![1;\, M]\!], \qquad h_{W_j,b_j}(X) = \frac{e^{-((W_j)^T \cdot X + b_j)}}{\sum_{i=1}^{K} e^{-((W_i)^T \cdot X + b_i)}} \tag{3.2.10},$$

$$\hat{y}_{W,b}(X) = \underset{j \in [\![1;\, M]\!]}{\operatorname{argmax}}(\, h_{W_j,b_j}(X)) \tag{3.2.11}.$$

Similar to linear regression, logistic and softmax regressions are considered linear classification algorithms and cannot take account of potential non-linearity and variable interactions in the dataset.

3.2.3 Optimization and Gradient descent

In both linear and logistic regression, model fitting requires to find the minimum of a function of the model's parameters, called the cost function. The problem of finding the minimum of the least square cost function in a linear regression model is relatively straightforward, as it can be explicated analytically. This is not necessarily the case when using logistic regression. As a consequence, the use of optimization algorithms is usually necessary to obtain an approximate, computational solution.

A number of optimization algorithm are available to the practitioner, who chooses from them according to the problem at hand in a case by case manner. Gradient descent is a well-known optimization algorithm that is consistently used in machine learning to minimize cost functions.

Gradient descent is an iterative optimization algorithm that explores functions in a search for a local minimum by making use of its first-order derivatives in the form of the function's gradient. The gradient of a function f from $\mathbb{R}^n$ to $\mathbb{R}$ is defined such that $\forall(x_1, -, x_n) \in \mathbb{R}^n$:

$$\nabla(f(x_1, -, x_n)) = (\frac{\partial f}{\partial x_1}\, , -\, , \frac{\partial f}{\partial x_n}) \tag{3.2.11}.$$

The general idea behind gradient descent is to first evaluate the function and its gradient at a random location, identify the direction in which the function is "going down the most", and take a step in that direction. After a number of steps, the gradient will evaluate close to 0, meaning that the location in which the function is being evaluated is "flat", meaning that there is no immediate local neighborhood to explore providing lower values of the investigated function. As a consequence, the final position is a local minimum. More formally, for a function f from $\mathbb{R}^n$ to $\mathbb{R}$, $a_0 \in \mathbb{R}^n$ a random vector and two real number γ and S, a gradient descent algorithm is based on iterating:

$$\forall n \in \mathbb{N}, \quad a_{n+1} = a_n - \gamma \nabla f(a_n) \quad as\ long\ as\ \nabla f(a_n) > S \qquad (3.2.12).$$

With sufficiently low values for S, (a_n) progressively converges toward a function's local minimum. During the practical implementation of a machine learning algorithm, the step size (or learning rate in machine learning terminology) γ needs to be fine-tuned by the machine learning practitioner. Its value has a significant impact on the optimization process, as it sets the size of the steps taken by the algorithm at each descent iteration. As a consequence, it needs to be chosen carefully. Indeed:

- Too big a learning rate value will lead to an unstable gradient descent. The steps taken at each iteration are too large to make use of the local information given by the gradient computation. A gradient descent with too big a learning rate might not converge, and can sometime, especially with deep neural network, diverge
- Too small of a learning rate will lead to an exceedingly slow optimization process. Each step will be taken in the right direction, but the new point will be located right next to the previous one, thus making close to no progress in finding a minimum.

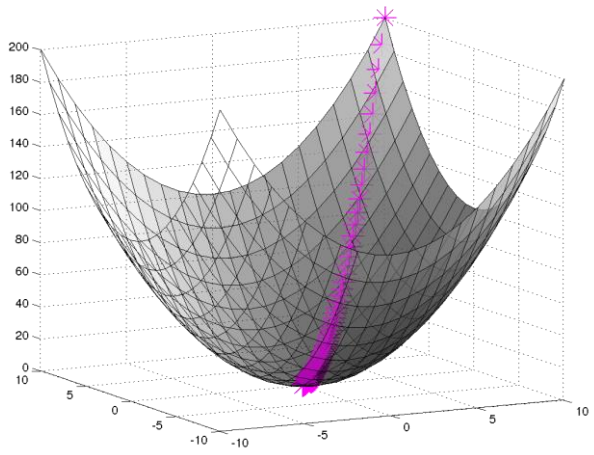


Fig 3.2.1 Example of gradient descent on a convex function. The gradient descent's steps (in purple) begins at a random point on the function's surface. Each step iteratively brings it closer to the function's minimum, where the gradient become null.

Finding a function's minimum using gradient descent requires certain hypothesis from the investigated function. Mainly:

- The investigated function needs to be defined and differentiable to perform gradient descent (these hypotheses can be weakened, but neural network all present those properties)
- The investigated function needs to be convex for the algorithm to find the function's global optimum. In other cases, gradient descent cannot provide any warranty regarding the identified local optimum's quality

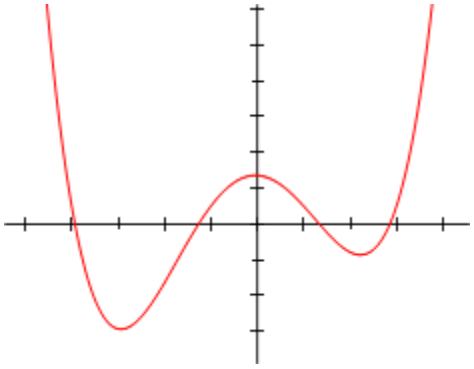


Fig 3.2.2 Example of non-convex function presenting two local minima. The left-hand minimum is the global optimum. However, a gradient descent algorithm initialized with a positive value will fall into the right-hand minimum, which is suboptimal

3.2.4 Multilayer perceptron

Multilayer perceptron are among the most widespread deep learning models [7]. They can be used in a variety of modelling tasks, such as regression or classification, where then can be seen as nonlinear expansions to respectfully traditional mean square linear regression and logistic regression.

The idea behind feedforward neural networks is to fit a linear model to a transformed set of observations $\{(\phi(X_i), y_i)\}_{0<i<N+1}$ where not only the model's parameters, but also ϕ, are learnt from the data.

The traditional approach to enable a neural network model to learn non-linear interaction from the data is to inject linear combination of the investigated parameters into simple, nonlinear functions, whose outputs are then either used to perform a linear modelling task, or as inputs to be injected in another set of nonlinearities.

The toy network presented in figure 3.2.3 is an example of a feedforward neural network with one hidden layer, used in a regression setting. As in linear regression, the objective is to fit a set of N observations $\{(X_i, y_i)\}_{0<i<N+1}$ with a parametric, non-linear approximation:

$$\hat{y} = w_{11}^3 \cdot f_1(w_{11}^2 \cdot x_1 + w_{12}^2 \cdot x_2 + b_1^2) + w_{12}^3 \cdot f_2(w_{21}^2 \cdot x_1 + w_{22}^2 \cdot x_2 + b_2^2) + b_1^3 \quad (3.2.13).$$

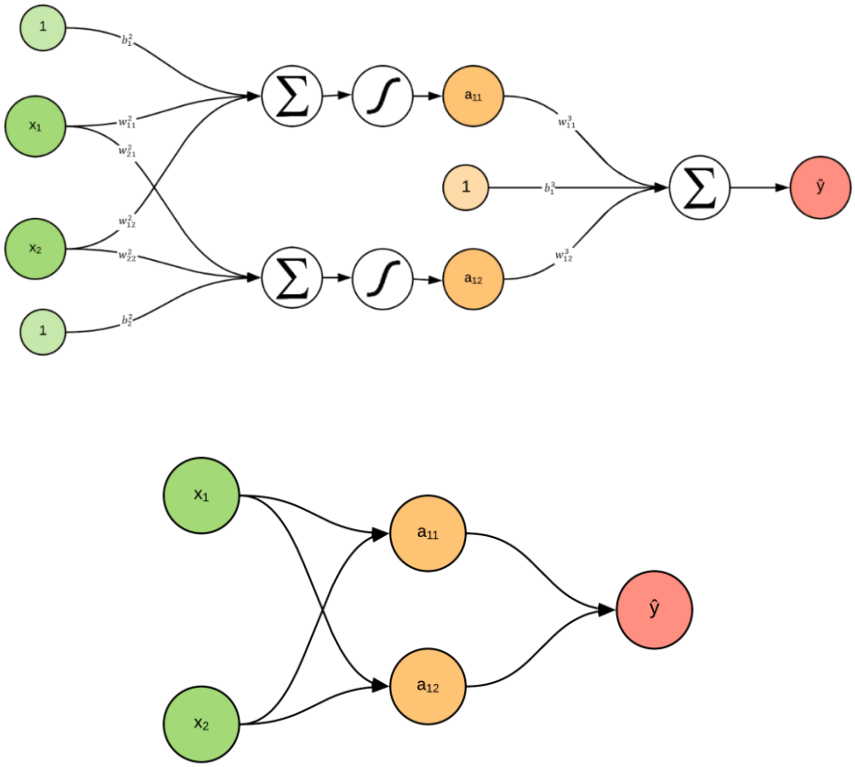


Fig. 3.2.3: Top: simple example of a feedforward neural network in regressive settings. Bottom: The same network in its traditional, simplified representation. The biases are omitted and the linear combinations and injections in nonlinearities are done implicitly for ease of notation

While the parameters are typically obtained from the minimization of the square loss function, two differences arise from the introduction of parametric non linearities in the model:

- The parametric solution that minimizes the cost function cannot be expressed in closed form, and must be estimated through gradient based optimization
- The cost function's gradient is not straightforward to compute

- The resulting optimization problem loses its convexity, and the warranty of finding the global minimum

The entire family of feedforward neural network can then be derived from this toy example, mainly through the variation of the cost function and three properties of the network:

- Cross entropy or least square cost function for classification and regression respectively
- Number of composed function in chain (number of layer)
- Number of newly build feature per layer (number of neurons)
- Type of nonlinearity applied to the neurons output (i.e. choice of functions f^1 and f^2)

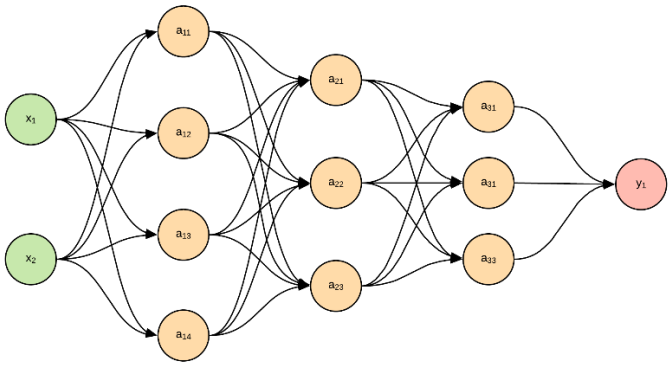


Fig. 3.2.4 Neural network model with three hidden layers (in orange). The green and red nodes are respectively the model's input (endogenous variables) and output (exogenous variables

These network properties, however, define the functional family across which the optimization algorithm will perform its search, and as such cannot be learnt during the training process. As such, they constitute what are called model hyper-parameters, and require additional fine-tuning that can

be critical to the model's performance, and which is typically done through a simple search process on a dedicated validation dataset, as will be described further down in part 3.3.5.

Increasingly complex models can be built by tuning these hyper-parameters, thus designing a model most fitted to the problem's complexity.

Theoretically, any type of nonlinearity *f* can be applied to the neurons' outputs. In real life applications, however, practitioners usually choose between a limited number of options that have been empirically proven to perform well, with the most notable example being the linear rectified unit [13] defined for $x \in \mathbb{R}$ as:

$$ReLU(x) = \begin{cases} x & if\ x > 0 \\ 0 & otherwise \end{cases} \quad (3.2.14).$$

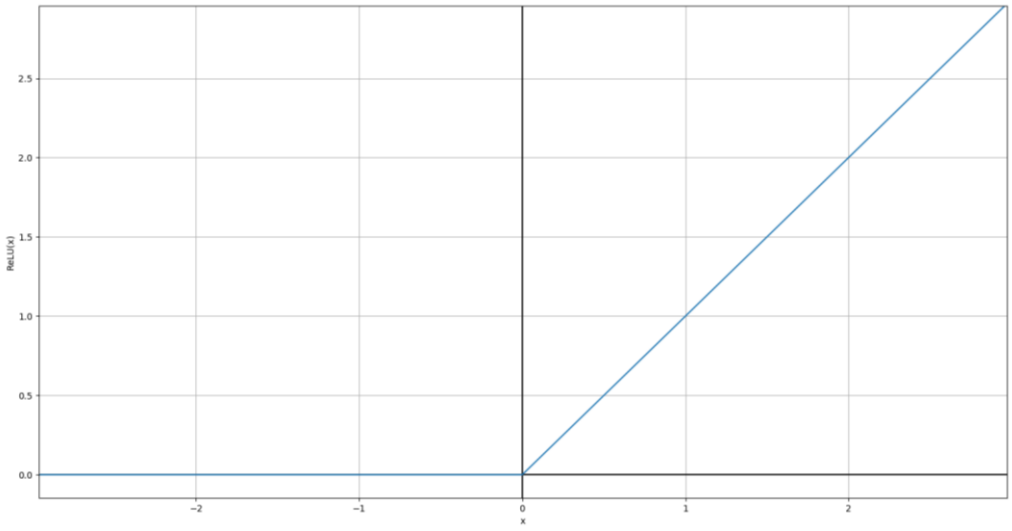


Fig 3.2.5 Plot of the ReLU non linearity

While the linear rectified unit is currently one of the most widely used nonlinearity in the deep learning academic literature, a lot of work has been done toward trying to improve on its design, and has given

birth to a family of linear rectifier type units, as for instance the leaky rectified linear unit (LReLU) [14] defined for $x \in \mathbb{R}$ as:

$$LReLU(x) = \begin{cases} x & if\ x > 0 \\ 0.1x & otherwise \end{cases} \quad (3.2.15).$$

The LReLU only differs from the traditional ReLU for a negative x, where it is given a strictly negative slope. The rationale behind this subtle change is related to the optimization process and more specifically the network's gradient shape. Indeed, a traditional ReLU displays a null derivative for any negative x, which can potentially hurt the gradient descent process. The LReLU prevents this problem by giving the nonlinearity a relatively small (but nonzero nonetheless) slope on the $\mathbb{R}^-$ domain. The subsequently obtained nonlinearity displays a closely similar behavior as the ReLU, but offers a nonzero derivative, potentially improving the gradient based optimization procedure used during the learning process [14].

In addition to the nature of the output variable, the only difference between a deep neural network in regressive or classification setting is its associated cost function, which does not in itself have any impact on a neural network's architecture. As a consequence, and for simplicity of writing, every neural network further described will be so in regressive settings. Its classifier counterpart can systematically be derived by changing the least square objective function by the cross-entropy cost function.

3.2.5 Backpropagation algorithm

In a similar fashion to statistical parametric models, the extraction of a neural network-based model from an investigated dataset is done through the computation of the solution minimizing a predefined cost function across the entire family of investigated parametric model (such as for instance the set of all straight line in univariate linear regression models). Although simple models such as linear

regressive models allow for a closed form expression of these optimal solutions, they are significantly more complex to derive when it comes to artificial neural networks such as the multilayer perceptron. A number of approaches have been devised over the years to automatically tune the model's parameter to best fit the observed data, typically through gradient based optimization methods.

As mentioned before, finding a given function's minimum through a descent-based optimization algorithm relies on an iterative process involving the computation of the function's gradient at each step. The strongly composed nature of the multi-layer perceptron's parametric form, especially when given a large number of layers, can make the computation of its gradient's manual implementation cumbersome. In addition, the network's architecture, which can significantly vary during the practical implementation of a neural network, has a direct influence on the gradient's shape. As a consequence, manual implementation of the gradient's computation is time-consuming and prone to errors. The backpropagation algorithm [15] was developed as a general method allowing for efficient cost function's gradient computation in any kind of feedforward neural network such as the multilayer perceptron.

Whether for classification or regression analysis, a machine learning's objective function is built to quantify the difference between the investigated dataset's output variable and its proposed approximation obtained through the model, for every available subject independently

Conceptually, this cost function defines the idea of distance between the observed data samples and the model by observing the difference between every single sampled example and their associated predictions, to then average over the whole dataset. As a consequence, both the cross entropy and least square cost function are written as a sum over a sample-wise error function applied to every example present in the dataset (and potentially a multiplicative factor):

$$L(W, b) = \alpha \sum_{i=1}^{N} L_{ind}(W, b, X_i, y_i) \quad (3.2.16).$$

The cost function's gradient can thus be written, because of the gradient's linear properties:

$$\nabla(L(W,b)) = \nabla\left(\alpha \sum_{i=1}^{N} L_{ind}(W,b,X_i,y_i)\right) \quad (3.2.17),$$

$$= \alpha \sum_{i=1}^{N} \nabla(L_{ind}(W,b,X_i,y_i)) \quad (3.2.18).$$

As a consequence, computing the entire cost function's gradient can be directly obtained by computing the individual error function's gradient for every candidate present in the dataset. This interesting property shared by many of the traditionally used cost functions is used in the backpropagation algorithm to provide an efficient procedure to compute their gradients by evaluating gradient contributions from every sample and combining them into the global gradient. The interest of this approach is dual:

- The sub gradients can be evaluated in a fairly straightforward approach using derivative chain rules
- As the computation of gradient contribution by every sample is completely independent, they allow for a powerful parallel implementation that significantly reduce the required computation time

For any observation (X_i, y_i) from the investigated dataset, given a multi-layer perceptron with n hidden layers of $\left(m^{[1]}, -, m^{[n]}\right)$ neurons respectively, parameterized with weights and biases real-valued vectors $\left(W^{[1]}, -, W^{[n]}, b^{[1]}, -, b^{[n]}\right)$ and non-linearities $(g^{[1]}, -, g^{[n]})$ and a cost function L adapted to the investigated modelling problem, the computation of the gradient of L with respect to the weight and biases $\left(\frac{\partial L}{\partial W^{[1]}}, -, \frac{\partial L}{\partial W^{[n]}}, \frac{\partial L}{\partial b^{[1]}}, -, \frac{\partial L}{\partial b^{[n]}}\right)$ can be obtained with the following procedure:

1. Use the current parameter values to compute the model's approximation of the current data sample's output variable (forward propagation), while keeping track of every layer's pre-activation vector (the neurons' outputs before injection in a nonlinearity) $(Z^{[1]}, -, Z^{[n]})$
2. Use the approximated values to compute the cost function's value
3. Directly compute $\frac{\partial L}{\partial A^{[n]}}$, the cost function's derivatives with respect to the network's last layers activations.
4. Starting with the n[th] layer and proceeding in a backward fashion, compute the cost function's derivatives with respect to the l[th] layer's parameter using $Z^{[l]}$, $\frac{\partial L}{\partial A^{[l]}}$ and the following set of equations [15]:

$$\frac{\partial L}{\partial Z^{[l]}} = \frac{\partial L}{\partial A^{[l]}} \odot g^{[l]\prime}(Z^{[l]}) \quad (3.2.19),$$

$$\frac{\partial L}{\partial W^{[l]}} = \frac{1}{m^{[l]}} \frac{\partial L}{\partial Z^{[l]}} \cdot (A^{[l-1]})^T \quad (3.2.20),$$

$$\frac{\partial L}{\partial b^{[l]}} = \frac{1}{m^{[l]}} \sum_{i=1}^{m^{[l]}} (\frac{\partial L}{\partial Z^{[l]}})_i \quad (3.2.21),$$

$$\frac{\partial L}{\partial A^{[l-1]}} = W^{[l]T} \cdot \frac{\partial L}{\partial Z^{[l]}} \quad (3.2.22).$$

An interesting property of this procedure, that gives it its name, lies in the fact that at each gradient descent iteration, the neural network currently trained is used twice:

- Explanatory variables are injected in the network's inputs and are propagated through the entire network to obtain output estimations. This step is called forward propagation
- The cost function's gradient is injected at the end of the network and is propagated in backward manner through the network to obtain derivative with respect to every parameter. This step is called backward propagation.

In present times, most deep learning programming libraries incorporate automated procedures to compute a neural network's gradient using the backpropagation procedure and allow for parallel implementations given appropriate hardware (typically high-end Graphics Processing Units).

3.2.6 Recurrent neural networks

3.2.6.1 Naïve (or vanilla) recurrent neural networks

Recurrent neural networks were the first implementation example of an idea that would end up creating most of the powerful neural architectures that have led to the dawn of the deep learning era, which is exploiting a modelling problem's symmetries by utilizing weight sharing.

Recurrent neural networks are a family of neural network that specialize in the analysis of sequential data[16]. The main idea behind the elaboration of a recurrent neural network is to devise a model that shares its parameter across all time steps within the data sequence. Instead of feeding the whole sequence to a standard perceptron, each time step in the data is sequentially fed to the network, which also takes as input its previous output in order to allow the model to condition both on the present and past observations as can be seen on figure 3.2.6. As a recurrent neural network requires this past connection for each time steps, an additional input is given to the model when evaluating the first sequential observation. This vectorial input is called an initial state and is typically either set to 0 or considered as learnable parameters for the model[17].

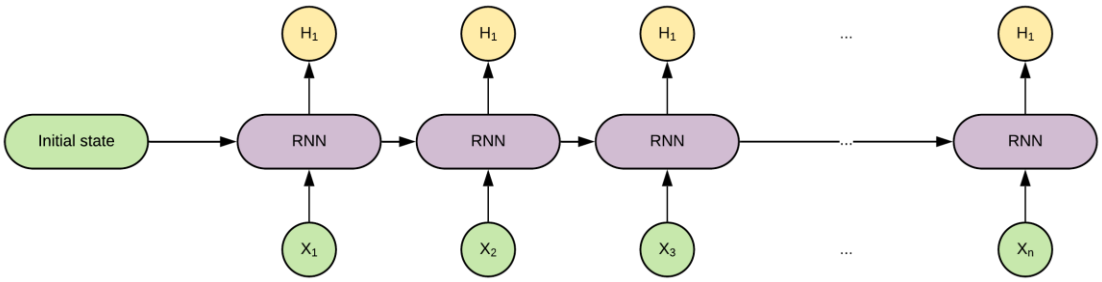


Fig. 3.2.6: A recurrent neural network architecture outputs a vector for each sequential observation that depends on both the current and all previously observed time-steps

This family of neural network can be used in a variety of settings that can be broadly gathered into 3 main categories that can be seen on figure 3.2.7:

- Modelling a non-sequential response variable from sequential explanatory variables (e.g. Text classification)
- Modelling a sequential response variable from sequential explanatory variables (e.g. Optical character recognition)
- Autoregressive modelling of a sequential variable (e.g. Language models)

These architectures can be fitted just as traditional feedforward neural networks using an adaptation of backpropagation called backpropagation through time. However, this naïve approach to recurrent neural networks is notorious for its poor behaviour during model fitting. As a consequence, several better behaved architectures have been devised as a replacement, such as the Gated Recurrent Unit or the Long Short Term Memory cell, the latter being the most wildly used.

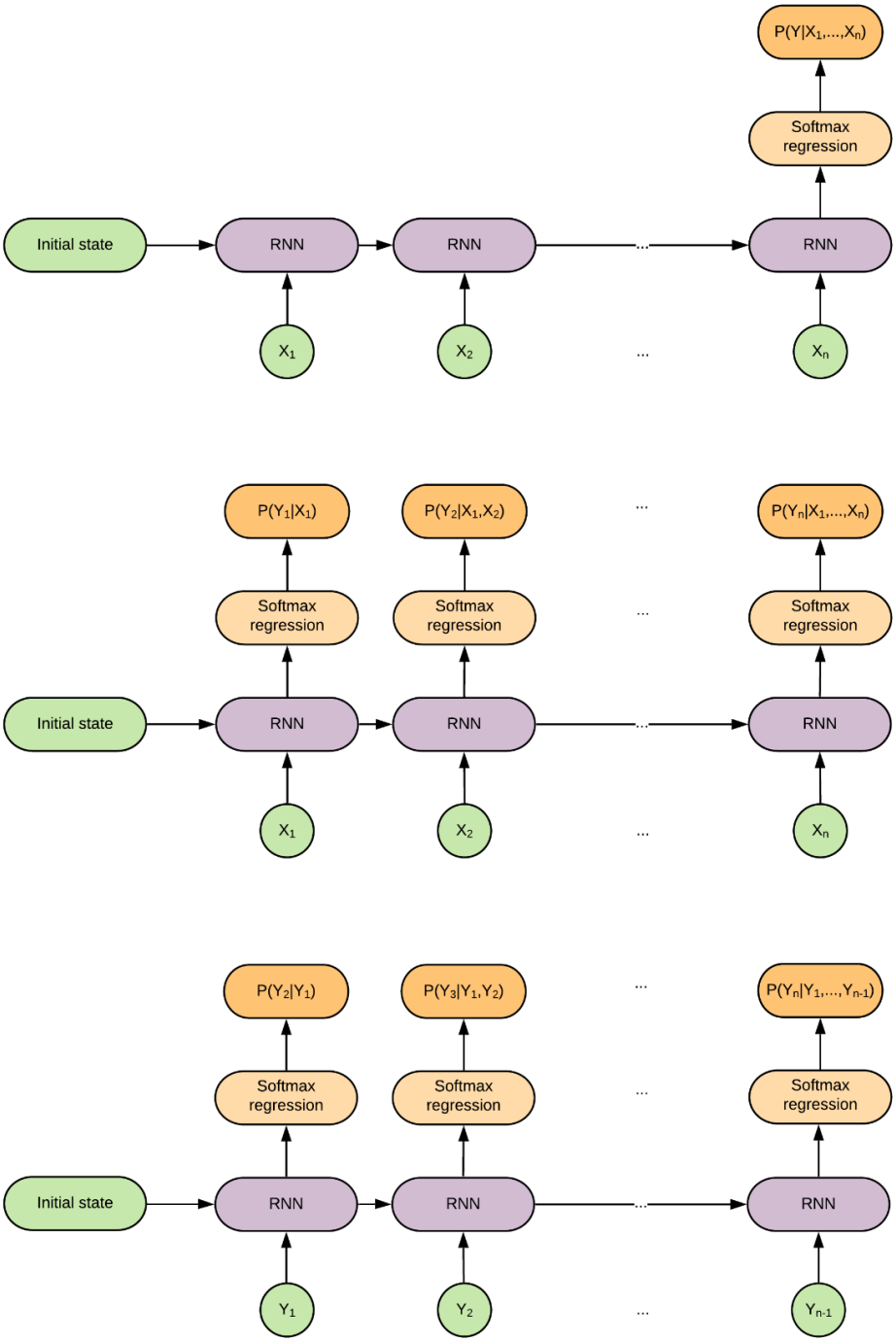


Fig. 3.2.7: Top: Recurrent neural network in regression setting. The RNN sweeps the entire input sequence, and its last output is used as inputs in a regression model conditioned on the entire sequence. Middle: RNN in sequential regression setting. The RNN sweeps the entire input sequence, and all of its outputs are used to fit a sequence of regression models conditioned on all previous observations. Bottom: RNN in autoregressive setting. The model sweeps the entire input sequence, and all of its outputs are used to fit a regression model to predict the next input from all previous observations

3.2.6.2 Long-Short-Term-Memory Cell

The Long-Short-Term-Memory cell [18] is a variation of a traditional recurrent network, and defines the parametric function f defined for the proposed approach to regression on sequential data as:

$$f_t = \sigma_g(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f) \quad (3.2.23),$$

$$i_t = \sigma_g(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i) \quad (3.2.24),$$

$$o_t = \sigma_g(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o) \quad (3.2.25),$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_g(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f) \quad (3.2.26),$$

$$h_t = o_t \circ \sigma_h(c_t) \quad (3.2.27),$$

with :

- x_t as the input vector
- h_t as the output vector
- c_t as the cell state vector
- $W, U, and\ b$ the cell's parameter matrices and vector
- σ_g the logistic function
- $\sigma_c\ and\ \sigma_h$ the hyperbolic tangent function
- $\cdot$ the dot product
- $\circ$ the Hadamard product

3.2.6.3 Gated Recurrent Unit

Gated recurrent units constitute another example of a variation on the naïve recurrent neural network. Although the LSTM cell has been shown to have a better modelling capability than the GRU, the latter was selected in the proposed approach for both its lesser amount of parameter and ability to handle smaller datasets. For a sequence of real valued vectorial input $(x_1, \dots, x_n)$ the output h_t at time step t of a GRU is defined from both h_{t-1}, x_t as follows[19]:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \quad (3.2.28),$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \quad (3.2.29),$$

$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \quad (3.2.30),$$

$$h_t = (1 - z_t) \odot \hat{h}_t + z_t \odot h_{t-1} \quad (3.2.31),$$

with:

- x_t the input vector
- h_t the output vector
- $\hat{h}_t$ the candidate activation vector
- z_t the update gate vector
- r_t the reset gate vector
- $\{W_i, U_i, b_i \; \forall \; i \in (z, r, h)\}$ learnable parameter matrices and vectors

3.2.7 Convolutional neural networks

Recurrent neural networks allow for the modelling of sequential datasets by sharing their neurons' parameters across all time-steps and the addition of a "memory" to the network. The idea of several neurons sharing the same parameters is a powerful concept that can be adapted to a number of model architectures, in order to extract powerful predictive models from structured datasets. Convolutional

neural networks offer a different approach to parameter sharing that was developed for computer vision task and the analysis of images as real valued 2-dimensional grids [20]. Indeed, multilayer perceptron-based models could be fit to such a data type, but similar to sequence datasets, the high number of variables contained in an image make for a combinatory explosion which usually prevent their use.

To prevent this problem, convolution based neural networks focus on the modelling of local information, by limiting the neurons' access to the input variables to small sub-grids (typically 2*2 to 5*5 areas) of the entire matrix. Each neuron is then copied across the entire image, to output not only one scalar activation, but a 2-dimensional array called an activation map. Similar to multilayer perceptrons, every neuron's activation map can then be injected into additional, neural layers.

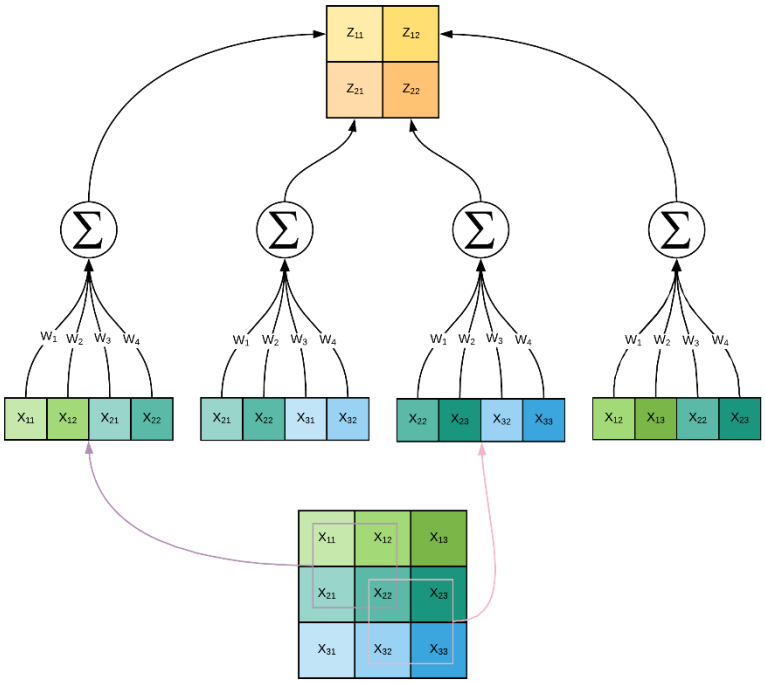


Fig. 3.2.8 Example of a convolution neuron applied to a 3x3 matrix. The neuron has access to limited 2x2 sub grids but sweeps the entire input grid. As a consequence, the neuron outputs several real values, instead of a single one

Although copying a neuron with limited connections to its input data presents a powerful way to make use of the idea of parameter sharing for the analysis of grid-like objects, its practical implementation as is can quickly become quite cumbersome. Indeed, naively copying the same neuron across the entire input data structure makes for a substantial amount of memory used to store redundant information. However, the transformation linking the input grid and the resulting activation map depending on one neuron's parameter is actually equivalent to a well-known signal processing tool, called cross-correlation. Through a slight modification of the traditional representation of an artificial neuron, this transformation can be used to implement a computationally efficient version of the aforedescribed neural architecture. In the deep learning academic literature, the cross-correlation operation is wrongly called a convolution operation. Although different transformations, the two share similarities and symmetries that make them yield similar results when used in neural network. As a consequence, the neural networks making use of this sub grid approach are called convolutional neural networks, in spite of using a cross-correlation operation. In order to prioritize consistency with the deep artificial neural network literature, the following formal description of convolutional neural network will make use of this designation, and the convolution operation defined further will be equivalent to the signal processing's cross correlation.

Even though convolutional neural networks were specifically designed for the analysis of image-based datasets as 2-dimensional grids, it can easily be adapted for use on sequential datasets [21], by considering a sequence of a variable as a grid of length the sequence's length and unitary width. Its applications in fields such as text or vocal analysis-based classification is actually a growing area of research [22].

3.2.7.1 Convolution operation

The convolution operation is a function that takes two grid objects as entries, with one usually smaller than the other and outputs another grid that gathers the results of the sum of the element wise products of the smaller grid and every similar sized sub-grid from the bigger grid. Formally, for $\boldsymbol{A} \in$

$\mathbb{R}^{n\times n}, \boldsymbol{B} \in \mathbb{R}^{m\times m}$ with $(n,m) \in \mathbb{N}^2$ such that $n > m$, the convolution of $\boldsymbol{A}$ and $\boldsymbol{B}$ noted $\boldsymbol{A} * \boldsymbol{B}$ is a matrix of size $(n-m+1)\times(n-m+1)$ defined as:

$$\forall i,j \in [\![1, n-m+1]\!],\ \ (\boldsymbol{A} * \boldsymbol{B})_{i,j} = \sum_{k=1}^{m}\sum_{l=1}^{m} (\boldsymbol{A})_{i+k-1,j+l-1}(\boldsymbol{B})_{k,l} \tag{3.2.32}$$

To better understand the properties of this operation, as well as its relationship with the neural architecture of the convolutional neural network, the same example used previously can be considered. Let X and W be 3 by 3 and 2 by 2 real valued matrices, respectively. The convolution of X and W can then be written as a 2 by 2 matrix such that:

$$\forall i,j \in [\![1,2]\!], \qquad (X * W)_{i,j} = \sum_{k=1}^{2}\sum_{l=1}^{2} (X)_{i+k-1,j+l-1}(W)_{k,l} \tag{3.2.33}$$

$$(X * W)_{i,j} = (X)_{i,j}(W)_{1,1} + (X)_{i,j+1}(W)_{1,2} + (X)_{i+1,j}(W)_{2,1} + (X)_{i+1,j+1}(W)_{2,2} \tag{3.2.34}$$

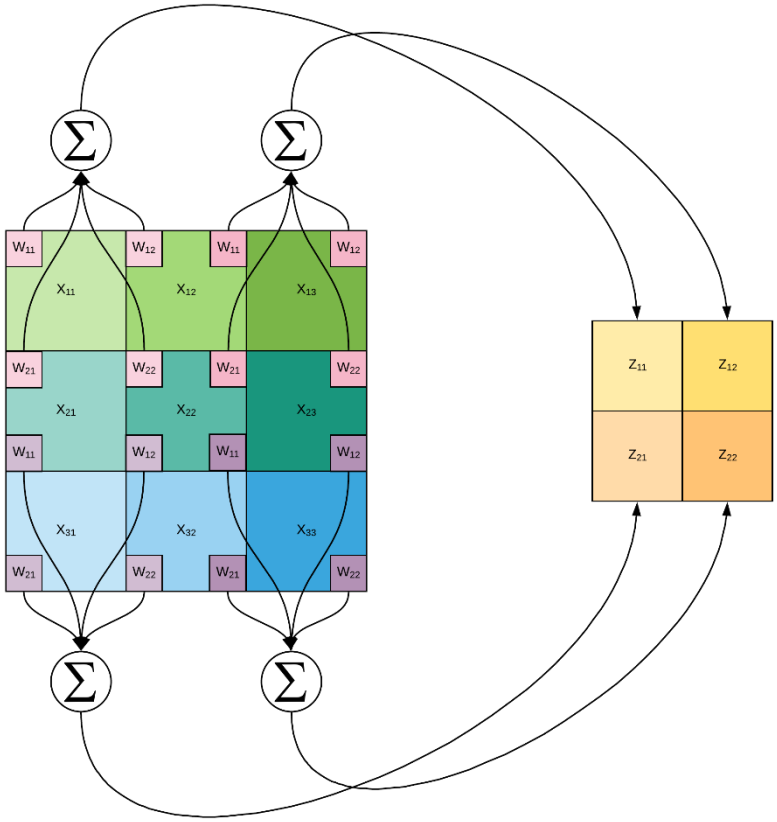


Fig 3.2.9 Visualization of the convolution of 3 by 3 and 2 by 2 matrices $\boldsymbol{X}$ and $\boldsymbol{W}$, and its similarity to the convolution neuron architecture

On another hand, when considering a 3 by 3 grid matrix input variable $\boldsymbol{X}$ and a neuron given limited access to 2 by 2 sub grids from it (and as such having a 4-dimensional weight vector), the neuron's pre-activation map can be written as a 2 by 2 matrix such that:

$$\forall i, j \in [\![1, 2]\!] \tag{3.2.35}.$$

$$(\boldsymbol{X} * \boldsymbol{W})_{i,j} = (\boldsymbol{X})_{i,j}(\boldsymbol{W})_1 + (\boldsymbol{X})_{i,j+1}(\boldsymbol{W})_2 + (\boldsymbol{X})_{i+1,j}(W)_3 + (\boldsymbol{X})_{i+1,j+1}(\boldsymbol{W})_4$$

By comparing the two previous equations, the neuron's pre-activation map can easily be identified as being the convolution of the input grid by a grid composed of the neuron's weight parameters, and as a consequence can be computed as such.

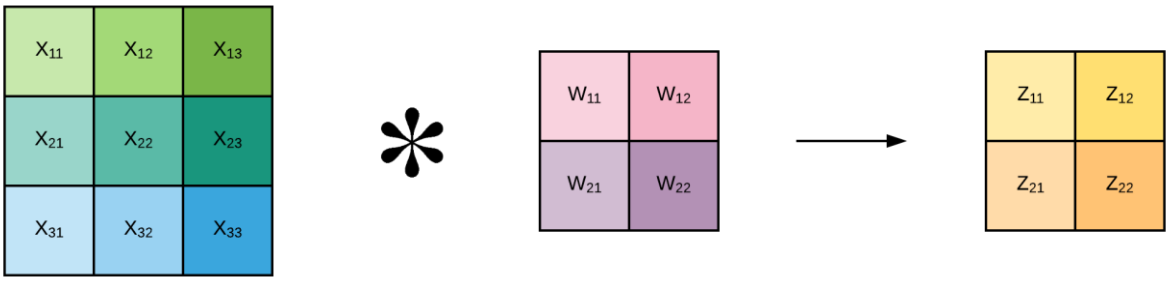


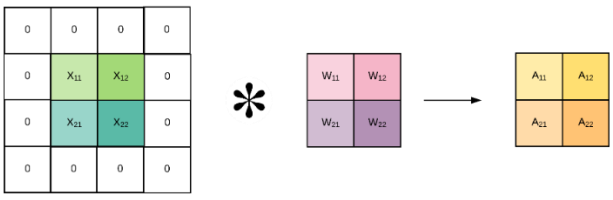
Fig 3.2.10 Same neuron as in figure 3.2.9, written as a convolution product

When applied in artificial neural networks, the convolution operation is usually extended in two distinct ways in order to provide the practitioner with more flexibility:

- Stride variations allow for the reduction of the output's dimensionality, by allowing more space between each copied neuron. As a consequence, some of the input matrixes sub grids are not observed by neurons. Formally, for $\boldsymbol{A} \in \mathbb{R}^{n\times n}, \boldsymbol{B} \in \mathbb{R}^{m\times m}$ with $(n,m) \in \mathbb{N}^2$ such that $n > m$, the convolution of A and B with stride s noted $\boldsymbol{A} * \boldsymbol{B}(s)$ is a matrix of size $\left\lfloor \frac{n-m+1}{s} \right\rfloor \times \left\lfloor \frac{n-m+1}{s} \right\rfloor$ defined as [23]:

$$\forall i,j \in \left[\!\left[1, \left\lfloor \frac{n-m+1}{s} \right\rfloor \right]\!\right], \quad (\boldsymbol{A} * \boldsymbol{B}(s))_{i,j} = \sum_{k=1}^{m}\sum_{l=1}^{m} (\boldsymbol{A})_{i*s+k-1,j*s+l-1}(\boldsymbol{B})_{k,l} \tag{3.2.36}.$$

- 0 padding can be added to the input matrix, in order to artificially raise its dimension, thus raising the convolution's output dimensionality. In practice, padding is typically used to set the convolution's output to be of same size as the input matrix.



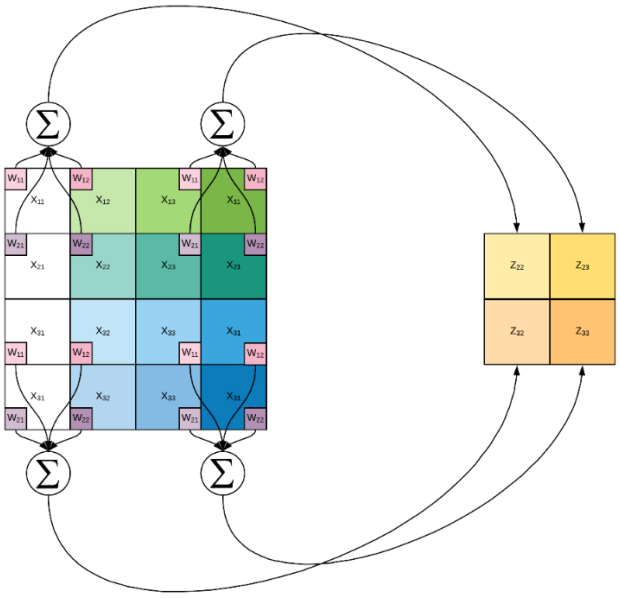


Fig. 3.2.11 Top: Convolution with padding. The result of the padded convolution of A and B has the same dimension as A. Bottom: Convolution with stride of 2. As a consequence of the increased translation step, each element of the input grid is accessed once

Finally, the concept of convolution can easily be expanded to 1-dimensional, 3-dimensional or n dimensional grid (with n any positive integer) structures, by adapting the parameter vector to a similar structure. As an example, the convolution of two 3-dimensional grid structures $\boldsymbol{A} \in \mathbb{R}^{n\times n}, \boldsymbol{B} \in \mathbb{R}^{m\times m}$ with $(n, m) \in \mathbb{N}^2$ such that $n > m$, can be defined as:

$$\forall i,j,k \in [\![1, n-m+1]\!],\ \ (\boldsymbol{A} * \boldsymbol{B})_{i,j,k} = \sum_{l=1}^{m}\sum_{q=1}^{m}\sum_{s=1}^{m}(\boldsymbol{A})_{i+l-1,j+q-1,k+s-1}(\boldsymbol{B})_{k,l,s} \tag{3.2.37}.$$

Its behavior can easily be linked to the aforedefined concept of neuron by the same identification argument used for their 2-dimensional counterparts.

3.2.7.2 Convolution layer

A convolution layer is essentially similar to the concept of neural layer in the multilayer perceptron, with the neurons previously defined as input variables linear combinations replaced with convolutional neurons. Opposite to traditionally defined neurons, their convolutional counterparts present some hyper-parameters (namely their number of connections, stride and padding). Although some advanced neural architecture allows for the joint use of neurons with different hyper-parameters in the same neural layer, a traditional convolution layer typically uses a set of neurons tuned with the same hyper-parameters, for both simplicity of implementation and limitation of the number of hyper-parameters. As a consequence, a convolutional neural layer can be implemented through the following steps:

- Define a set of neurons (as a set of small real valued matrices)
- Compute the convolution of the input variable and each neuron, resulting in as many pre-activation maps as there is neurons in the layer
- Inject each pre-activation map in an element-wise nonlinear function (typically a ReLU nonlinearity)
- Concatenate every activation map into a 3-dimensional grid.

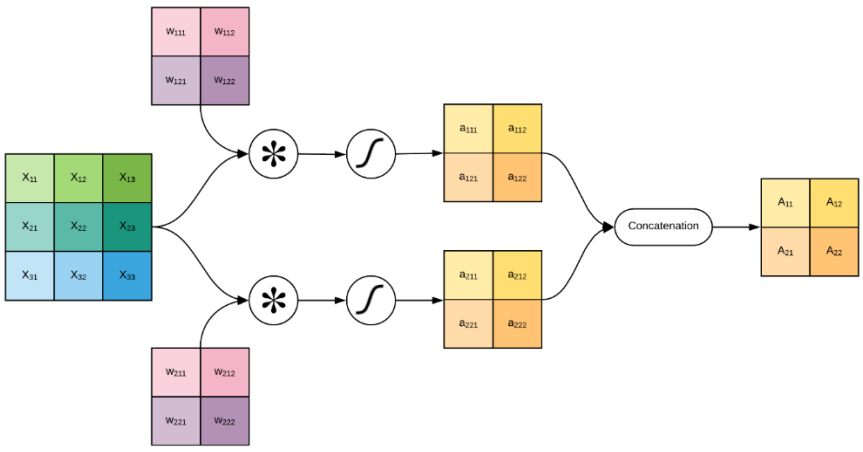


Fig 3.2.12 Example of a convolution layer with two neurons applied to a 3 by 3 matrix. The layers final output is a 2 by 2 by 2 grid (first and second axes represent the location of the pattern, the final axis resulting from the concatenation of each neuron's convolution output)

In a similar fashion to the multilayer perceptron, convolution layers are typically applied back to back to allow for the modelling of powerful nonlinear relationships characterizing the investigated dataset. However, they present a peculiarity that does not make the implementation of a sequence of layer as straightforward. Indeed, the output of a traditional neural layer is sensibly similar to its input, in the sense that both of them are simple real valued vectors. Convolutional layers, however, due to the final concatenation step, typically output data structures of dimensionality bigger than their input (N+1-dimensional grids for N-dimensional grid inputs). To prevent for this undesirable phenomenon potentially leading to an unreasonable number of unnecessary parameters, convolutional layers receiving the output of previous ones will typically apply to their input a 3-dimensional neuron of depth (last dimension) equal to their input's ones, resulting in a stabilization of the grids' dimensionality across layers.

3.2.7.3 Maximum pooling

Successive convolutional layers can be used to progressively decrease the local information contained in grid-like data structures by including it in the neurons' representation of the investigated dataset. Essentially, each layer combines local information from the previous activation maps, and so on until the input grid variable. The use of convolutional layers alone in convolutional neural network can end up being considerably expensive, specifically with same padding types. Indeed, in these conditions, the size of each activation map does not decrease across layers and make for a considerable amount of computations to perform at each step. Similarly, with a valid type of padding the size of activation decreases relatively slowly across layers (it follows an arithmetic progression). Max-pooling layers were designed in order to efficiently increase the decreasing rate of activation map sizes across layers, in order to speed up computations, typically by outputting the maximum value of its input matrix's each adjacent sub grid [24]. Similar to convolutional layers, several sizes of filter and stride can be used,

but most max-pool implementations pool maximum values from 2 by 2 sub grids with a stride of 2 in both directions, essentially taking the maximum value from every adjacent and disjoint 2 by 2 sub grid from its input matrix, as can be seen in figure 3.2.13.

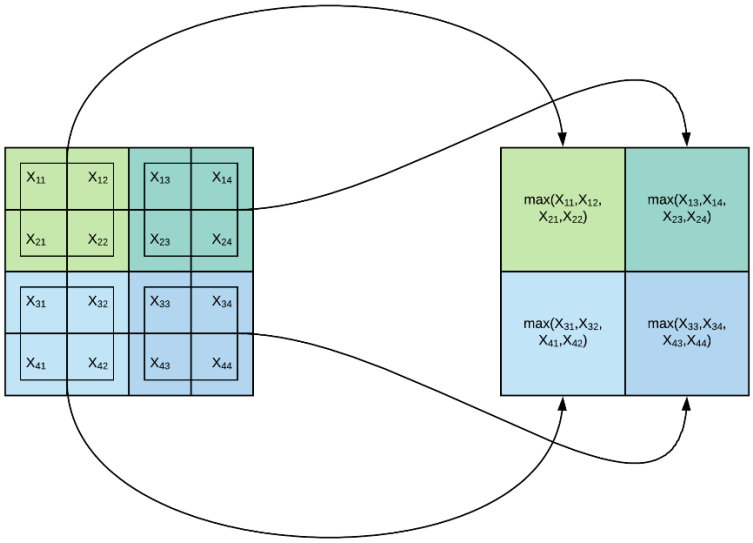


Fig 3.2.13 Example of max-pool layer of filter size 2x2 and stride 2 applied to a 4 by 4 matrix. The resulting 2x2 grid contains the maximum values from every disjoint 2 by 2 sub grid from the input matrix.

Although the theoretical justification of maximum pooling layers is still not fully understood, a heuristic explanation of their behavior is quite well received in the academic community. Typically, a convolutional neuron is expected to output high values when it detected the pattern it has been trained to recognize. As a consequence, the max pooling transformation allows for a significant decrease of representation sizes (following a geometric progression) while keeping for each neuron's activation map the most important information, as well as its location in the input grid.

3.2.7.4 Convolutional neural network

The concepts of convolutional and maximum pooling layer are typically used together to build what are caller convolutional neural networks, which are a family of parametric models that currently hold state of the art performances on a variety of tasks ranging from machine vision to voice detection.

Convolutional neural networks typically incorporate the concepts of convolution and maximum pooling layers into a multilayer perceptron, allowing for the implementation of powerful models of grid like data structures by in the following way:

- Inject the input data into a succession of convolution and maximum pooling layers
- Once the model's representation sizes have sufficiently decreased, inject the last layer's activation map into a standard, multilayer perceptron linking it to the output variable
- Fit the model through cost function optimization with gradient descent and backpropagation, similar to a traditional multilayer perceptron

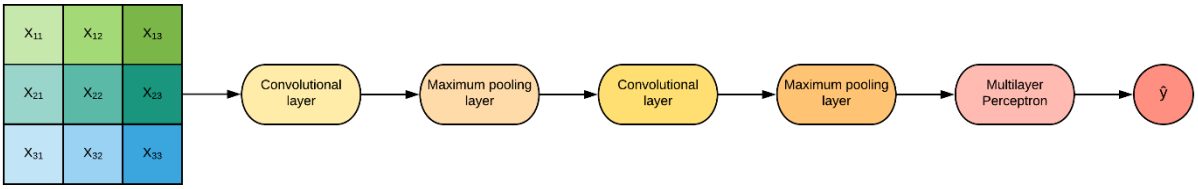


Fig 3.2.14 Example of a typical convolutional neural architecture. The grid shaped input variable is injected into a succession of convolutional and maximum pooling layers whose final output is injected directly into a multilayer perceptron

The main rationale for using a multilayer perceptron as the last part of a convolutional neural networks is related to the way convolutional neural layers build dense, hierarchical representations of the investigated dataset's input variables. Essentially, each network's layer converts the local information it is injected with into independent, increasingly complex concepts describing growing sub grids of the input variable. As a consequence, successive layer's activation maps slowly reduce in size until reaching a threshold where internal representation become non-local and dense enough for the use of standard, multilayer perceptrons.

This phenomenon was actually illustrated for better understanding in [25], which essentially introduced a method for visualizing the type of input grids maximizing a given neuron in a deep convolutional neural network trained for image classification. Figure 3.2.15 shows examples of such visualization for neurons selected from different layers, which exhibit the concept of successive, hierarchical representations inherent to deep neural networks:

- The first layer's neurons (top) typically respond to simple, unspecific patterns (colored blobs, linear patterns)
- The intermediate layers (middle) use the simple representations established in previous layer to respond to more complicated concepts
- The final layers (bottom), through successive refinements of representations, manage to encapsulate highly advanced concepts (such as a dog's face) into a single neuron

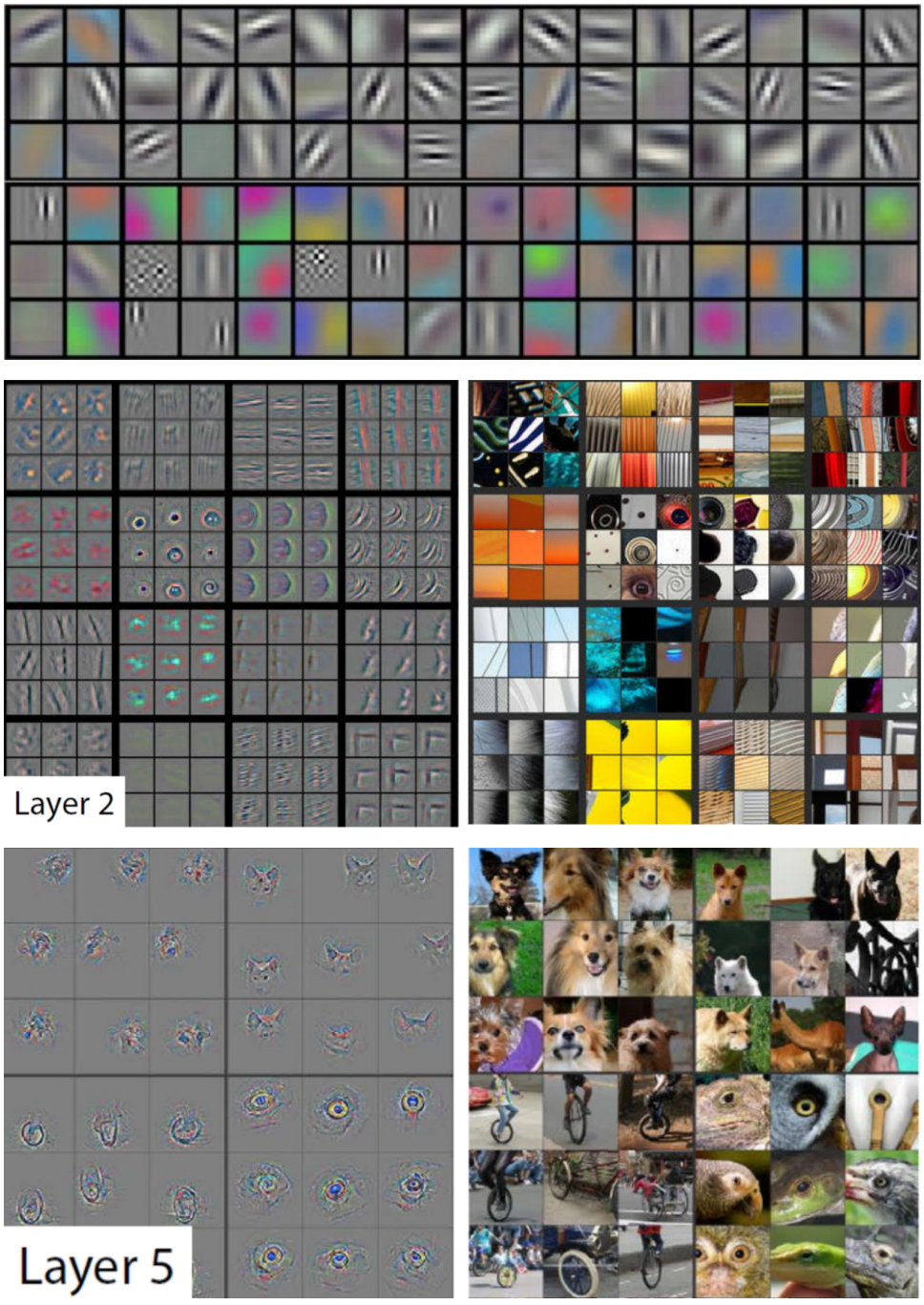


Fig. 3.2.15 Visualization of maximum activation inputs for several layer depths [25]

3.2.7.5 Dilated causal convolutions for sequence analysis

Although convolutional neural networks were typically created for computer vision and the analysis of grid-like structures, they can also be used for sequence analysis. After all, a sequence is nothing but a grid with one dimension reduced to 1. However, their use as is might not be perfectly adequate to the analysis of long sequence. Indeed, convolutional neural network in computer vision rely on the strong locality hypothesis inherent to pictures, which is not necessarily as verified in sequence analysis (specifically in natural language processing, for instance). As a consequence, specific convolutional architectures were designed specifically for sequence analysis, such as temporal convolutional networks, which were shown to outperform all recurrent neural network based approaches in a number of sequence modelling benchmark datasets [26]. Temporal convolutional networks, are essentially convolutional neural networks with two major modifications. First, in order to provide the user with sequential outputs of same length as the model's inputs, down sampling methods traditionally used in convolutional neural networks such as striding are discarded, and zero padding is used. Finally, the traditional convolution operation at the basis of the convolution is modified into what is called a dilated causal convolution, which differs twofold from its traditional definition:

- In order to respect causality, filters used to define the output at time-step t should not have access to inputs for times $t' > t$
- A dilation factor d is introduced to the convolution operation. This dilation factor allows the convolution to access non-adjacent part of the input grid structure, thus expanding the length of the corresponding neuron's receptive field without using more parameters. Formally, for $A \in \mathbb{R}^n, B \in \mathbb{R}^m$ with $(n, m) \in \mathbb{N}^2$ such that $n > m$, the dilated convolution of A and B noted $A * B(d)$ is a vector defined as:

$$\forall i, j \in [\![1, n-m+1]\!], \quad (A * B(d))_i = \sum_{k=0}^{m-1} (A)_{i+k.d} (B)_k \tag{3.2.38}.$$

Fig. 3.2.16 shows an example of a temporal convolutional network. By applying successive convolutions with exponentially increasing dilation factors, these convolutional networks are able to model long sequences of data while requiring a surprisingly low number of parameters (the number of parameters evolves logarithmically with the maximal sequence length the network is able to fully model). In addition, temporal convolution still benefits from the powerful parallel implementation of convolution layers which makes them significantly faster than traditional recurrent neural networks.

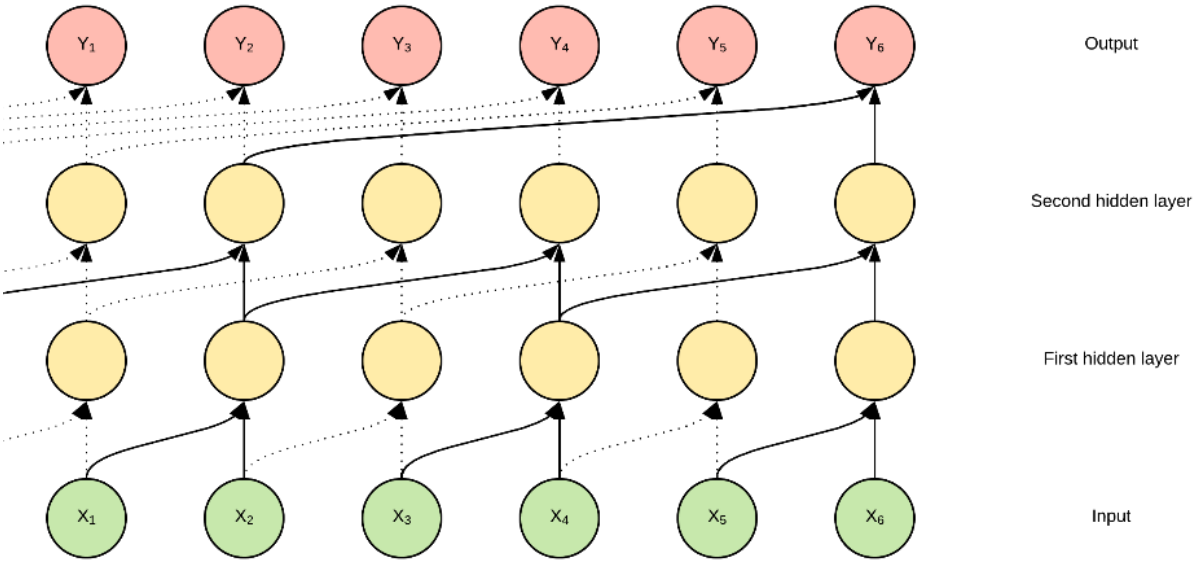


Fig. 3.2.16 temporal convolution network with 2 hidden layers. The dilation factors of the neural layers are in growing depth order 1, 2 and 4. With sufficient depth, the network's output at a given time-step has access to the input's every time-step before it.

3.3 Practical aspects of neural networks

Although the theoretical basis of artificial neural network-based learning algorithms is fairly simple, their practical implementation often results in catastrophic failures (such as, for instance, gradient explosion) without careful considerations [27]. As a consequence, a set of practical rules were derived from experiments as well as practitioner insight to improve their overall performances.

3.3.1 Model evaluation

The evaluation of a supervised model's performance is paramount to the practical implementation of a deep neural network. Indeed, machine learning based models are typically used for their predictive capabilities and are consequently expected to not only correctly model the dataset, but also any additional example. However, as the model can only learn from the gathered dataset during the training process, its ability to generalize to never before seen cases cannot be guaranteed from its performance on the sole training data. In practice, because of their ability to model close to any relationship between input and output variables, deep neural networks have a tendency to derive models that are too specific to the investigated dataset. As a consequence, they require a careful analysis of their performance.

3.3.1.1 Training, development and testing sets

The learning process of an artificial network typically consists of minimizing its prediction error on the investigated dataset. As a consequence, the estimation of the model's performance on the data used for training is inherently biased. Indeed, on this specific set of examples, the model usually shows good predictive capabilities, because it was actually chosen in a highly dimensional space models as the one with best predictive capabilities. It is consequently standard practice to divide the investigated dataset into two distinct, independent subgroups. One group will be used for optimizing the model's parameters, and the second one will be used to assess the model's performance [28]. As the algorithm does not have access to the second group during training, its derived model is not optimized to properly model them and allow for the extraction of an unbiased estimation of the model's predictive

power. The group of subjects used for the parameter's optimization is usually called the *training set*. Its counterpart used for model evaluation is called the *testing set*.

Although the training and testing sets are sufficient to obtain an unbiased estimate of a given model's performance, the practical implementation of a machine learning based model is usually done by fitting an important number of different models to the investigated dataset in the search for the best one. For instance, a practitioner could try a lot of different multilayer perceptrons on a given dataset, by varying the number of layers in the networks, or the number of neurons per layers. This search for the best available model can also be seen as an optimization process that biases the model's evaluation. Indeed:

- Trying to compare the performance of two models on the training set is not reliable, as a bigger network will typically (but not necessarily) yield better training accuracies than a smaller one, without any indication on the actual gain in performance
- Trying to compare the performance of a set of models on the test set adds bias to the estimation. Indeed, once again, the final model will be chosen as the one with best accuracy on the test set, and as a consequence the argument used to illustrate the biased aspect of the training accuracy can yet be used on the test set; the model's performance on the test set is "good" because it was defined as the best among every proposed solution

To counter this additional undesirable bias, an additional group is made from the dataset, called the *validation set* or *development set,* and is used to choose the best model, before using the test set to obtain an unbiased estimation of the latter's performances.

To conclude, the standard methodology to derive a good model from a machine learning algorithm is the following [29]:

- Divide the investigated dataset into three distinct parts, the training set, the development set and the test set

- Use the training set to optimize the learning algorithms used during the investigation
- Use the development set to choose the best model among the different ones considered during the investigation
- Use the test set to obtain an unbiased estimation of the final model's performance

An important question during the separation of the investigated dataset into training, development and test set is the percentage of data samples to allocate to each of the sets. Indeed, the model's parameters optimization usually requires a substantial amount of examples to behave properly. However, the computation of the models' performances, whether for model selection of final estimation, needs also be reliable, especially to be able to evaluate performances on rare subcategories.

A standard practice for datasets presenting a sample size less than a ten thousand is to allocate 60% of the samples to the training set, and an equal 20% for both the development and test sets.

3.3.1.2 Bias variance dilemma

During the practical implementation of a machine learning algorithm, two typical unwanted phenomena can arise [30]:

- The model's training error is low, but its test error is significantly higher,
- Both the model's training and test error are significantly higher than expected by the practitioner.

These two phenomena are respectively called over and underfitting. Underfitting is typically observed when trying to model complex relationships with simple models. The model then does not have enough expressive power to identify the investigated dataset's patterns properly, and consequently shows poor predictive ability on both the training and testing set. Overfitting, on the other hand, is a result of using a family of algorithms with so much explanatory power, that they yield unstable models, meaning that two separate datasets gathered from the same phenomenon will be explained by

significantly different models. Indeed, the algorithm is powerful enough to account for the random noise typical of statistical dataset, usually to the detriment of the test error.

Figure 3.3.1 shows a simple example of these concepts in the context of a univariate polynomial regression problem. The left-hand plot shows an attempt to fit the dataset with a linear regression model (polynomial regression of degree 1) and is consequently unable to account for the data's non-linear nature. In contrast, the right-hand figure consists of a polynomial regression model of degree 15. The model has enough explanatory power to obtain a fit close to perfect to the noisy training dataset but yielding an unstable approximation function that equally fails to understand the true underlying function. The centered image shows the polynomial regression best suited for the investigated dataset and could be obtained for instance from a model selection scheme using training, development and test sets.

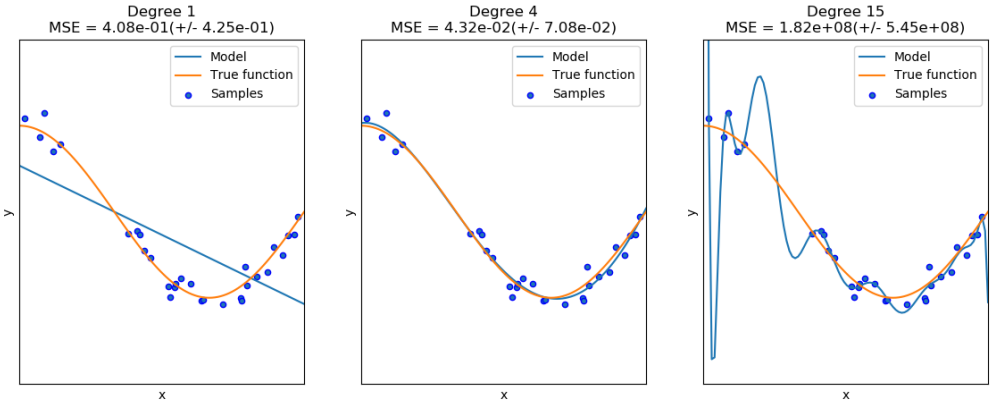


Fig. 3.3.1 Examples of under and overfitting of a non-linear regression problem. Right: Overfitting model Left: Underfitting model, Center: Fine-tuned model

This phenomenon relying on the difference of explanatory power between different types of models is called the bias variance dilemma. Typically:

- An algorithm with high bias will typically make too much assumption on the relationship it is trying to model, as in the linear regression example, where a straight line is fitted to a curve

- An algorithm with high variance, however, will be able to model a large number of relationships, but with that gained explanatory power comes the potential to be data-specific, and as a consequence to overfit the dataset

The model selection scheme discussed previously, and for which the development set is paramount, is typically used to fine tune this tradeoff to the investigated dataset.

In practice, deep neural network only underfit the dataset in exceptional circumstances but have a strong tendency toward overfitting that usually requires the use of additional techniques to reduce their variance, such as regularization.

3.3.2 Regularization

Deep artificial neural networks are universal approximators, meaning that they can essentially approximate any continuous function. As a consequence, the neural network's family of models' are sensibly prone to overfitting, even on very large datasets. Regularization schemes are a variety of techniques used in practical applications to limit this variance, and as such overfitting, typically by preventing the network from learning too complex of a function. Regularization techniques for neural networks are typically divided into two distinct approaches:

- The addition of a penalty term in the cost function [31]
- The drop out method, which is specific to artificial neural networks [32]

3.3.2.1 Penalization methods

The concept of regularization through the addition of a penalty term in the cost function is widely used in both machine learning and data driven statistical modelling. A typical example of such an approach can typically be observed when performing Ridge or Lasso regression [33,34]. The rationale behind penalization regularization scheme is to limit the explanatory power of the learning algorithm, by preventing it of outputting artificially too complex of a function. For instance, one can notice from empirical observations that overfitting functions are typically not "smooth". They instead tend to

oscillate rapidly between observations to fit the dataset as well as possible. As a consequence, the practitioner might want to constrain the learning algorithm into prioritizing “smooth” function against “non-smooth” ones, even if the latter makes a better job at approximating the training data.

Although quite ambiguous, the concept of a model’s “smoothness” can be formally expressed in a relatively straightforward manner. Indeed, for a given parametric model, “non-smooth” functions will typically present with higher parameter values than “smooth” function. As a consequence, constraining the model’s parameter values to remain low can limit how “non-smooth” the model can eventually become during the learning process. This can typically be achieved by the addition of a constraint on the model’s parameters in the cost function. Considering a ridge linear regression model on a dataset $\{(X_i, y_i)\}_{0<i<N+1}$ of N observations, the new, regularized cost function can be defined as:

$$L(W,b) = \frac{1}{2N} \cdot \sum_{i=1}^{N}((W)^T \cdot X_i + b - y_i)^2) + \lambda \left(\|W\|_2{}^2 + b^2\right) \quad with\ \lambda \in \mathbb{R}^+ \tag{3.3.1}$$

The cost function’s first term is identical to that of the standard linear regression one. The second term constitutes the regularization constraint. Where a standard linear regression’s model parameters are optimized in order to best fit the data, a ridge regression’s objective is to jointly minimize the model’s fit to the data as well as its squared parameter weights. As a consequence, the model will prioritize parametric solutions that might not best fit the data, but that keep its parameters to low values, typically leading to “smooth” solutions.

The λ value is to be considered a model hyper-parameter and denotes the regularization scheme’s importance during the model fitting process. It allows for the practitioner to choose how much importance is to be allowed to the “smoothness” constraint. Typically:

- High values of λ will guide the model toward emphasizing its smoothness requirement. In such conditions, the model's fitness to the data might be overlooked, which can lead to underfitting
- Low values of λ will result in a cost function close to its unregularized counterpart, and result in overfitting

As a consequence, the fine-tuning of the λ value is paramount to a successful regularization, and its best value is typically determined from testing on the development set.

3.3.2.2 Drop out method

Although penalty-based regularization methods constitute a reasonably simple approach to preventing overfitting which can be applied in a wide variety of situations, its rationale fails to encapsulate the inherent reason for overfitting in neural networks. By using its highly connectionist structure, a big enough artificial neural networks can explain any observed dataset, simply by organizing its neurons and their high number of connections to each other into highly complex data representations. As the network grows bigger, the number of connections between neurons substantially increases, which often lead to overfitting by the conception of features specific to the dataset, thus failing to generalize to new examples. The drop out method is a regularization method that was created in order to prevent neurons to build their features by relying on highly specific combinations of connections, by stochastically shutting down some of the network's neurons during training [32]. Essentially, for each training step during the network optimization process, the modelled output variable inference process will be performed by a random subnetwork of the actual implemented neural network. Each neuron builds its representation to become robust to the random absence of input features coming from the previous layer, which prevents them from building artificially too complex and observation specific representations.

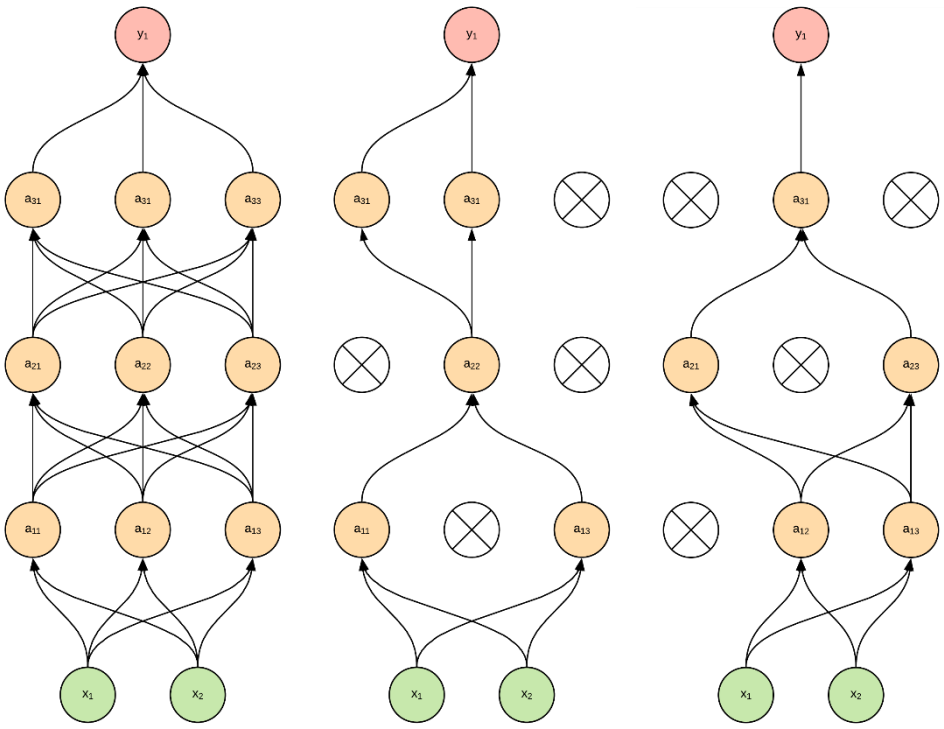


Fig. 3.3.2 Three layers perceptron with drop out regularization, in its complete version and during two distinct training operations. Neurons are shut down randomly in each layer for each iteration, and the output variable is inferred from the obtained subnetwork

Several implementations of the drop out method have been devised over the years, the traditional approach being:

1. Define a neural network for the investigated modelling problem
2. During every training iteration, randomly choose several neurons to shut down (each neuron has a shutdown probability of p)
3. Mute the selected neurons' outputs to 0
4. Use the partial network to infer the output variable
5. Backpropagate the gradient through the subnetwork
6. Perform the gradient optimization step

7. After descent convergence, use the entire network for model evaluation, with every neuron's output multiplied by its drop out probability distribution's expectation

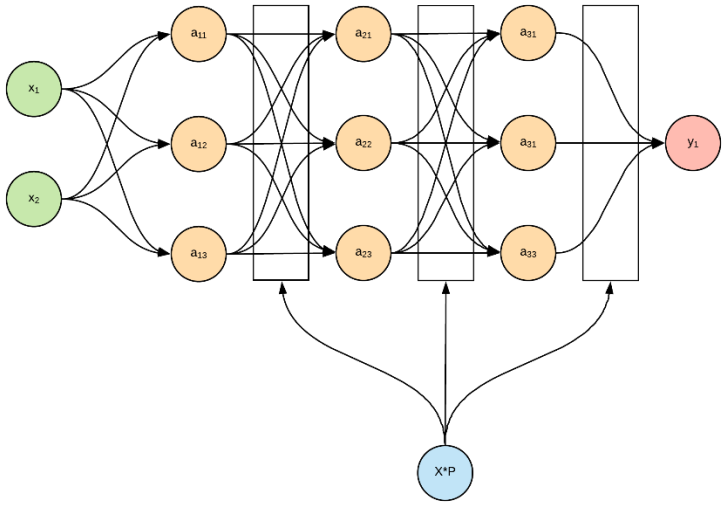


Fig. 3.3.3 Same neural network as in Figure 3.3.2 at test time of for true model inference. The neurons' outputs are all used (no drop out) but are multiplied by the drop out probability distribution's expectation

3.3.3 Practical aspects of neural network optimization

The process of optimizing an artificial neural network's parameters is typically difficult [35]. Not only does it require the use of gradient based methods on non-convex cost functions which, as mentioned previously, can behave sub-optimally, but neural network's natural properties tend to make said cost function quite unfit to be explored with gradient based methods. Typically, a naïve implementation of an artificial neural network optimized by an equally simple gradient descent algorithm will fail to find a correct parametric model for the investigated dataset. Instead, two different undesirable phenomena will usually arise during optimization:

- The cost function's gradient will become significantly larger than its previous values, causing the gradient descent algorithm to diverge

- The cost function's gradient will become so low that the gradient procedure will stop making any progress, without having reached a local minimum

These two complications are known together as the vanishing and exploding gradient problem [27] and are the reason artificial neural network were discarded by the majority of the academic community as interesting machine learning models until the 2010s. Even though this problem is inherent to deep neural networks, it can in practical cases be overcome enough to obtain really powerful predictive models, typically by using careful parameter initialization schemes and subtler versions of the gradient descent algorithm.

3.3.3.1 Parameter initialization

The first step required during the optimization of a parametric model with a gradient descent procedure is to initialize the model's parameter, typically randomly. The algorithm typically uses this initialization as a starting point and begins its search for a minimum from it.

When it comes to convex optimizations problems, this starting point does not have any effect on the obtained solution. Indeed, assuming that the algorithm's step rate is correctly tuned, the algorithm is guaranteed to converge toward the function's global minimum. However, this desirable property does not remain true when gradient methods are applied to non-convex functions such as deep neural network's cost functions. Indeed, as can be seen in figure 3.3.4, two gradient descent procedures initialized slightly differently can converge toward significantly different solutions. As a consequence, properly initializing a gradient descent in non-convex optimization problems, and as such, in the process of training a deep neural network, has a tremendous impact on its results. In fact, the new artificial neural network parameter initialization procedures introduced in the late 2000s and early 2010s are considered a major factor in the first successful practical implementations of deep learning algorithms.

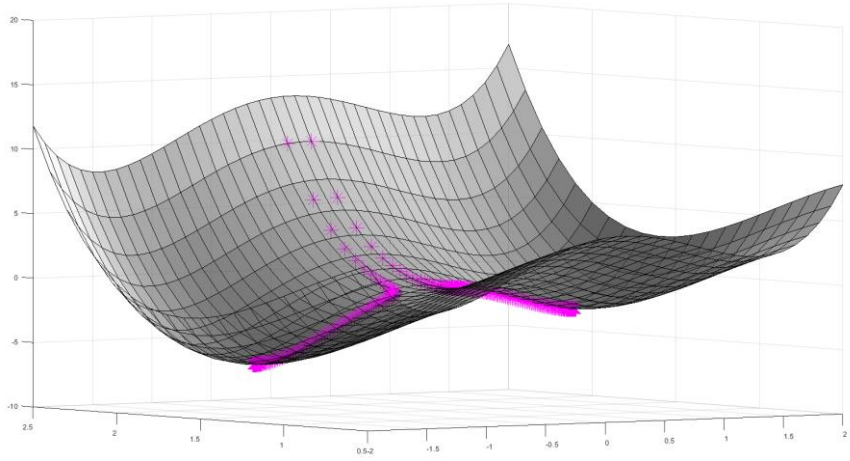


Fig 3.3.4 Two gradient descent procedures on a non-convex function, with different initialization (starting point). Even though the two procedures are initialized closely, the resulting local optima are sensibly different

Historically, the first parameter initialization scheme resulting in the successful training of deep neural networks was obtained through the use of unsupervised layer-wise pre-training [36]. However, recent work has led to the creation of initialization schemes that outperform such pre-training both in performance and computational costs, such as, for instance, the Xavier initialization procedure [35].

The Xavier initialization scheme is a probabilistic parameter initialization method that relies on heuristic considerations on information flows within a deep neural network using hyperbolic tangent nonlinearities. For an L-layered neural network, the i^{th} layer's parameters are sampled from a random normal distribution with 0 mean and variance σ^2 such as:

$$\sigma^2 = \sqrt{\frac{1}{n^{i-1}}} \tag{3.3.2}$$

$$with\ n^{i-1}\ the\ number\ of\ neurons\ in\ the\ network's\ i-1^{th}\ layer$$

In a similar fashion, the i^{th} layer of a deep neural network using ReLU nonlinearities is to be initialized by sampling parameters from a random normal distribution with 0 mean and variance σ^2 such as:

$$\sigma^2 = \sqrt{\frac{2}{n^{i-1}}} \quad (3.3.3).$$

$$with\ n^{i-1}\ the\ number\ of\ neurons\ in\ the\ network's\ i-1^{th}\ layer$$

3.3.3.2 Advanced gradient-based optimization method: Adagrad

While a proper parameter initialization can significantly improve the quality of a neural network parametric model solution extracted from a given dataset, using the classical implementation of a gradient descent procedure during the learning process can result in substantial practical difficulties:

- As the dataset and the neural network get bigger, the gradient gets increasingly more expensive to compute, both in time and memory, and rapidly becomes practically intractable on machines with limited computing power
- Even with proper parameter initialization, the cost function's surface remains difficult to explore. The gradient descent can end up "stuck" on a function's neighborhood with a gradient close to 0 that is not a local optimum (degenerate critical point or saddle points), as is shown in figure 3.3.5

Two major concepts have been introduced to adapt gradient descent methods to these difficulties:

- Stochastic (or mini-batch) gradient descent
- Gradient descent with adaptive learning rates (Adam optimization)

For the practical optimization of a deep neural network's cost function, these two improvements on the traditional gradient based procedure are usually combined into powerful optimization algorithms.

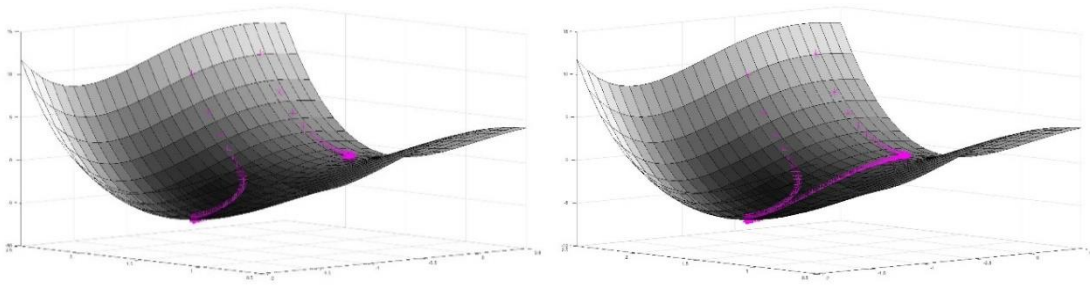


Fig 3.3.5 Two standard gradient descent procedures with different initialization. Left: one procedure has converged toward a minimum, the other is stuck on a saddle point. Right: The second descent algorithm "got out" of its saddle point, and converged to the same local optima as the first version, but took a significantly larger amount of time

Mini-batch gradient descent

Theoretically, the gradient of any given neural network's associated cost function can be obtained through the use of the backpropagation algorithm. By making use of the cost functions' inherent additive nature, its gradient computation is divided into sub gradient computations at the sampled individual level. Although an efficient method to automatize the computation of the gradient process, this method presents a significant drawback when applied to dataset with large sample sizes. Indeed, it requires the capacity to store and compute as many gradients as there are examples in the dataset, usually resulting in memory overflow and unreasonably long computation times.

Mini-batch gradient descent offers a solution to this physical problem by approximating the cost function's gradient through the use of probabilistic sampling [37]. For a given iteration of the descent procedure, the model's cost function is estimated from a small number of subject (a mini-batch) randomly sampled from the dataset. As a consequence, this approximation of the cost function's gradient can be computed through backpropagation on a limited subsample of examples, significantly

diminishing its computational cost. By successive random sampling at each step, the gradient descent algorithm can then be approximated with limited computational costs.

A typical implementation of a mini batch gradient descent with mini-batch size m during the fitting of a linear regression model on a dataset $\{(X_i, y_i)\}_{0<i<N+1}$ of N observations, can be described as follows:

1. Define a random permutation σ of $[\![1, N]\!]$ that will be used to shuffle the dataset
2. For each gradient descent iteration step s, compute an approximation of the cost function

$$L_s(W, b) = \frac{1}{2N} \cdot \sum_{i=m*s}^{m*(s+1)} \left((W)^t \cdot X_{\sigma(i)} + b - y_{\sigma(i)}\right)^2) \quad (3.3.4).$$

3. Compute the approximated cost function's gradient (through the backpropagation algorithm in the case of artificial neural network based models), and use it to update the parameters (standard descent step)
4. Once every subject from the investigated dataset has been used for cost function approximation (in other words when $m * s > N$), reiterate the process with a new random permutation
5. Continue steps 1 through 4 until convergence

The number of individuals to sample for cost function estimation is considered a model hyper-parameter and has a significant impact on both the method's computational efficiency and the outputted minimum's quality. Indeed, low mini-batch size will require lower computational requirements when computing the objective's gradient, but in the other hand typically make for a more unstable optimization algorithm, as the statistical estimation of the cost function gets poorer with few sampled examples. In practice, typical mini batch size values lie between 50 and 200 examples.

Adaptative Moment estimation optimisation (Adam)

Although mini-batch gradient descent allows for the approximation of a usually prohibitively expensive gradient for the optimization of artificial neural networks, it is not enough alone to allow for optimization of the difficult functional surfaces typically observed in deep artificial neural network learning processes. As aforementioned, the gradient descent's ability to converge towards a local minimum is strongly conditioned on the proper tuning of its hyper-parameter, the learning rate. Indeed, as it quantifies the size of the step to be taken in the direction defined by the gradient, strong learning rate values yield an unstable optimization algorithm that fails to stop when arriving close to the found local optima, and weak values result in an unreasonably slow optimization algorithm. Unfortunately, the fine tuning of this hyper-parameter is sometimes not efficient on its own to obtain a good optimization algorithm. Indeed, using the same step size value to update every parameter implicitly makes the assumption that the investigated functional surface has roughly the same properties in every direction, which might not be the case, especially when it comes to neural network cost functions. The Adam optimization method [38] was introduced specifically to address this issue and provides a gradient based descent algorithm which adapts its learning rate to each parameter individually. Its definition is substantially similar to the traditional gradient descent's one, with for only difference a modified update rule and consists, for a function f from $\mathbb{R}^n$ to $\mathbb{R}$ and four real number η, ε, β_1 and β_2, of the following:

1. Initialize $a_0 \in \mathbb{R}^n$ randomly and $[m_0, v_0] \in \mathbb{R}^n \times \mathbb{R}^n$ to $[0, 0]$
2. Iterate through the parameter update rule at step $t \in \mathbb{N}$:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla f(a_{t-1}) \quad (3.3.5),$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\big(\nabla f(a_{t-1}) \odot \nabla f(a_{t-1})\big) \quad (3.3.6),$$

$$\hat{m}_t = \frac{m_t}{(1-\beta_1)^t} \quad (3.3.7),$$

$$\hat{v}_t = \frac{v_t}{(1-\beta_2)^t} \quad (3.3.8),$$

$$\forall i \in [\![1, n]\!], \qquad (a_t)_i = (a_{t-1})_i - \frac{\eta}{\sqrt{(\hat{v}_t)_i} + \varepsilon}(\hat{m}_t)_i \quad (3.3.9).$$

$$with\ (x)_i\ the\ i^{th}\ parameter\ of\ vector\ x$$

3. Stop iterating steps 1 and 2 when a_t has converged toward a local minimum

Once again η, ε, β_1 and $\beta_2$are to be considered model parameters. Although the tuning of η has a significant impact on the optimization procedure's quality, ε, β_1 and β_2 are typically chosen to be respectively 10^{-8}, 0.999 and 0.9 [38].

3.3.3.3 Normalization

As aforementioned, improving on the gradient descent algorithm used during the training of deep neural networks, through careful initialization and refined update rule, can significantly improve the quality of the obtained parametric model, as well as speed up convergence time. However, using a better optimization algorithm is not the only way possible to improve the process of searching for the minimum of a non-convex function. Indeed, despite the substantial advances introduced in gradient based optimization procedures, a deep neural network's cost function remains substantially difficult to optimize. A different approach to overcome this inherent complication is, instead of improving on the optimization algorithm, to try to modify the cost function itself into a simpler, but equivalent one

that can then be optimized in a more straightforward manner. This idea has led to two main methods that significantly improve the quality of deep neural network models, input and batch normalization.

Input normalization

The cost function of a deep neural network being built upon samples from the investigated dataset, its shape and nature highly depend on the latter. As a consequence, applying simple transformation to the dataset can potentially have a positive effect on the model's cost function, by for instance improving its compatibility with gradient based optimization methods. Although many options are available to the practitioner to preprocess a dataset, the most widely used method of data preparation for the implementation of deep neural network is input normalization.

As aforementioned, one major difficulty with optimizing neural network models comes from the heterogeneity of their cost function across directions, which lead to the implementation of adaptive learning rates gradient methods. Input normalization works toward an additional improvement on the optimization process, this time by modifying the cost function itself to be more homogenous across directions [39].

The process in itself is fairly straightforward and consist of normalizing every input variable from the investigated dataset to be of mean 0 and variance 1 by applying to them both an offset and a linear scaling estimated from the dataset. For a dataset with m input variables $(x_i)_{0<i<m+1}$, input normalization can be performed by applying the following affine transformation to each input variable:

$$\forall i \in [\![1, m]\!], \qquad f(x_i) = \frac{x_i - \mu(x_i)}{\sigma^2(x_i)} \tag{3.3.10}.$$

$$with\ \mu(x_i)\ and\ \sigma^2(x_i)\ the\ mean\ and\ variance\ of\ x_i, estimated\ from\ the\ dataset$$

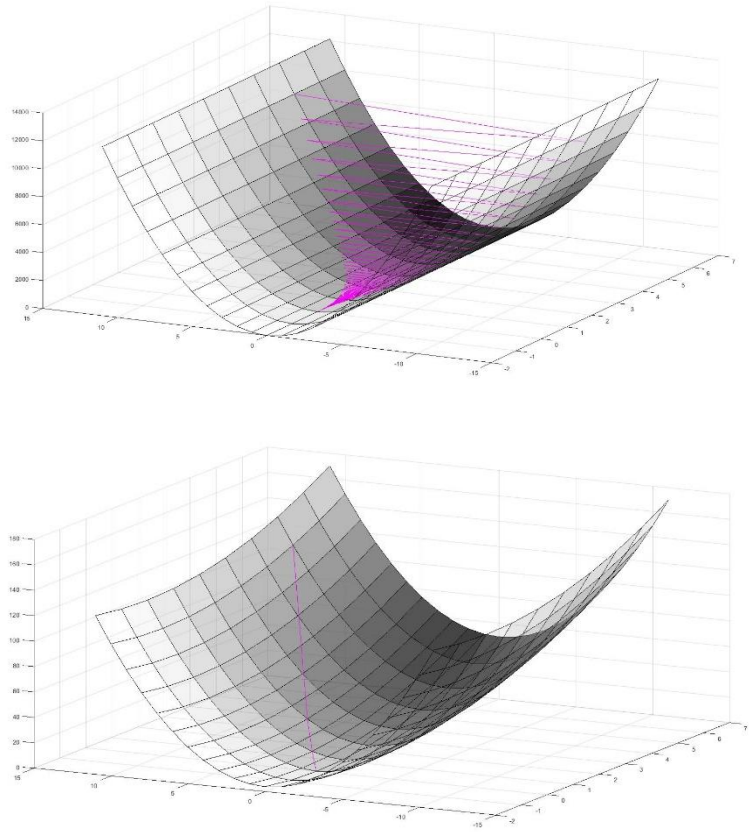


Fig. 3.3.6 Standard gradient descent on a linear regression model's cost function before and after input normalization. Top: un-normalized function with learning rate of 0.0098. Convergence is reached in 280 iterations, and higher learning rate lead to divergence. Bottom: normalized function with learning rate of 0.3. Convergence is reached in <10 steps

The positive effect of input normalization on the optimization of a model's cost function through gradient based methods can be visualized in figure 3.3.6, which illustrates an example of linear regression model's cost function and its dedicated optimization process both before and after input normalization. Before normalization, the cost function presents a strong slope heterogeneity across directions. As a consequence, the maximal allowed value for the descent procedure's learning rate is strongly limited to small values, to prevent gradient instability and divergence. The same cost function after input normalization presents a functional surface with a slope that is more homogenous across directions, which allows for much higher learning rates, and converges much faster to a local optimum.

Although this phenomenon is also typically addressed by the implementation of adaptive learning rate descent methods such as the Adam optimization algorithm, a joint use of both methods typically increases the convergence speed of the training process in artificial neural network-based models.

Batch normalization

Although Batch normalization shares some conceptual and practical similarities to input normalization, its main rational is quite different.

The power of deep neural network models comes from its ability to use its layers to build increasingly complex representations of the investigated dataset, by letting each layer build on its predecessor. During the learning process, each layer adapts itself to the previous layer's outputs in order to build the best representation at its level, thus contributing in improving the model's overall quality. However, the previous neural layer is also modifying itself, for the exact same reasons. As a consequence, for a given layer, the parameter update fulfills two purposes:

- Adapting the layer to the changes made in the previous layer's representation,
- Adapting the layer to improve the model's performance on the training set.

Batch normalization was introduced to reduce the need for each layer to adapt to changes in its predecessors and "focus" on the modelling task at hand, by fixing each layer output's mean and variance to 0 and 1, respectively, typically with an affine transformation similar to input normalization [39]. However, unlike input normalization, which is a rather straightforward procedure, batch normalization requires more thoughts in its practical implementation. Indeed, as the learning process progresses, the network's layer outputs are continually changing, and with them their first order statistics. As a consequence, the normalization step is updated at every mini-batch gradient descent iteration, and the mean and variance are estimated for every layer from the currently used batch of training examples.

Although this normalization procedure allows the model to learn on a more stable version of its internal representations, fixing the mean and variance of each layer's outputs might actually hurt the model in itself, by preventing each layer to make use of its nonlinearities correctly. As a consequence, a final affine transformation is typically applied to the normalized outputs, with its offset and scaling factors considered model parameters that are trained by the model through the gradient process.

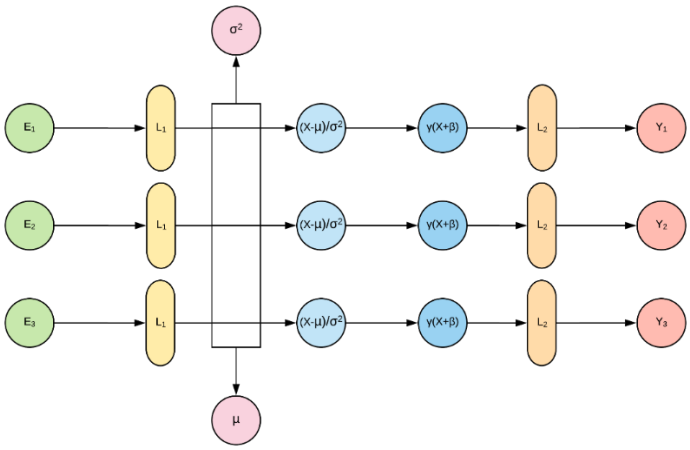


Fig 3.3.7 Example of a two layers' perceptron with batch normalization between the first and second layers, with a batch of 3 examples (E_1, E_2, E_3)

For a given model's layer l with m output neurons $(a_i)_{0<i<m+1} \in \mathbb{R}^m$ and two scalars $\gamma, \beta \in \mathbb{R}$, the batch normalization process during an iteration of the mini-batch gradient descent process can be formally defined the following way [39]:

- Estimate mean and variance for the layer's every output neuron from the current iteration's batch
- Normalize the batch outputs to null mean and unitary variance in a similar fashion as input normalization
- Rescale the outputs by applying the following parametric affine transformation:

$$\forall i \in [\![1, m]\!], \qquad f(a_i) = \gamma\,(a_i + \beta) \tag{3.3.11}.$$

Layer normalization

Although the idea of batch normalization constitutes a powerful mean of improving very deep neural networks' behavior during model fitting, it suffers from a significant issue that arises once the model is fully trained and ready to make predictions. Indeed, batch-normalization relies on having a batch of examples on which to compute all the layer outputs' mean and standard deviation. However, one might want to make predictions on single examples at a time, where these concepts are not defined. Even in the case of predicting on several examples at a time, for a given example, predictions would depend on other values present in the batch, leading to non-deterministic predictions if those were to be randomly sampled. This issue is typically overcome by storing a moving average of the mean and variance of each layer during training, and use these when predicting. This fix comes however with great computational cost. As a consequence, the use of batch normalization is usually discarded in modern neural architectures in favor of layer normalization. Layer normalization also focuses on freezing a perceptron's outputs mean and variance. However, instead of normalizing one given layer's outputs across all candidates in a batch, it focuses on normalizing each individual's layer's vectorial output. As a consequence, the mean and standard deviation can be computed for each individual separately, which overcomes the necessity of keeping moving average updates of the batch statistics proper to batch normalization. In a similar fashion to what is done in batch normalization, the neural layers' normalized outputs can afterwards be rescaled. For a given model' layer l with m output neurons $(a_i)_{0<i<m+1} \in \mathbb{R}^m$ and two scalars $\gamma, \in \mathbb{R}$, the layer normalization process during an iteration of the mini-batch gradient descent process can be formally defined the following way:

- Estimate mean and variance of $(a_i)_{0<i<m+1}$
- Normalize the layer's outputs with these derived values so that $mean((a_i)_{0<i<m+1}) = 0$ and $std((a_i)_{0<i<m+1}) = 1$
- Rescale the outputs by applying the following parametric affine transformation:

$$\forall i \in [\![1, m]\!], \qquad f(a_i) = \gamma\,(a_i + \beta) \tag{3.3.12}$$

3.3.3.4 Hyper-parameter tuning

Deep artificial neural network models present a substantial number of hyper-parameters:

- The learning rate
- The batch size
- The number of hidden layers
- The number of neurons per layer
- The regularization hyper-parameters
- Other model specific hyper-parameters (stride parameter in convolutional layers for instance)

Finding appropriate values for this set of hyper-parameter is paramount to the successful practical implementation of a deep learning based algorithm to a given modelling problem. As mentioned before, the estimation of the hyper-parameter values needs to follow a specific methodology, to avoid overfitting and biased estimation of model accuracy:

- One model with a given set of hyper-parameters is extracted from the training dataset,
- The best set of hyper-parameters is inferred by testing several combinations on the development set,
- The best model's accuracy is estimated on the test set.

Although most hyper-parameter tuning in deep artificial neural network follows this methodology, the question of how to come up with potentially good sets of hyper-parameters to try on the modelling task, has yet to be defined.

Several procedures have been devised over the year to explore the hyper-parameter space to discover proper model settings, varying from powerful, but computationally expansive solutions to simpler search procedures, with the two most widely used approaches being grid and random search.

Grid search

The grid search approach to hyper-parameter optimization is one of the most widely used method in machine learning. It consists of limiting the space exploration to a subset of values for every hyper-parameter. The resulting value space forming an approximation of the original in the shape of a grid-like structure. More formally, for a given model with n hyper-parameters $(\lambda_i)_{0<i<n+1} \in \mathbb{R}^n$ a grid search can be implemented as following:

1. For each hyper-parameter λ_i, define a number of candidates to sample m_i and a bounded interval to sample them from
2. For each hyper-parameter λ_i , divide the selected interval into $m_i - 1$ regular intervals $\left([(\lambda_i)_1,\ (\lambda_i)_2], -, \left[(\lambda_i)_{m_i-1},\ (\lambda_i)_{m_i}\right]\right)$ and select $S_{i,m_i} = ((\lambda_i)_j)_{0<j<m_i+1}$ as the subset of λ_i values to explore
3. Train a model with the training set for every hyper-parameter combination $(\lambda_{c1}, -, \lambda_{cn}) \in S_{1,m_1} \times - \times S_{n,m_n}$
4. Select the best set of hyper-parameters as maximizing the model's accuracy on the development set
5. Test the model's accuracy on the test set

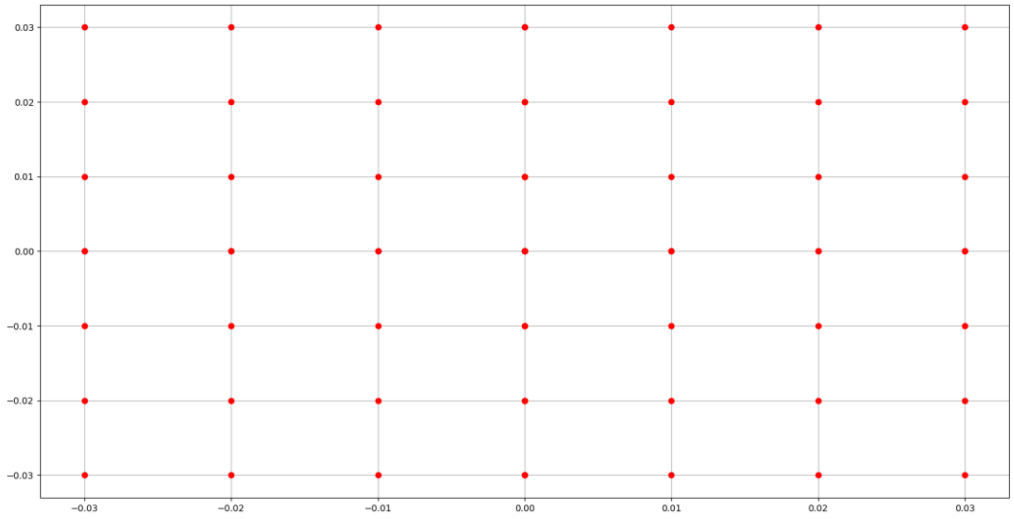


Fig 3.3.8 Grid search-based exploration of $[-0.03, 0.03]^2$ with 7 samples per axis

The grid search process can be refined by iterating several grid searches on increasingly smaller areas of the hyper-parameter space, each one identified as a promising hyper-parameter region from the previous grid search.

Although grid search methods can easily be implemented in parallel, significantly speeding the hyper-parameter tuning process, the computational cost of deep artificial neural networks typically prevent it on reasonably sized machines. In addition, the quality of grid-based exploration approaches tends to substantially decrease in high dimensional spaces that can typically be observed in deep neural network hyper-parameters.

Random search

Grid search methods methodically explore the entire bounded subspace that it is provided. As a consequence, and specifically in deep artificial neural network which present a high number of hyper-parameter, these processes are substantially computationally expansive. In addition, grid search methods are specifically adapted to tuning hyper-parameters with comparable impact on the model's

performances. However, it has been found that the influence of hyper-parameters on the quality of the final algorithm is substantially heterogeneous; oftentimes, some hyper-parameters have close to no influence on the model. As a consequence, random searches-based exploration methods have been found to significantly outperform their grid-based counterparts in the fine-tuning of hyper-parameters in deep artificial network-based models.

The optimization of a given model's hyper-parameter through random search is fairly straightforward to implement, and consists simply in sampling random combination of hyper-parameters, assessing their quality on the development set and selecting the combination yielding best model performances. More formally, for a given model with n hyper-parameters $(\lambda_i)_{0<i<n+1} \in \mathbb{R}^n$ a grid search can be implemented as following:

1. Define a number of hyper-parameters set to sample m,
2. For each hyper-parameter λ_i , define a bounded interval to sample from,
3. Sample m hyper-parameter set from their respective sampling intervals, typically through a uniform random distribution,
4. Train a model with the training set for every hyper-parameter combination,
5. Select the best set of hyper-parameters as maximizing the model's accuracy on the development set,
6. Test the model's accuracy on the test set.

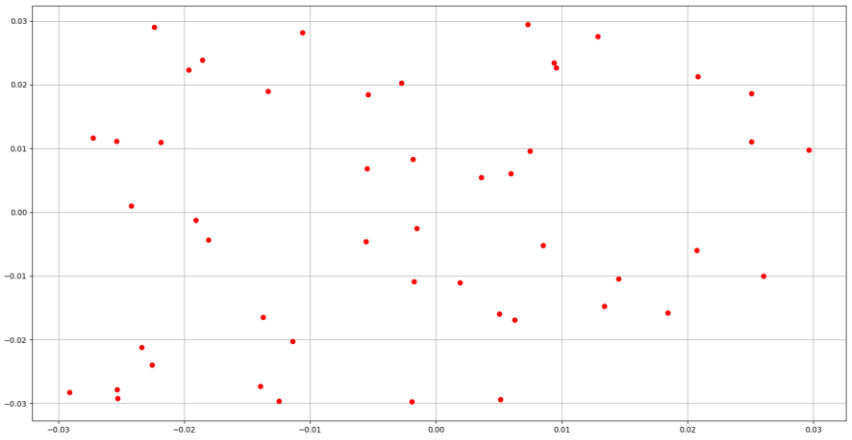


Fig 3.3.9 Random search based exploration of [-0.03, 0.03]^2 with candidates sampled from a bivariate uniform random distribution. The same number of candidates were sampled as in the previous grid search visualization example

In a similar manner to grid search methods, a random search process can be refined by iterative random searches in increasingly smaller areas of the hyper-parameter space identified from the previous search result as promising. Several approaches are available to identify such areas of the hyper-parameter space, from simple exploratory analysis of the hyper-parameter space relationship to the validation error to the use of an additional neural network in a reinforcement learning setting, the latter being extremely computationally intensive.

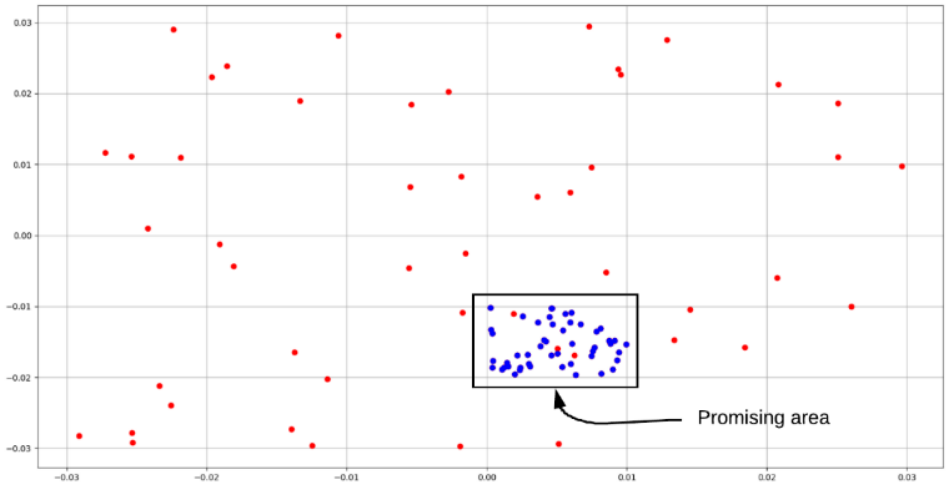


Fig. 3.3.10 Refined random search. A first random search (in red) is performed. An area within the search is selected as promising (for instance because hyper-parameter sampled from it all yielded good model performance), and a new random search (in blue) is performed on selected area

The superiority of random search based hyper-parameter tuning compared to grid-based approaches mostly can be explained from a combination of two phenomena:

- Random search investigates a significantly higher number of different values for each parameter than grid search
- Some hyper-parameters might not have any significant effect on the model's performance

A simple example can be considered in order to better visualize the impact of these two properties on the resulting obtained set of hyper-parameters. Let $(\lambda_1, \lambda_2, \lambda_3)$ be a set of three hyperparameters with λ_3 having close to no significant impact on the model's predictive quality.

A grid search hyper-parameter optimization procedure with n values sampled for each hyperparameter will require to fit n^3 distinct models. However, due to λ_3's absence of effect on the model's performances, only n^2 *efficient* hyper-parameter (λ_1 and λ_2) settings will have been tested.

A random search process with n^3 random sampled hyperparameter sets, however, will have tested n^3 distinct combination of λ_1 and λ_2, thus resulting in a finer space exploration and, usually, better model performances.

3.4 Neural sequence models and machine translation

The neural architectures defined so far allow for the implementation of powerful predictive models, from either vectorial or structured input data, whether sequential or grid-like. However, as non-linear extensions to more traditional models such as linear, logistic or multinomial regression, they share their limitations. In particular, the modelling of sequential, interdependent outcome variables using these methods is not straightforward, while this type of data is ubiquitous in epidemiology, for instance in cohort studies. Neural sequence models constitute a family of models that extend the traditional feedforward architectures to allow for the implementation of predictive models with sequential outcome variables, whether with fixed or variable sequence lengths. In practical applications, these models are essentially used in natural language processing, specifically in machine translation, where they have been representing the state of the art for a number of years. The first powerful neural sequence models were based on recurrent neural networks and their variants, such as Long Short Term Memory units[40] and Gated Recurrent Units. Although the deep learning academic community have found empirical evidence that now discourages their use in natural language processing tasks in favour of attention[41] or dilated convolutions[42] based models, for instance. However, these modern approaches stay far from traditional statistical predictive models typically seen in epidemiology or biostatistics, and can appear quite opaque at first sight. The older, simpler architectures, however, constitute a perfect entry point to state of the art machine translation models, and can still be used to derive original tools for biostatistics, such as the novel recurrent neural network based ordinal

regression method described below, which constitutes an original work produced within the context of this thesis.

3.4.1 Learning a binary search with a recurrent neural network for ordinal regression

Ordinal regression is a well-known predictive modelling problem used in fields as diverse as psychometry[43] to deep neural network based voice modelling. Their specificity lies in the properties of their outcome variable, typically considered as a categorical variable with natural ordering properties, typically allowing comparisons between different states[44] ("a little" is less than "somewhat" which is itself less than "a lot", with transitivity allowed). Although this additional prior knowledge should be incorporated into the modelling process, this state comparability property ends up being surprisingly hard to integrate to pre-existing qualitative or quantitative approaches.

Indeed, most traditional ordinal regression methods (typically based on thresholding or least-square like modelling objectives) make additional assumptions on the outcome variable that might not always be verified[45]. As an example, the ordered logits model relies on the proportional odds assumption and the hypothesis that the observed ordered dependent variable constitutes an imperfect observation of a latent quantitative variable.

In computer science, the manipulation of ordered table is a well-known problem for which simple yet powerful algorithms have been known for decades. Binary search, for instance, allows for the localization of a given value in an ordered table of predefined size N using at most Log2(N) comparisons.

The following section proposes to make use of this simple algorithm's essence by encoding an ordered variable as a binary tree. The resulting modelling problem is then shown to be reminiscent of sequential models traditionally seen in the deep learning academic literature, and a recurrent neural network variant, the Gated Recurrent Unit[19] (GRU) is proposed to solve it. The predictive power of the investigated method is then assessed on a dozen openly available benchmark datasets. Comparison

with traditional methods show a significant improvement in predictive power on a number of datasets, in term of both average error rate, squared Cohen Kappa score and squared error metrics.

3.4.1.1 Method

Ordinal variable encoding on a probabilistic binary search tree

A binary search is as simple yet powerful recursive algorithm that, from a sorted array, determines the position of one of its given element, by computing a logarithmic amount of comparison[46] between the investigated value and the table's elements as can be seen in figure 3.4.1:

1. Select the median element of the table, and compare it to the investigated value
2. If the median element is bigger than the investigated value, apply steps 1 to 4 on the table's lower half
3. If the median element is lower than the investigated value, apply steps 1 to 4 on the table's upper half
4. If the median element is equal to the investigated value, stop the algorithm and return the median element's position

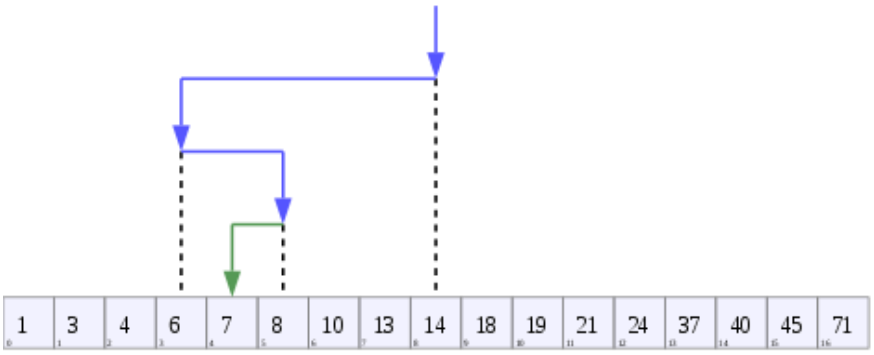


Fig. 3.4.1 Example of a binary search algorithm (source: Wikipedia). The algorithm needs only 4 comparisons to find the position of an element in a table of 16 elements

As a powerful approach to ordered sets manipulation relying solely on comparison operations, which are by definition perfectly acceptable in ordinal variable analysis, the binary search algorithm might constitute an interesting basis for the design of an ordinal regression method. For instance, directly applying the algorithm to an ordinal variable allows for its encoding on a binary tree. Figure 3.4.2 shows an example of such a decomposition, with an 8 states ordinal variable. Each path on the tree corresponds to a sequence of binary random variables defined as comparisons.

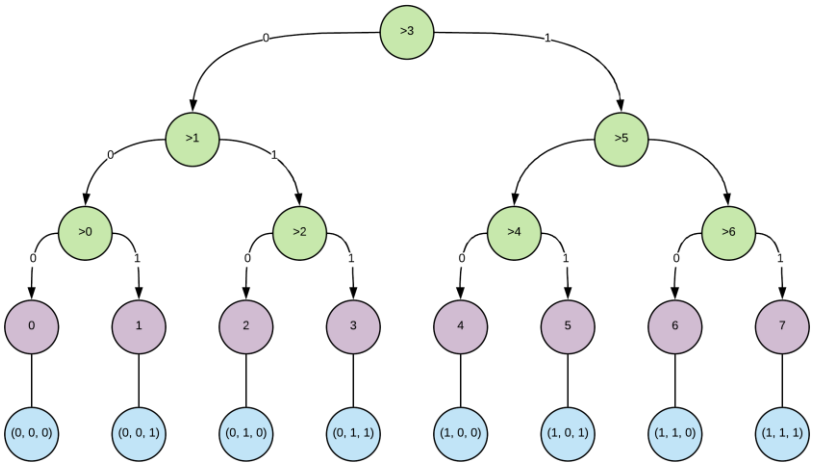


Fig. 3.4.2 Binary search tree. Each state of the ordinal variable is associated with a vector of binary variables representing its location on the tree, and the result of the equivalent binary search process. Note that binary vectors correspond exactly to the binary decomposition of the ordinal variable's state

By considering this binary search tree as a standard conditional tree diagram, and identifying the decision path leading to a given value corresponds to its decomposition in binary, ordinal regression can be formulated as a sequential modelling problem of binary variables as follows:

$$P(y \mid X, \theta) = P\left(\left\{\left\lfloor \frac{y}{2^{n-i}} \right\rfloor \bmod 2, i \in [\![1, n]\!]\right\} \mid X, \theta\right) \quad (3.4.1),$$

$$= P\left(\left\lfloor \frac{y}{2^{n-1}} \right\rfloor \bmod 2 \mid X, \theta\right) \prod_{i=2}^{n} P\left(\left\lfloor \frac{y}{2^{n-i}} \right\rfloor \bmod 2 \mid \left\{\left\lfloor \frac{y}{2^{n-j}} \right\rfloor \bmod 2, j \in [\![1, i]\!]\right\}, X, \theta\right) \quad (3.4.2),$$

$$= P\left(B_{1,n} \mid X, \theta\right) \prod_{\substack{i=1 \\ \in \mathbb{N}^2}}^{n} P\left(B_{i,n} \mid \{B_{j,n}, j \in [\![1, i]\!]\}, X, \theta\right) \text{ with } B_{i,n} = \left\lfloor \frac{y}{2^{n-i}} \right\rfloor \bmod 2 \ \forall (i, n) \quad (3.4.3).$$

Where:

- $y \in \mathbb{N}$ the ordinal dependent variable with $2^n, n \in \mathbb{N}$ states
- $X \in \mathbb{R}^d,\ d \in \mathbb{N}$, the explanatory variables (in vectorial form)
- $\theta \in \mathbb{R}^e,\ e \in \mathbb{N}$, the model's parameters

Note that so far, and for simplicity, the modelling problem is only defined for ordinal variable with a number of states that is a power of two (in other words where the ordinal variable's corresponding binary search graph is full). Extending the model to the general case of any given number of states is however quite straightforward and can be achieved as follows (and is shown in figure 3.4.3):

1. Model the $n \in \mathbb{N}$ states ordinal variable as having $\lceil 2^{log_2(n)} \rceil$ states,
2. Force all the unnecessary states to 0 after model inference and renormalize the resulting probability distribution.

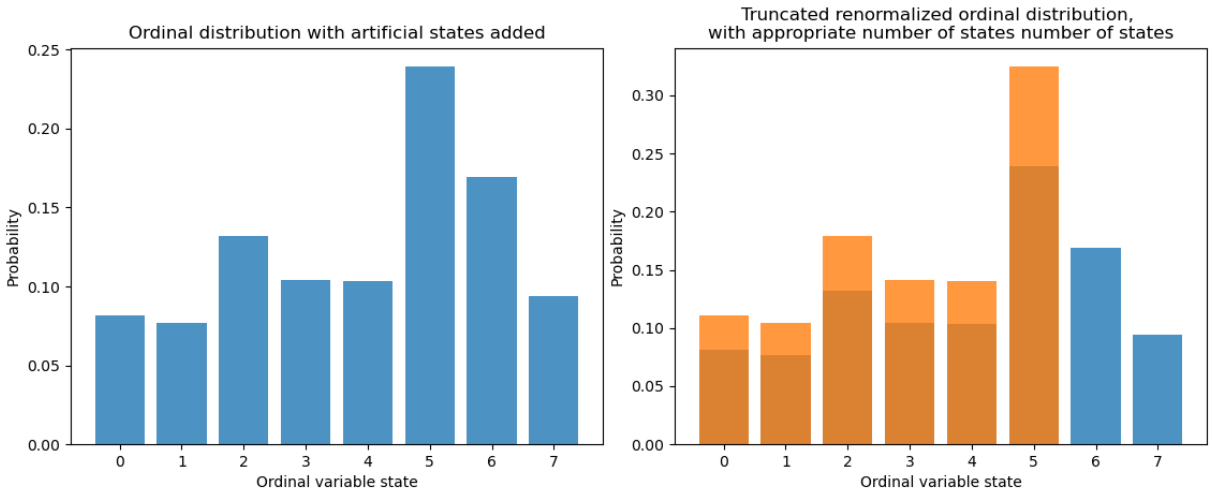


Fig. 3.4.3: Example of truncation and renormalization trick for an ordinal variable with 6 different states. The variable is projected into a binary search tree of depth 3 (on the left). The resulting probability distribution is then renormalized such that the two higher (and impossible) states are associated to 0 probability (on the right)

Model definition

Amongst the basic recurrent neural network based architecture described in 3.2.6, the autoregressive setting seems like a good candidate to model the joint probability of observing a sequence of event. Indeed, sequentially modelling all the output variables conditioned on all previous ones almost allows for the computation of the joint modelling by simple product of all derived probabilities. However, a problem arises with simple autoregressive models that prevents their use as is in the investigated modelling problem. Autoregressive models expect the first sequence element as a given, which is not the case in the investigated modelling problem, where $P(B_{0,n})$ requires an estimate as well. The neural machine translation literature, which encounters the same problem, introduced the idea of adding a "neutral" state to the categorical variable (typically denoted as "<START>" in the machine translation academic literature) from which the network starts its auto-regression process[47]. This additional value is given to the recurrent network as its first input element, from which the model learns to predict the sequence's actual first element, as can be seen in figure 3.4.4.

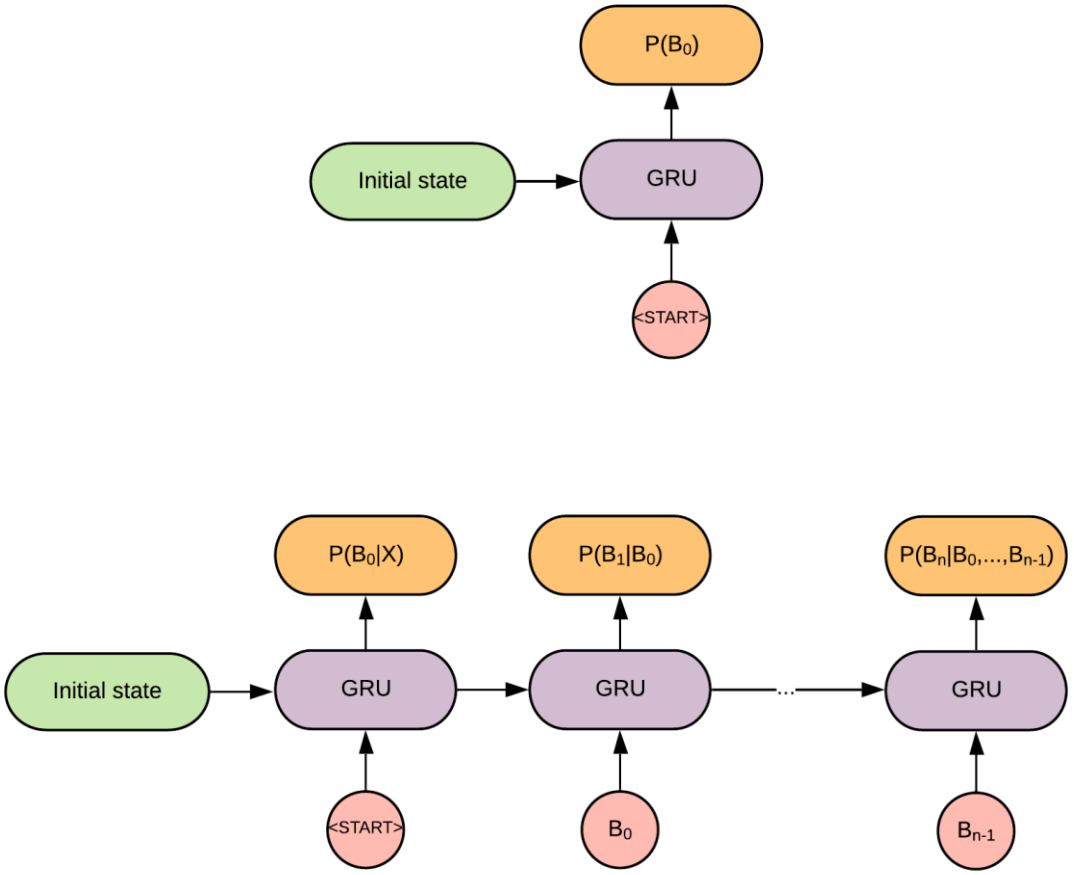


Fig. 3.4.4 Left: A GRU recurrent neural network during the first step of the joint modelling autoregressive process. Its input is an additional, artificial state given to the target variable that never changes from one individual to another, that is used to estimate target sequence's first element's discrete probability density. Right: After the first recurrent neural network iteration, the actual sequence is given to the model, apart from the last element, which is never conditioned upon

Although this neural architecture allows for efficient sequence joint probability modelling, it is not by itself sufficient in order to solve the modelling problem investigated here, which, as defined in 2.1, consists in estimating the joint probability of binary decision sequence *conditioned* on some explanatory variables. The machine translation literature academic also had the same problem (e.g.

estimating the probability of a sentence in French given a sentence in English) and came up with several solutions. The simplest, found in early RNN based encoder-decoder architectures, was to make the recurrent neural network's initial state a function of the input variables, as can be seen in figure 3.4.5 which is the solution that was chosen for the here defined architecture.

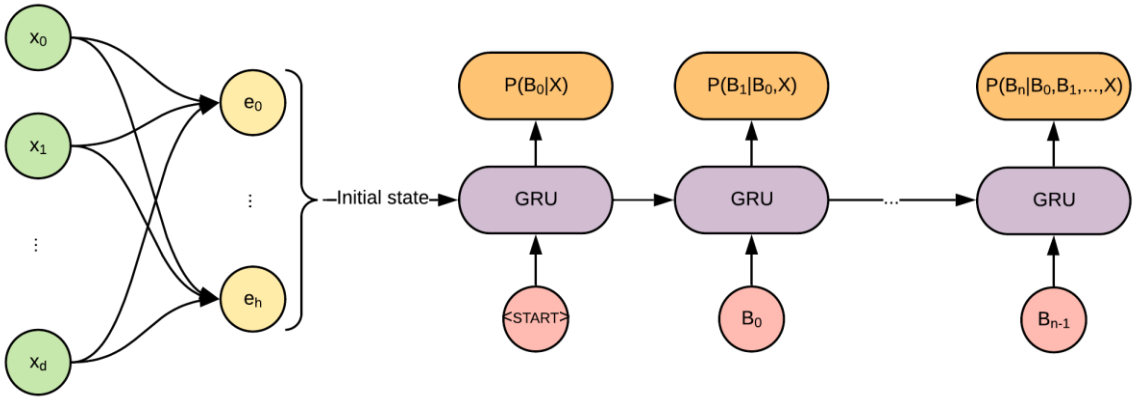


Figure 3.4.5: A GRU neural network able to estimate the joint probability of a sequential output variable conditioned on some explanatory variables. h linear combinations of the explanatory variables are used as the initial state of a GRU that sweeps through the padded target sequence in an autoregressive fashion

In summary, for an ordinal variable Y with $2^n, n \in \mathbb{N}$ and explanatory variables $X \in \mathbb{R}^d$, $d \in \mathbb{N}$, the entire model can be defined as follows, and its schematic representation can be seen in figure 3.4.6:

- The aforedefined random variables $B_{i,n}, i \epsilon [\![1, n]\!]$ are encoded as two valued one hot vectors ($(0,1)$ for $B_{i,n} = 0$, $(1,0)$ for $B_{i,n} = 1$)
- The neutral state used as a "<START>" token for the autoregressive token is defined as $(0,0)$
- A GRU recurrent neural network with dimensionality h is defined to sweep through the binary variable sequences (padded with the neutral state). h constitutes the model's only hyper parameter (the authors advise to set this value to n, although without any theoretical nor empirical evidence to back it up)

- h linear combination of the input variables are defined to build a vector E that is to be used as the GRU's initial state
- A logistic regression is then applied to all of the GRU's outputs (the same logistic regression is applied at each time-step) in order to estimate the probability of each $B_{i,n}, i \in [\![1, n]\!]$ in an autoregressive fashion
- By feeding all possible binary decomposition sequences to the GRU (for the same individual), the ordinal variable probability can be retrieved from the logistic regression output's products
- The entire model (logistic regression, GRU and linear combination parameters) is jointly fit though maximum likelihood with gradient descent and backpropagation

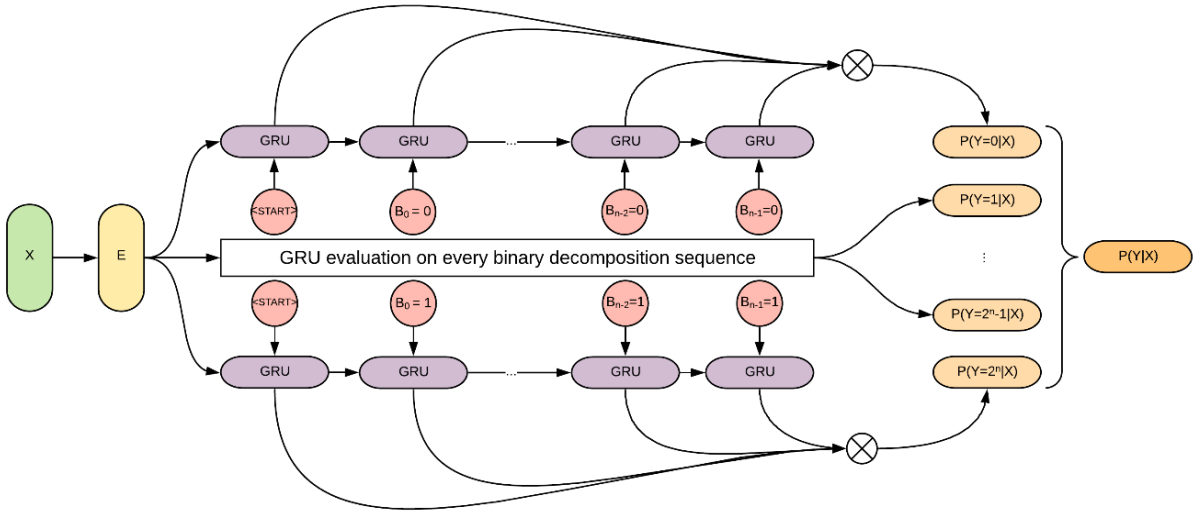


Fig. 3.4.6 The proposed neural architecture. From the explanatory variable, h linear combinations are used to initialize a GRU that sweeps through all of the ordinal variable state's binary decompositions. The latter's actual distribution is then retrieved through product of all of the GRU's outputs

Teacher forcing

The necessity of evaluating the recurrent neural network on every possible sequence in order to build the final probability distribution can result in significant computational needs, especially during model

fitting, where model inference and gradient computation through backpropagation is required at each gradient descent iteration step. In order to speed up computation times, neural translation model are typically trained nowadays using a technique called teacher forcing[48]. Instead of fitting the model through maximum likelihood on the final joint distribution, model parameters are inferred by maximum likelihood on the sequential variables, and only the correct sequence is given to the model for each observation, as can be seen in figure 3.4.7. As a consequence, each optimization step requires the recurrent neural network to only assess one sequence per observation, thus significantly improving computation times.

All experiments reported here were derived using this model fitting approach. The final predictions used for performance estimations were however derived from the more traditional approach of building the entire ordinal probability distribution.

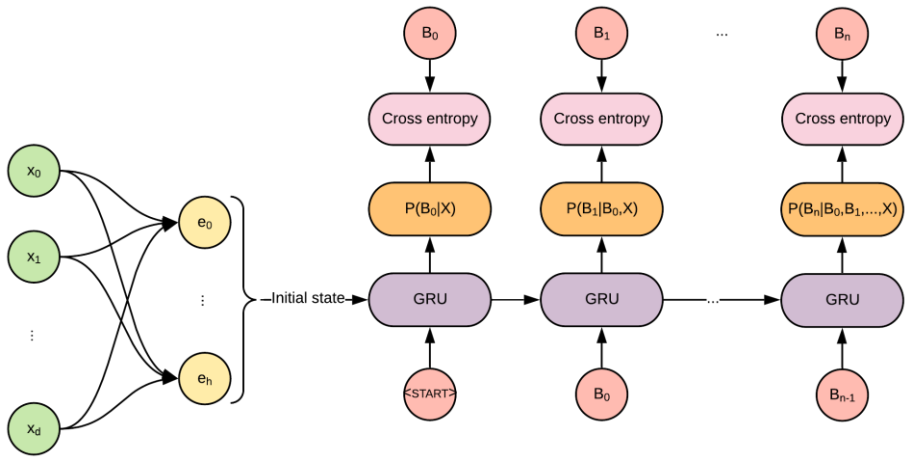


Fig. 3.4.7 Example of model training using teacher forcing. The GRU's initial state is derived from linear combination of the input variables as before. However, to improve computation time, only the correct output sequence is fed to the recurrent neural network. The model is then jointly fit through maximum likelihood on each individual $B_{i,n}, i \in [\![1, n]\!]$

3.4.1.2 Linear dimensionality reduction and visualization

The method proposed here was primarily intended as a purely discriminant model. However, its inherent architecture can be exploited (at least in a majority of use cases) to use it as a linear dimensionality reduction and visualization tool as well, with no additional work required. Indeed, the GRU-cell's initial state, a vector based on linear combinations of the explanatory variable whose dimensionality constitutes the model's only hyper parameter, contains all the information the trained model uses to predict the target ordinal variable. Consequently, as long as this initial state's dimensionality is set to a value lower than the number of explanatory variables, it constitutes a compressed representation of the explanatory variables built to preserve as much discriminative information regarding the target variable as possible. This approach shares some similarity (at least conceptually) with partial least square regression methods, but is here specific to ordinal valued target variables.

In addition, when setting the model's hyper-parameter to values 2 or 3, the subsequent linear projection of the explanatory variables can be plotted, which allows for the visualization of potentially insightful patterns regarding the relationship between explanatory variables and the ordinal target.

In short, these linear projections of the explanatory variables can be obtained as follows:

- Set the model's hyper-parameter to a value lower than the explanatory variables cardinality
- Train the model
- Discard the GRU cell from the model and only compute the linear projections used to build its initial state

Finally, as this projection is purely linear, the authors have some hope that they can retain some interpretability, which is quite rare in neural network models. However, additional work is required to properly assess how these linear combinations can be interpreted.

3.4.1.3 Experiments

Description

To assess its predictive performances, the aforementioned method was applied on a set of readily available benchmark datasets for ordinal prediction, and compared to results obtained from more traditional approaches. All the ordinal regression methods used in the following experiment (except for the one introduced in this chapter) were taken from the mord Python package, and all roughly follow two different approaches:

- Threshold based methods, comprised of three variants of the ordinal logistic model, the all-threshold ordinal logistic model, the immediate-threshold ordinal logistic model and the squared error ordinal logistic model, which are referred to as “AT”, “IT” and “SE” in the experiment’s results
- The regression based methods, comprised of the ordinal ridge regression model, and the least absolute deviation ordinal regression model, which are referred to as “Ridge” and “LAD” in the experiment’s results

In order to assess the proposed method’s performances in comparison to the state of the art, all these methods were used on 17 real-life datasets traditionally used for benchmarking ordinal regression methods. A summary of these datasets can be found in table 3.1. The following methodology was used for the experiment:

- Each input variable in all dataset was standardized to zero mean and unitary standard variance
- Every model (the proposed approach, and the mord package’s model) was fit to each dataset and three performance metrics were assessed using 10-fold cross validation: Model accuracy, squared Cohen Kappa score and squared error. Confidence intervals were estimated through bootstrap

Dataset	Number of ordinal states in the output variable	Number of input variables	Sample size
Abalone	8	7	4177
Abalone_ord	10	10	4177
Affairs	6	17	265
Ailerons	9	39	7154
Auto_ord	10	8	398
Auto_riskness	6	15	160
Bostonhousing_ord	5	13	506
Boston_housing	6	13	506
California_housing	6	8	20640
Cement_strength	5	8	998
Fireman_example	16	10	40768
Glass	6	9	213
Kinematics	8	8	8192
Machine_ord	10	6	199
Skill	7	18	3337
Stock_ord	5	9	950
Winequality_red	6	11	1359
Winequality_white	7	11	3961

Table 3.1: Datasets summary

The “Wisconsin_breast_ord” dataset, also readily available in the same source for ordinal regression method benchmarking, was discarded for the experiments due to its low observation to sample size ratio (only 194 observations for 33 variables). Indeed, as it stands now, the ordinal regression method presented in this dataset is not meant as a tool for scarce datasets as Ridge or Lasso regressions are. However, the application of such penalty based regularization methods will be the object of future work.

An additional experiment was designed in order to assess the model’s visualization capability. A supplementary model with hyper-parameter value set to two was adjusted to each of the aforedescribed dataset. The resulting bi-dimensional embedding were then plotted against the ordinal target value in order to qualitatively assess whether these linear projections can indeed capture interesting patterns in the data.

Results

Predictive performances

The predictive performance experiment's results are displayed in tables 3.2, 3.3 and 3.4 for accuracy, squared Cohen Kappa and mean squared error metrics respectively. For readability, only the best baseline method's performance metrics are reported for each dataset, and scores showing a significantly better performance are highlighted in bold. The interested reader can however find the experiment's complete results in the annex. For 7 datasets (namely "Auto_riskness", "Boston_housing", "Bostonhousing_ord", "Glass", "Machine_ord", "Skill" and "Winequality_red"), no significant difference in predictive performance could be found between the proposed approach and the best baseline method in all investigated metrics. For the 10 remaining datasets, significant differences in predictive power were found, and can be summed up as follows:

- For four datasets, namely "Cement_strength", "Fireman_Example", "Kinematics" and "Stock_ord"), the proposed approach significantly outperformed the best baseline method on all metrics
- For one dataset, namely "Affairs", the proposed approach was significantly outperformed by at least one baseline method on all assessed metrics. However, the best baseline approach for this dataset differs for all metrics (Logistic IT for accuracy, logistic AT for Cohen Kappa, and Ridge for the mean squared error)
- When focusing only on accuracy, the proposed approach outperforms all baseline methods on an additional three datasets ("Abalone", "Abalone_ord" and "California_housing"), and is not outperformed by any of them on any additional dataset beside "Affairs"
- When focusing only on squared Cohen Kappa score, the proposed approach outperforms all baseline methods on an additional two datasets ("California_housing" and

"Winequality_white"), and is only outperformed by any of them on any additional dataset beside "Affairs"

- When focusing only on mean squared error, the proposed approach does not significantly outperform all baseline methods on any additional dataset. It is however outperformed by an additional dataset, namely "Abalone"

Dataset	Proposed	Best other	Method
abalone	**37.6 [36.1, 39.1]**	32.8 [31.4, 34.1]	IT
abalone_ord	**58.7 [57.2, 60.1]**	55.2 [53.7, 56.7]	LAD
affairs	25.0 [20.0, 30.4]	**48.1 [41.9, 54.2]**	IT
ailerons	45.5 [44.3, 46.7]	43.2 [42.0, 44.3]	IT
auto_ord	51.0 [46.2, 55.9]	55.4 [50.5, 60.3]	SE
auto_riskness	66.9 [59.4, 73.8]	63.1 [55.6, 70.0]	LAD
bostonhousing_ord	73.6 [69.6, 77.4]	72.0 [68.0, 75.8]	AT
boston_housing	56.6 [52.2, 61.0]	61.6 [57.4, 65.8]	IT
california_housing	**57.9 [57.3, 58.6]**	54.0 [53.3, 54.7]	AT
cement_strength	**69.1 [66.2, 71.9]**	49.4 [46.2, 52.5]	LAD
fireman_example	**39.9 [39.4, 40.4]**	23.0 [22.6, 23.5]	IT
glass	57.1 [50.5, 63.8]	56.2 [49.5, 62.9]	IT
kinematics	**42.7 [41.6, 43.8]**	27.4 [26.5, 28.4]	IT
machine_ord	57.4 [50.5, 64.2]	66.3 [59.5, 73.2]	AT
skill	41.6 [39.9, 43.3]	40.5 [38.7, 42.1]	AT
stock_ord	**85.7 [83.5, 87.8]**	69.8 [66.8, 72.6]	IT
winequality_red	57.9 [55.2, 60.6]	58.0 [55.3, 60.6]	LAD
winequality_white	53.9 [52.4, 55.5]	52.9 [51.3, 54.4]	LAD

Table 3.2: Accuracy results (in %)

Dataset	Proposed	Best other	Method
abalone	74.6 [73.0, 76.1]	72.9 [71.4, 74.4]	AT
abalone_ord	62.5 [60.5, 64.5]	63.6 [61.6, 65.5]	Ridge
affairs	-11.1 [-23.0, 1.3]	**23.0 [12.2, 33.7]**	AT
ailerons	89.5 [89.0, 90.1]	89.6 [89.1, 90.1]	AT
auto_ord	88.5 [86.0, 90.6]	91.1 [89.3, 92.6]	SE
auto_riskness	61.3 [44.2, 75.6]	66.2 [56.3, 74.7]	AT
bostonhousing_ord	82.3 [77.6, 86.3]	82.5 [78.4, 85.9]	SE
boston_housing	85.9 [82.7, 88.6]	87.3 [84.4, 90.0]	IT
california_housing	**79.1 [78.4, 79.8]**	77.7 [77.0, 78.4]	AT
cement_strength	**88.0 [86.4, 89.5]**	71.3 [68.0, 74.2]	IT
fireman_example	**96.3 [96.2, 96.4]**	84.3 [84.0, 84.7]	AT
glass	71.3 [60.5, 80.1]	80.4 [73.4, 85.7]	LAD
kinematics	**84.5 [83.7, 85.3]**	62.0 [60.6, 63.4]	IT
machine_ord	80.7 [67.8, 89.9]	92.4 [87.0, 95.7]	SE
skill	73.0 [71.4, 74.6]	70.6 [68.9, 72.2]	IT
stock_ord	**95.1 [94.2, 95.9]**	88.6 [87.0, 90.1]	IT
winequality_red	50.4 [46.1, 54.5]	49.5 [45.8, 53.2]	LAD
winequality_white	**49.0 [46.8, 51.2]**	43.2 [40.9, 45.6]	LAD

Table 3.3: Quadratic Cohen Kappa results (in %)

	Proposed	Best other	Method
abalone	2.46 [2.33, 2.59]	**2.13 [2.03, 2.22]**	SE
abalone_ord	0.77 [0.72, 0.83]	0.76 [0.71, 0.81]	Ridge
affairs	7.90 [6.83, 9.01]	**3.45 [2.99, 3.93]**	Ridge
ailerons	1.45 [1.38, 1.52]	1.35 [1.30, 1.41]	SE
auto_ord	0.98 [0.78, 1.20]	0.73 [0.60, 0.87]	SE
auto_riskness	1.14 [0.69, 1.71]	0.81 [0.62, 1.03]	AT
bostonhousing_ord	0.38 [0.29, 0.47]	0.34 [0.28, 0.40]	SE
boston_housing	0.73 [0.61, 0.88]	0.64 [0.52, 0.78]	IT
california_housing	0.76 [0.74, 0.78]	0.77 [0.75, 0.79]	SE
cement_strength	**0.35 [0.31, 0.39]**	0.74 [0.68, 0.81]	SE
fireman_example	**1.58 [1.55, 1.61]**	6.09 [5.99, 6.19]	SE
glass	1.67 [1.18, 2.24]	1.01 [0.77, 1.29]	Ridge
kinematics	**1.66 [1.59, 1.74]**	3.22 [3.13, 3.31]	SE
machine_ord	1.92 [1.02, 3.18]	0.79 [0.44, 1.25]	SE
skill	1.02 [0.96, 1.07]	1.03 [0.98, 1.08]	SE
stock_ord	**0.14 [0.12, 0.17]**	0.32 [0.29, 0.36]	IT
winequality_red	0.56 [0.51, 0.61]	0.53 [0.48, 0.57]	LAD
winequality_white	0.61 [0.58, 0.65]	0.65 [0.62, 0.69]	LAD

Table 3.4: Mean squared error results

In order to provide better insight on the proposed approach's performances against all baseline methods, figure 3.4.8 displays for each dataset which of the baseline methods either significantly outperform, is outperformed, or does not perform significantly differently than the proposed approach, for all chosen metrics.

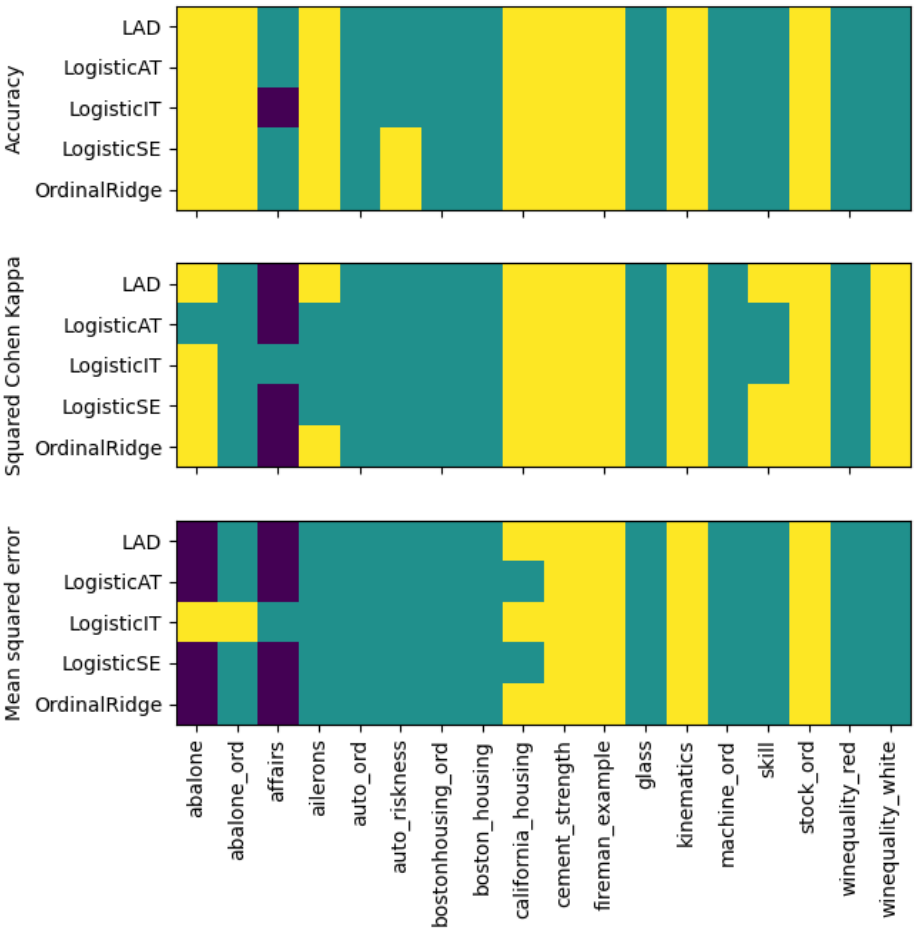


Fig. 3.4.8 Results comparisons between the proposed approach and all baseline methods on all datasets. Yellow, blue and purple cases denote baseline methods that respectively perform significantly worse, not significantly better or worse and significantly better than the proposed approach on the given dataset

As can be seen on figure 3.4.8, the proposed approach significantly outperforms any baseline approach on any given dataset for a total number of 106 times, and is significantly outperformed 13 times. Moreover, the binary search based method is never significantly beaten by all baseline methods on any of the investigated datasets, this for all chosen metrics.

As was already shown in table 3.4, the proposed approach's performances in term of mean squared error are a bit weaker. Indeed, it is outperformed by all baseline methods besides the "LogisticIT" on both the "Abalone" and the "Affairs" datasets. These poorer performances might be explained by the fact that the proposed method's approach does not rely on any mean squared (or mean squared surrogates) objective for model fitting. In any cases, additional analysis of these datasets to better understand these poorer performances will be treated in the discussion.

Linear dimensionality reduction for ordinal visualization

As previously described, in order to assess the proposed approach' potential for data visualization, a set of additional models were fit to each dataset, with hyper-parameter set to 2 to allow for efficient scatterplot.

Some of the resulting bi-dimensional projections of the input data can be seen in figure 3.4.9, with each point colour-mapped according to its ordinal target variable value. The remaining visualizations can be found in the annex, with varying results. For instance, as the "affairs" dataset suffered from extremely poor prediction performances, it is reasonable to expect its resulting projection to yield few to no insight about the relationship between target and explanatory variables.

Insights gathered from the visualizations displayed in figure 3.4.9 can be summed up in two major points:

- When it comes to the "Fireman example" and "Ailerons" datasets, the relationship between the target variable and the explanatory variable linear combinations is quite smooth and

progressive. The “Fireman example” dataset however appears to have non-linear decision boundaries

- For the “Boustonhousing ord” and “Stock ord” datasets, however, the relationship between explanatory and target variables is not as straightforward. Indeed, the visualizations show clusters of data points that each keep an ordered relationship with the target variable. However, the decisions boundaries are not the same for all clusters, indicating that stratification might be of interest in the analysis of these datasets.

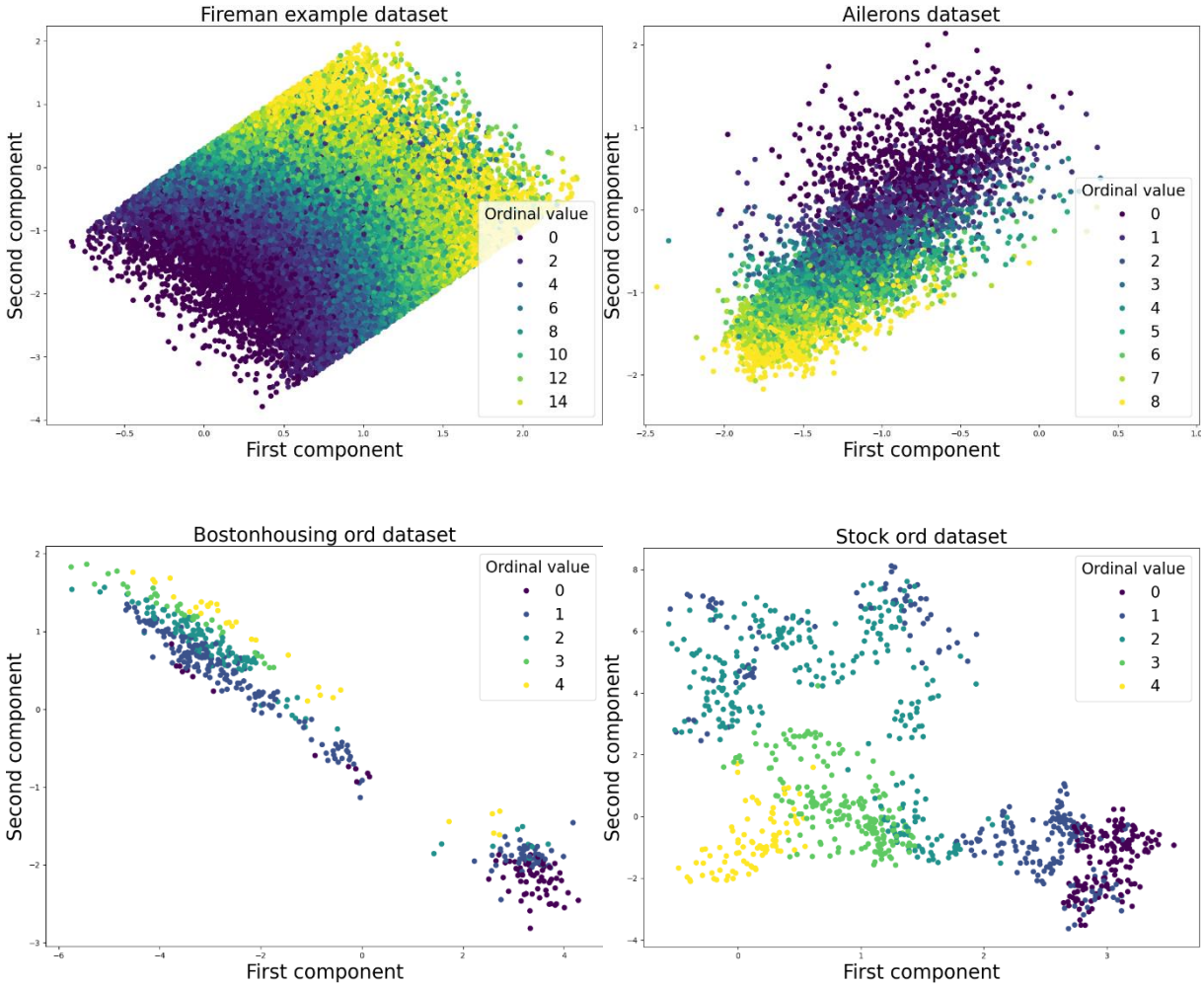


Fig. 3.4.9 Examples of bi-dimensional projections obtained by fitting the proposed model (with number of neurons in the recurrent network parameter fixed to 2) to a selected sample of datasets used in the experiment. For each dataset, the linear projections lead to highly readable visualization of the explanatory variables’ relationship to the ordinal target variable

3.4.1.4 Discussion

As previously seen in the result part, the proposed approach tends to yield better or similar predictive performances to all baseline approach on most datasets, with the exception of the « affairs » and « abalone » datasets. Consequently, developing a better understanding of these datasets might be helpful in order to assess cases where the approach for ordinal regression presented in this chapter might not be advisable. In addition, it might also lead to empirically derived conjectures regarding hypotheses the model requires in order to perform well, or provide elements that might lead to further improvements.

"Affairs" dataset

The "affairs" dataset constitutes the dataset on which the proposed approach' overall performances are the lowers. Indeed, it is the only investigated dataset where recurrent neural network based ordinal regression is outperformed by at least one baseline method for all selected performance metrics.

The first thing that can be noticed about the "affairs" dataset is its low sample size compared to its high number of explanatory variables. Indeed, this dataset is comprised of 265 observations each comprised of 17 explanatory variables. In addition, its ordinal target variable is made of 6 different states. Such a poor dimensionality to sample ratio typically requires regularization methods. In addition, neural network based methods for predictive modelling are known to easily overfit. However, no regularization methods were used during model training in the experiments presented in this chapter, which might explain the model's poor performance. This hypothesis is further confirmed by assessing the model's differences in performance between the training and validation dataset that are displayed in table 3.5. Indeed, a significant gap can be observed between training and validation metrics, and constitutes strong evidence indicating the model is overfitting the dataset.

Dataset	Accuracy (%)	Cohen Kappa	Mean Squared Error
Training	54	.42	4.1
Validation	27	-.53	7.9

Table 3.5: Training and validation performance metrics for the "affairs" dataset. The significant decrease in performance from training to validation suggests that the model is strongly overfitting the dataset

As a consequence, incorporating regularization methods in the proposed approach should be the object of future work. A promising candidate to do so can be found in the dropout method, traditionally used in deep learning models (see paragraph 3.3.2.2), which could be applied to the recurrent neural network part of the presented architecture. Another solution that might be of interest lies in penalized methods. Indeed, one could add a lasso, ridge or elastic-net penalization to the objective function. This penalization could typically be applied to the linear combination weights that are used to build the recurrent neural network's initial state. For a lasso penalization applied on these weights, for instance, the feature selection interpretation of lasso regression methods would remain heuristically valid.

Another remarkable property of the "affairs" dataset is that in addition to having considerably low sample size, it is considerably unbalanced. Indeed, as can be seen in figure 3.4.10, more than half of the dataset's observations are associated with a target value of 1, with some target values only observed as few as 15 times. Consequently, sampling or loss weighting techniques should be considered necessary in order to properly solve this modelling problem.

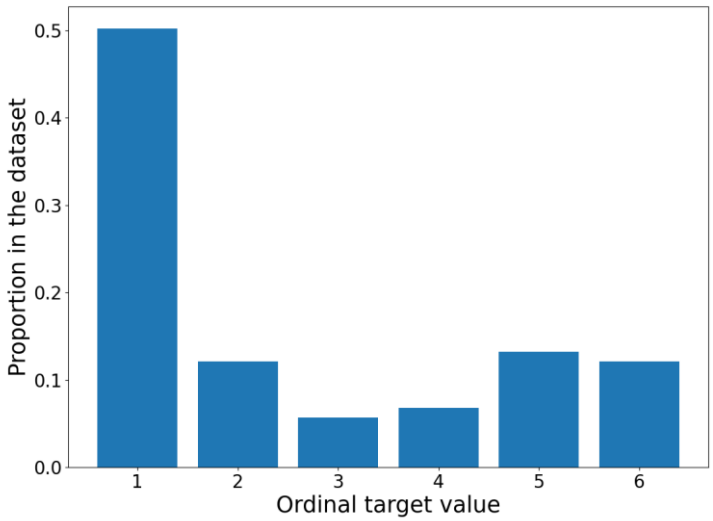


Fig. 3.4.10 Distribution of the ordinal target variable in the dataset. The dataset is extremely unbalanced. Approximately 50% of observations correspond to the first target value. All other values have less than 35 observations

Although sampling techniques can perfectly adapt to the proposed approach without any additional work, loss re-weighting techniques are not as straightforward, especially when training with teacher forcing. Indeed, in teacher forcing, the actual labels used to fit the model are the binary decomposition of the target values. As such, weighting methods should be adapted in order for weighting to apply for this sequence of target variable.

"Abalone" dataset

Although not as concerning in terms of predictive performance, compared to the "affairs" dataset, the model's behaviour on the "abalone" dataset is more complicated to explain. Indeed, the model is only significantly outperformed on mean squared error metrics, and significantly outperforms almost all baseline approaches on the two other selected metrics (apart from the "LogisticAT" model with regard to the quadratic Cohen Kappa score). However, the authors could not find any satisfactory explanation for this unique behaviour. Indeed, the dataset's dimensionality (7 explanatory variables for approximately 4 thousands data points) seems quite sufficient, which is further confirmed when

estimating performance metrics on the training set, which are essentially identical to those evaluated through cross validation. In addition, the distribution of the ordinal target variable does not suffer from severe unbalance such as could be observed with the “affairs” dataset.

However, the visualization capability of the proposed model can constitute a way to further analyse this dataset in order to build hypotheses that might explain this phenomenon, at least qualitatively. Figure 3.4.11 shows the two dimensional projection of the “abalone” dataset’s explanatory variables, with each point colour coded in three different manners:

- Each point colour coded according to the ordinal variable’s true value,
- Each point colour coded according to the proposed approach’ prediction,
- Each point colour coded according to the “LogisticSE”’s (best model in term of mean squared error) prediction.

Qualitatively, the proposed approach seems to derive decision surfaces that better fit the true ordinal values than the “LogisticSE” does (which is further confirmed by its significantly better performance in terms of accuracy). However, these boundaries are not entirely ordinal. Although decision boundaries for states 0 through 6 are organized in a fairly sequential, ordered approach, state 7’s decision surface appears to be adjacent to states 3, 4 and 6’s boundaries. This adjacency property is lost when using the baseline models, that all have as a hypothesis either a mean squared property or a proportional odds assumption (guaranteeing parallel decision surfaces). As such, the baseline approaches on this dataset might be biased towards maximizing the mean squared error at the detriment of finding true decision surfaces. As a consequence, one could emit the hypothesis that the ordinal properties of the “abalone” dataset’s target variables are not as clear cut as would be expected, and that mean squared error metrics are not quite perfectly fit to evaluate model performances on this particular dataset.

To conclude, figure 3.4.11 shows a potential use of the visualization for qualitative exploration of datasets in relation to an ordinal variable. Indeed, it allowed us to propose a hypothesis to explain the

model's behaviour on the "abalone" dataset by investigating the decision boundaries derived by different models. In addition, the reader can notice that the decision surfaces of the "LogisticSE" model are remarkably preserved by the projection, showing that these linear combination of the explanatory variables do capture efficient representations of the explanatory variables.

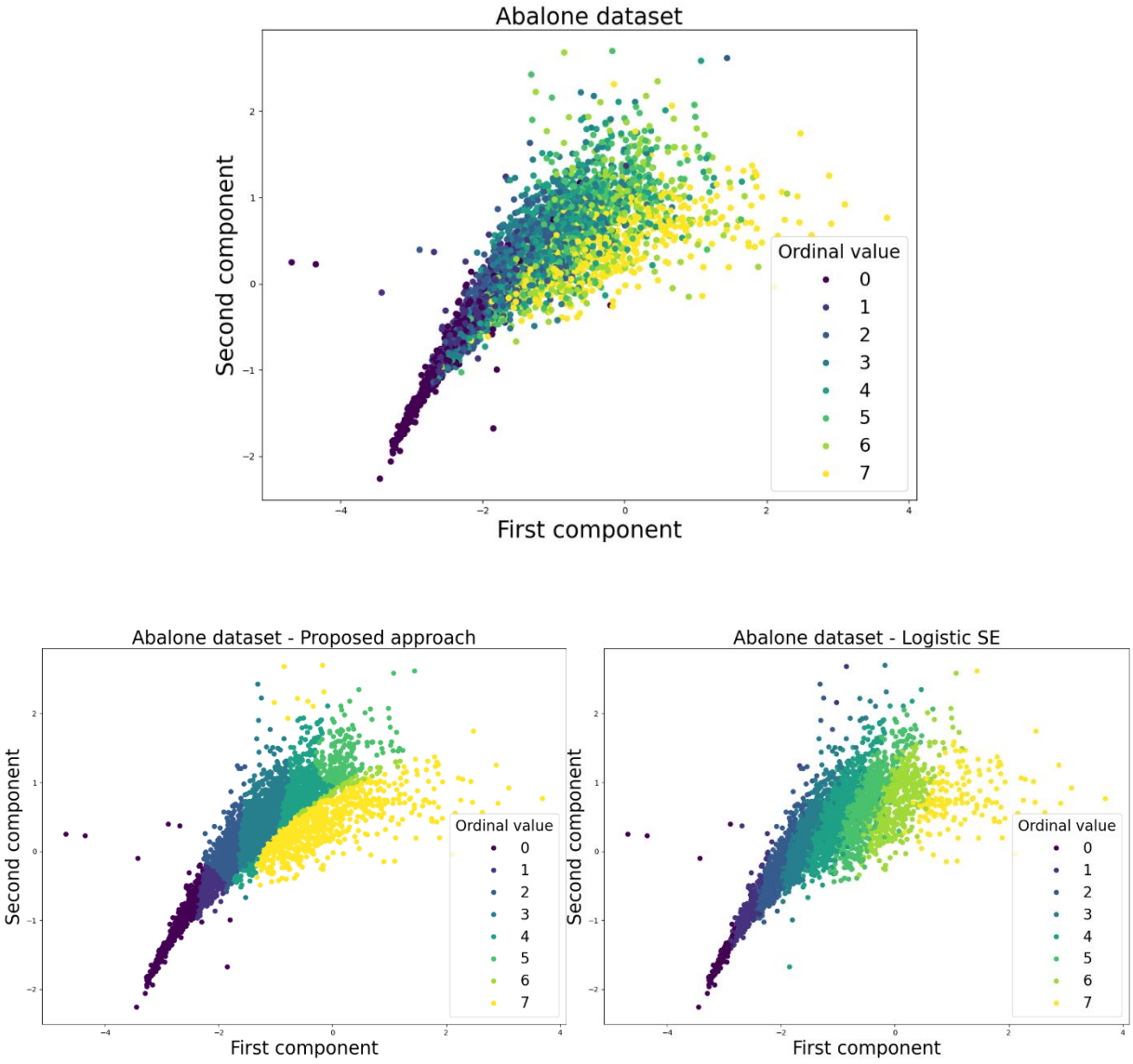


Fig. 3.4.11 Top: Bi-dimensional projection of the input variables color-coded according to the ordinal target values. Bottom-left: Same projections color-coded according to the proposed approach' predictions of the target variable. Bottom-left: Same projections color-coded according to the Logistic SE' predictions of the target variable.

3.4.2 Modern neural machine translation and Transformer architecture

Machine translation is a classical natural language processing thematic. Its main purpose is to design artificial intelligence based algorithms that are able to automatically translate one sentence from a given input language into another target language (e.g. Translating sentences from French to English). Countless approaches have been derived over the years to address this task, from expert system to modern deep learning approaches, and is still an active area of research.

At first glance, the use of deep neural networks in solving such a task can appear surprising. Indeed, as predictive models, deep artificial neural networks can only solve statistical modelling problems, and formulating a machine translation task as such is not so straightforward. A naïve, yet simple approach to do so, would be to consider both the input and output sentences as two categorical random variables, with as many states as there are possible sentences in their respective languages. One could then use a dataset of paired sentences in the two languages to estimate the target sentence's probability distribution conditioned on the input sentence's observation. The resulting modelling problem can be written as follows:

$$P(T|I) = f_\theta(I) \tag{3.4.4}$$

With:

- $T \in [\![0, 1]\!]^{C_1}$ the target sentence,
- $I \in [\![0, 1]\!]^{C_2}$ the input sentence,
- $C_1 \in \mathbb{N}$ the number of existing sentences in the target language,
- $C_2 \in \mathbb{N}$ the number of existing sentences in the input language,
- f_θ a mapping from the problem's input space to its output space, parameterized in $\theta \in \mathbb{R}^n$ a real-valued vector (typically a neural network) of dimensionality $n \in \mathbb{N}$ the model's dimensionality.

An immediate problem arises when estimating the number of potential sentences in a natural language. Indeed, considering a natural language sentence as a sequence of 10 words, drawn from a vocabulary of 20 000 potential words, results in values of N and M of around 10^{40}, which renders the modelling problem grossly intractable. For instance, using a simple multinomial logistic regression would result in a model of dimensionality 10^{80}.

This naïve approach, however, does not make use of the sequential nature of natural language, where sentences are usually built as variable lengths sequences of words. It only feels natural then to rewrite the aforedefined modelling problem, this time considering both the input and outcome variables as two sequential observations of a random variable each, the existing words in both languages. These variables can typically be expressed as categorical variables with as many states as there are words in each languages' vocabularies. The resulting modelling problem can then be defined by estimating the following probability distribution:

$$P(T_1 \dots T_N | I_1 \dots I_M) = f_\theta(I_1 \dots I_M) \quad (3.4.5).$$

With:

- $T_i \in [\![0, 1]\!]^{V_1}, i \in [\![1, N]\!]$ the i^{th}word present in the target sentence of length $N \in \mathbb{N}$,
- $I_i \in [\![0, 1]\!]^{V_2}, i \in [\![1, M]\!]$ the i^{th}word present in the input sentence of length $M \in \mathbb{N}$,
- $V_1 \in \mathbb{N}$ the target language vocabulary's cardinal,
- $V_2 \in \mathbb{N}$ the target language vocabulary's cardinal,
- f_θ a mapping from the problem's input space to its output space, parameterized in $\theta \in \mathbb{R}^n$ a real-valued vector (typically a neural network) of dimensionality $n \in \mathbb{N}$ the model's dimensionality.

Compared to the naïve formulation of the machine translation task as a statistical modelling problem, this sequential approach has both advantages and drawbacks:

- The dimensionality of both the input and categorical variables are significantly reduced. Indeed, the cardinal of natural languages vocabulary is typically counted in the tens of thousands of words, which remains significant. However, this can be made tractable, especially when using methods such as word embeddings,
- The parametric function f_θ now takes as input a vectorial sequence whose length varies, which is not typical of traditional statistical models. However, artificial neural networks are known to be perfectly able to handle such inputs, for instance with recurrent or convolutional neural networks,
- Even though the outcome variables' dimensionality is sensibly lower than that of the previous formulation, their joint density probability still remains intractable. However, this problem can be addressed using exactly the same method used previously, when modelling a conditional binary tree. Following the same rational, the investigated probability distribution can be written as follows:

$$P(T_1 \dots T_N | I_1 \dots I_M) = P(T_1 | I_1 \dots I_M) \prod_{n=2}^{N} P(T_n | T_1 \dots T_{n-1}, I_1 \dots I_M) \quad (3.4.6).$$

- This decomposition allows to train the model in a teacher-forcing manner, and by extension allows practitioners, following model fitting, to estimate the probability of a sentence in the target language to be a likely translation candidate for a given sentence in the input language. The entire distribution across all potential sentences, however, still remains intractable. As a consequence, finding the target distribution's maximal argument remains impossible. "Good" translation candidates (in the sense of candidates associated with high conditional probabilities), however, can still be derived from this approach using approximate methods such as greedy or beam search approaches that will be introduced in the following chapter.

3.4.3 Predicting sentences in neural machine translation

Although recurrent neural networks allow for sequential random variables joint probability distribution estimation, as seen in the previous example investigating ordinal regression, this method cannot be applied as is to the problem of machine translation due to its outcome's extreme dimensionality. However, given a trained model, estimating the probability of a given sentence being a good target language translation candidate to an input language sentence remains possible using the same approach. The ordinal regression method introduced above initialized a recurrent neural network's initial state as a function of its input variables, and would then build each of the outcome's state's probability in an autoregressive fashion. At no point, however, is it compulsory to build the entire distribution to estimate the probability of a given outcome's state (and, by extension, sequence). One could indeed simply choose a random sequence, and estimate its probability following the same method, as can be seen in figure 3.4.12.

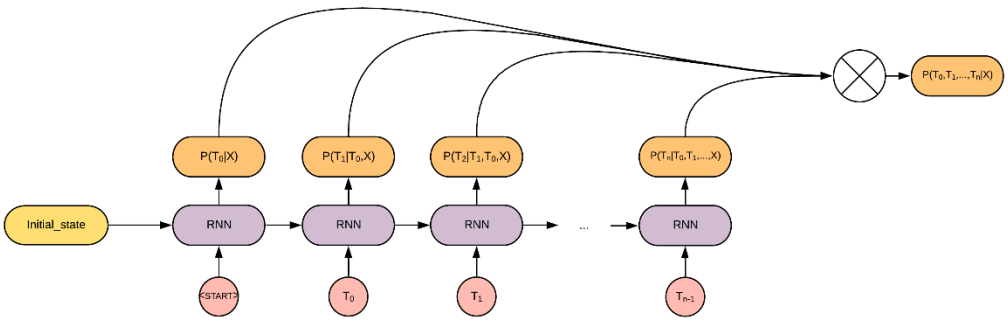


Fig. 3.4.12 Even though the entire probability distribution across sequences is intractable, estimating the probability of one predefined target sequence conditioned on an input sequence is still feasible

Exploiting this simple idea allows for guided exploration of the target sentence space in order to look for potential translation candidates, for instance by generating sentences in a purely autoregressive approach, beginning with the <START> token and predicting the sentence, one word at a time, by selecting the recurrent neural network's last output's maximal argument as the sentence's next word. This approach, however, presents a significant drawback when naively implemented. Indeed, contrary to the ordinal regression example, the target sequences in machine translation have undetermined length, implying that the model, as is, does not have any adequate stopping criterion when generating sentences, and could potentially get stuck in an endless loop. This problem is simply addressed by adding another token to the target language's vocabulary, called the <STOP> token. The stop token is appended to every target sentence during training, and the model learns to predict it just as any other word in the vocabulary, thus yielding a stopping criterion while predicting in an autoregressive fashion.

To sum up, a "good" target sentence can be predicted in this autoregressive fashion, called greedy prediction (because the sentence is built from locally optimal predictions), as follows:

- Use the encoded initial state and the <START> token to estimate the target sentence's first word's distribution using the decoder recurrent neural network
- Select the sentence's first word as this distribution's maximal argument
- Estimate the second word's distribution using this word as the sentence's first word
- Repeat the process in an iterative fashion until a <STOP> token is predicted (a global limit on target sentence size can be set in order to prevent potential endless loops)

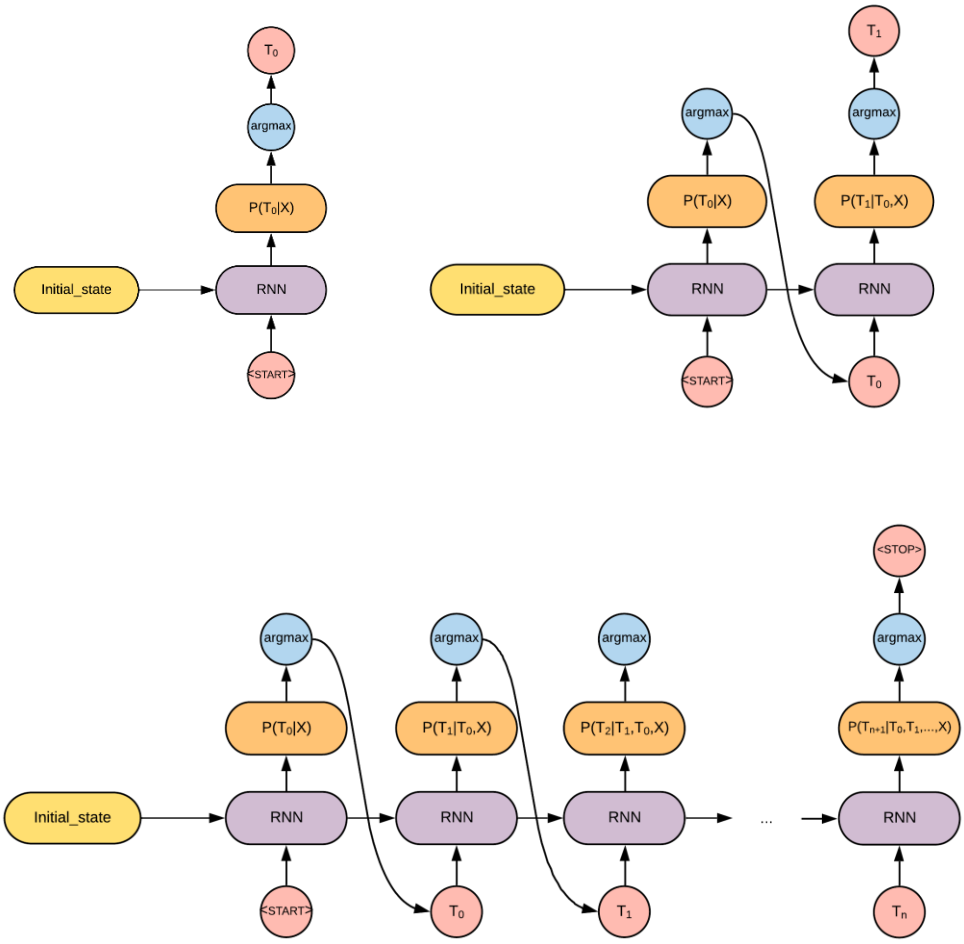


Fig. 3.4.13 Model in greedy prediction setting. The model predicts only one word at a time, starting from the initial state and the token <START>, and taking the resulting word distribution's maximal argument as the output sentence's first word. The token <START> and this word are then concatenated and reinjected into the recurrent neural network to obtain the sentence's next word. The sentence is then built incrementally following this method and stops once a token <STOP> has been predicted. The first recurrent neural network used to build the initial state is not displayed to help readability.

Although greedy search based prediction constitutes a simple way of performing machine translation using neural networks, it can be improved upon to better explore the space of potential target sentences, potentially leading to better predictions. Neural translation models nowadays typically use a beam search approach to sentence prediction, which constitutes an expansion of greedy search. The main idea behind beam search based predictions, is to not only select the most likely sentence at each autoregressive step, but rather a fixed number of them, thus leading to predictions more robust to early mistakes in the prediction process. The advantages of this method compared to greedy search can be visualize with a simple example. Consider a hypothetical input sentence leading to a first probability distribution in the prediction process that would be bimodal (for instance, with one word W_1 associated with probability .45, and W_2 with .44). The greedy search process would dictate to select the first word, associated with maximum probability, and to keep expanding from it. However, nothing guarantees that the sentence obtained following a greedy process after selecting W_2 as the first sentence word would end up yielding a lower joint probability distribution. As an example, if the second probability distribution obtained after injecting W_1 into the recurrent neural network were to yield a maximum value of .5 while the one obtained after injecting W_2 were to yield a maximum value of .9, the joint probabilities after selecting W_1 or W_2 would respectively be .225 and .396, strongly suggesting that sentences beginning with W_2 are better translations than those beginning with W_1. As a consequence, keeping several sentences in memory, and not only the most likely at all time, might lead to better overall predictions, in term of joint probability, which is in the end the value that the sentence search process is designed to maximize. Strictly speaking, a beam search based prediction from a neural sequence model can be obtained as follows:

1. Use the initial state and the <START> token to predict the target sentence's $k \in \mathbb{N}$ most likely words
2. Similar to greedy decoding, compute the second word's probability density, using each of the k likeliest first words, resulting in k distinct probability distributions

3. Compute all available 2-grams probability distributions
4. Retain the k most likely 2-grams candidates
5. Repeat steps 1 through 4 in an iterative fashion until all k sentence candidates contain a <STOP> token
6. Select the sentence candidates associated with highest probability

3.4.4 RNN based encoder-decoder architecture

3.4.4.1 Word embedding

One major difference between the sequential learning problem previously defined for ordinal regression and machine translation lies in the sequential variable's dimensionality. Indeed, although the former modelled sequences of binary variables, which are fairly straightforward to handle from both a statistical and computational perspective, the latter focuses of sequences of words variables, which are notoriously complex to handle and incorporate into traditional statistical models. Considering words as a categorical variable with as many states are there are words in the language's vocabulary, for instance, presents several drawbacks. First, natural language vocabularies are typically comprised several thousands, to several tens of thousands of words, thus leading to highly dimensional vector encoding. In addition, this type of approach does not translate well the actual nature of the data. Indeed, such an approach implies that all the categorical variable's states are perfectly independent, which is far from being the case with words, where typically a representation where the word "cat" is considered closer to the word "dog" than the word "supernova" under a predefined similarity measure might be desirable.

This problem is typically addressed in NLP by using learnable linear projections of one-hot encoded variables as inputs for neural network. This embedding typically ends up capturing meaningful representations of the concepts it represents, as can be seen in figure 3.4.14. These linear projections

are typically called “word embeddings” in the deep learning academic community, and can be derived as follows:

- Convert words into one-hot vectors:
    - Associate each word present in the vocabulary with one unique index, an integer between 0 and the vocabulary’s cardinal,
    - Associate each word present in the vocabulary with a vector of dimensionality the vocabulary’s cardinal, with every vector’s coordinate set to 0 except for the one corresponding to the word’s previously defined index.
- Feed these vectors to a one-layered perceptron, without any non-linearity applied to it (which essentially equates to a simple linear transformation)
- Fit the embedding jointly with the model in a purely supervised fashion. This practice is most efficient in supervised tasks but requires a significant amount of observations,
- In cases with limited sample sizes, pre-trained, out of the box word embeddings are readily available online for use

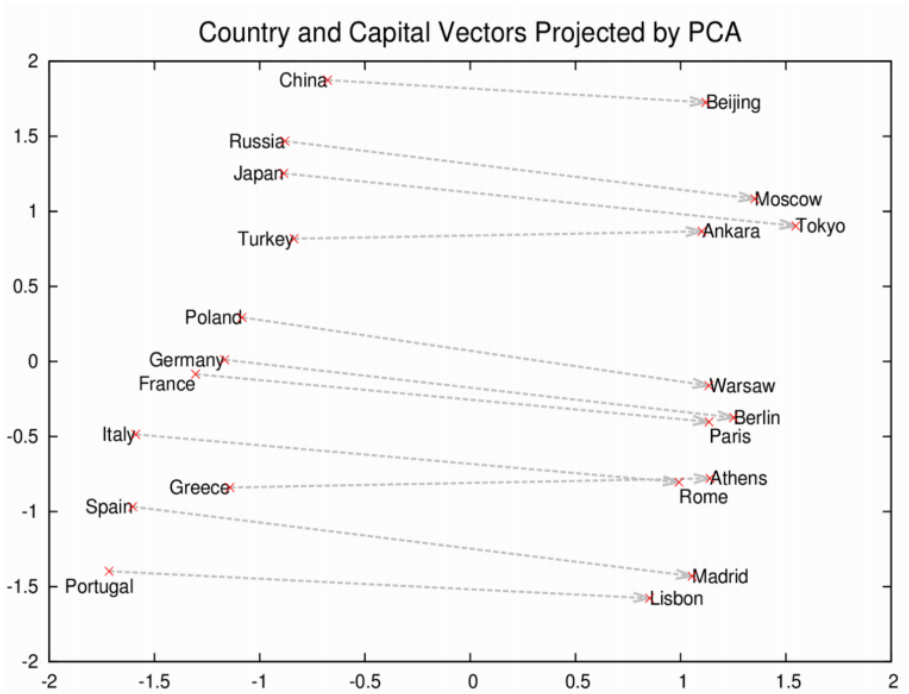


Fig. 3.4.14 Visualization of a word model used in natural language processing. The model here shows its ability to understand the relationship between countries and their capital cities. No information besides text data (typically Wikipedia) is given to the model during training

3.4.4.2 Model architecture

The ordinal regression method introduced before presented a fairly straightforward manner of modelling correlated sequential output variables, by using linear combinations of input variables to build a recurrent neural network's initial state, which was then used to build the output variable's probability distribution in an autoregressive fashion. As such, this approach constitutes a promising candidate for a machine translation statistical model. However, as aforementioned, transitioning from the simple, binary search based ordinal regression to a neural translation still present one major issue. Indeed, the ordinal method's inputs were typically assumed to be vectorial. However, machine translation algorithms take as inputs sentences which are variable length sequences of inputs. As a consequence, the sequential nature of the model's input variables will have to be taken into account when building the vector serving as an initial state to the recurrent neural network used to predict the

target sentence. One straightforward approach to do so can be found in using an additional recurrent neural network, in a regressive setting this time, in order to build the autoregressive neural network's initial state. This type of architecture is called an encoder-decoder architecture, where a first recurrent neural network is used to encode the input sentence into a fixed sized, vectorial representation, that another recurrent neural network then uses as an initial state to predict the sentence in the target language. These two recurrent neural networks are usually referred to as encoder and decoder, as the first reads through the source sentence in order to encode it into a quantitative, vectorial representation that the second recurrent neural network decodes into another language. This type of architecture is called an encoder-decoder architecture, and was the first historical neural network based predictive model able to learn to perform machine translation in an end-to-end fashion. This model's architecture, and its training methodology can be seen in figure 3.4.15 and summed up as follows:

1. Create word embeddings for both languages and two recurrent neural network (typically all sharing the same dimensionality),
2. Convert the input and target sentences into two vectorial sequences using their respective embeddings,
3. Sweep through the input sequence with the first recurrent neural network (its initial state can typically be set to 0 or considered as an additional set of model learnable parameters),
4. Use the first neural network's last output as an initial state for the second neural network,
5. Sweep through the target sequence (with a <START> token appended to its beginning) with the second recurrent neural network (initialized with the previously obtained initial state),
6. Fit a logistic multinomial regression on the second RNN's outputs to predict each element of the target sequence (including the <STOP> token appended to its end) using teacher forcing (as described in 3.4.1.1), updating all model parameters using gradient descent and backpropagation,

In modern approaches, the final multinomial logistic regression's parameter matrix is replaced by the transposed of the second language's word embedding matrix. The rationale behind this idea lies in the fact that the second recurrent neural network's output variables share the same essence as their input (e.g. they both represent words in the target language). Technically, the word embedding converts a density distribution across all possible words in the target language into a dense vectorial space, while the last multinomial logistic regression layer converts dense vectorial output into a density distribution across all possible words in the same language. As a consequence, at least heuristically, having these two linear transformations share their parameters could potentially improve model performance, which is verified empirically[49].

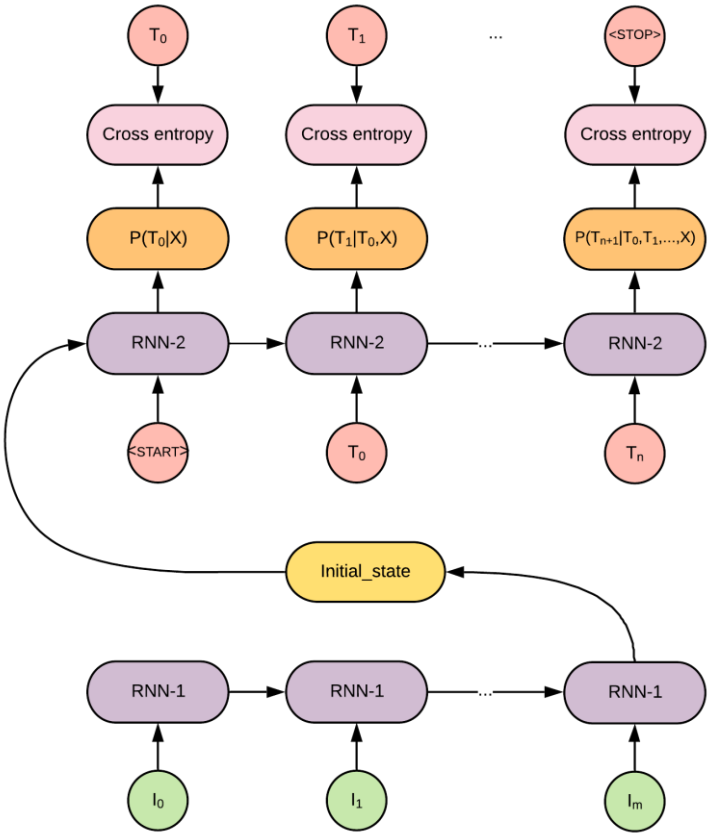


Fig. 3.4.15 RNN based encoder decoder architecture for neural machine translation. A first recurrent neural network sweeps through the input sentence's sequence of word embeddings. Its last output is used as the initial state for a second recurrent neural network that predicts the output sentence in an autoregressive fashion, in a similar approach to what was introduced in binary search based ordinal regression.

The encoder-decoder architecture was the first approach to machine translation that offered an entirely end-to-end, fully differentiable solution. However, when introduced, this approach to machine translation was still outperformed by the best expert systems, until progressive improvement made it the historical state of the art in the field. The most notable concept introduced in RNN based machine translation being the attention mechanism.

3.4.5 Attention mechanism

Attention mechanisms were one of the most impactful innovation in the field of neural machine translation since the advent of the encoder-decoder recurrent architecture. As aforementioned, although the latter constitutes the first fully differentiable, end-to-end trainable neural network able to output variable length sentences conditioned on a source sentence, they were not able to outperform the best expert system available at the time of their creation. Attention mechanisms were created in order to improve upon this concept, by trying to answer to major issues inherent to the encoder-decoder architecture as presented above: First, the encoder-decoder architecture relies on its ability to encode any given sentence in a fixed size vector (the initial state first fed to the decoder to start the autoregressive prediction process). However, a sentence's length, and amount of information (in the informal sense) can significantly vary. Fine-tuning the initial state's dimensionality thus becomes an ill-defined task. Too low a value would yield to poor behaviour on longer sentences, and too high would result in potential overfitting, and inefficiency for shorter sentences. Finally, using the same context vector to predict every word can feel a bit cumbersome as well. Indeed, cases where translating a given word from a sentence requires to know the entire sentence should be relatively rare, at least when it comes to technical translation (as opposed to philosophical or poetic translation, for instance). An example of this idea can be found in figure 3.4.16, showing a sentence in English and

its translation in French. Were it not for the French conjugation, the sentence could almost be translated word for word.

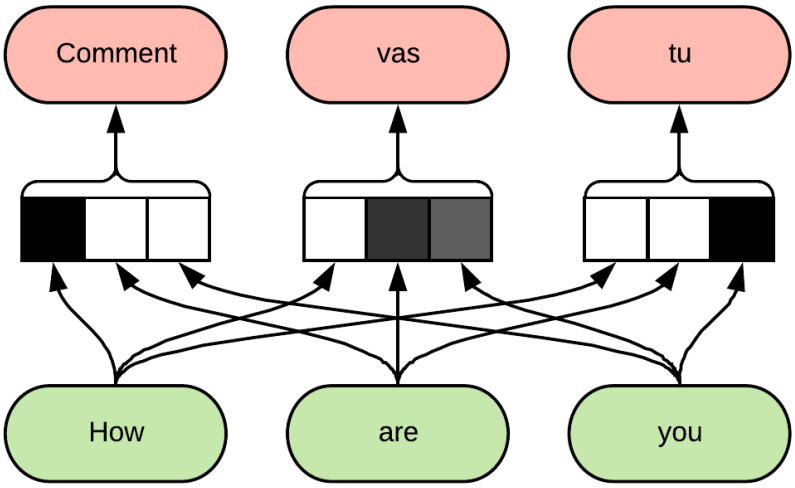


Fig. 3.4.16 Example of English to French translation, and the importance of each word in the source sentence to obtain the translation (in increasing order from white to black). Not all words in the source sentence matter to translate the input sentence. The second word, however, requires information from the input sentence's two last words, in order to choose the proper conjugation

The concept of attention mechanism was introduced by Bahdanau et al. in 2015 to address both these problems by introducing one fundamental idea, that instead of building only one context vector (the decoder's initial state) to predict the translated sentence, specific context vectors should be derived for each word predicted in the target sentence. This idea might appear quite counterintuitive, in regard of the encoder decoder architecture presented so far. Indeed, the entire connexion between encoder and decoder was based on a single context vector, the encoder's last output, that was used as an initial state for the decoder. The idea of attention thus raises two questions:

- How to build these distinct context vectors?
- How to incorporate them in the decoder?

In order to create these new context vector, the encoder-decoder architecture can offer some insights. Indeed, the encoder's last output was used as an overall context vector because, having swept through the entire source sequence, it encapsulated a representation of all the required information present in the sentence to perform the translation task. However, a recurrent neural network not only gives one unique vector, but one for each of the sequence element it goes through, as can be seen in figure 3.4.17. In the context of neural translation, the encoder's output vector can be interpreted as a quantitative representation of the last word the encoder just received, contextualised on every word before it in the sentence.

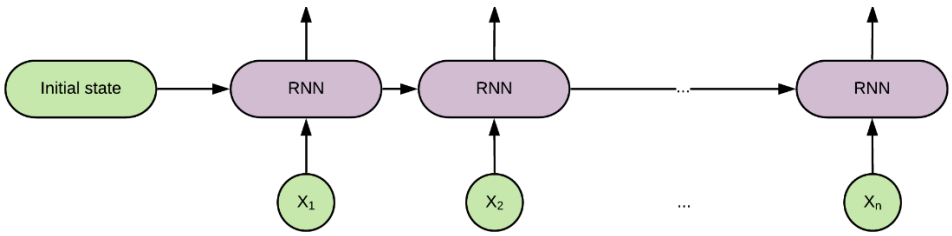


Fig. 3.4.17 A recurrent neural network outputs a vector for each of the input sequence element it receives

These vectorial outputs, when considered in parallel to the idea of word importance expressed in figure 3.4.16, constitute a promising set of candidates as building blocks for the specific context vectors at hand in attention models. One could for instance define the context vectors as weighted means of the encoder's outputs, which is exactly what attention models do. The question of obtaining these weights, however, still remain unanswered. A desirable property for these weights would be to depend on both the input word on which they are applied, and the word in the target sentence they are supposed to contextualise. This can typically be done by injecting both the decoder's last state and the encoder's

output into a one-layer perceptron in a regressive setting, as can be seen in figure 3.4.18. This perceptron's parameters can be fit through gradient descent and backpropagation alongside the encoder and decoder's parameters. Formally, the context vector for target word $i \in \mathbb{N}$ as:

$$C_i = \sum_{j=0}^{m} \alpha_{ij} H_j \quad (3.4.7),$$

with:

- $C_i \in \mathbb{R}^d, i \in [\![0, n]\!]$ the context vector for the target word $i \in \mathbb{N}$ and $d \in \mathbb{N}$ the decoder's hidden size
- $\alpha_{ij} \in \mathbb{R},\ i, j \in [\![0, n]\!] \times [\![0, m]\!]$ the attention weight to be attributed to the output of the encoder at source word j in order to build the context vector for decoding target word i
- $H_j \in \mathbb{R}^d, j \in [\![0, m]\!]$ the encoder's output from source word j

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=0}^{m} \exp(e_{ik})} \quad (3.4.8),$$

where :

$$e_{ij} = a(S_{i-1}, H_j) \quad (3.4.9),$$

with:

- $e_{ij} \in \mathbb{R}, i, j \in [\![0, n]\!] \times [\![0, m]\!]$
- a a parametric function taking as input the decoder's state at time step $i - 1$ and the encoder's output at time step j. a is typically chosen as a simple one-layer perceptron
- $S_{i-1} \in \mathbb{R}^d$ the decoder's state at time step $i - 1$
- $H_j \in \mathbb{R}^p$ the encoder's output at time step j with $p \in \mathbb{N}$ the decoder's hidden size

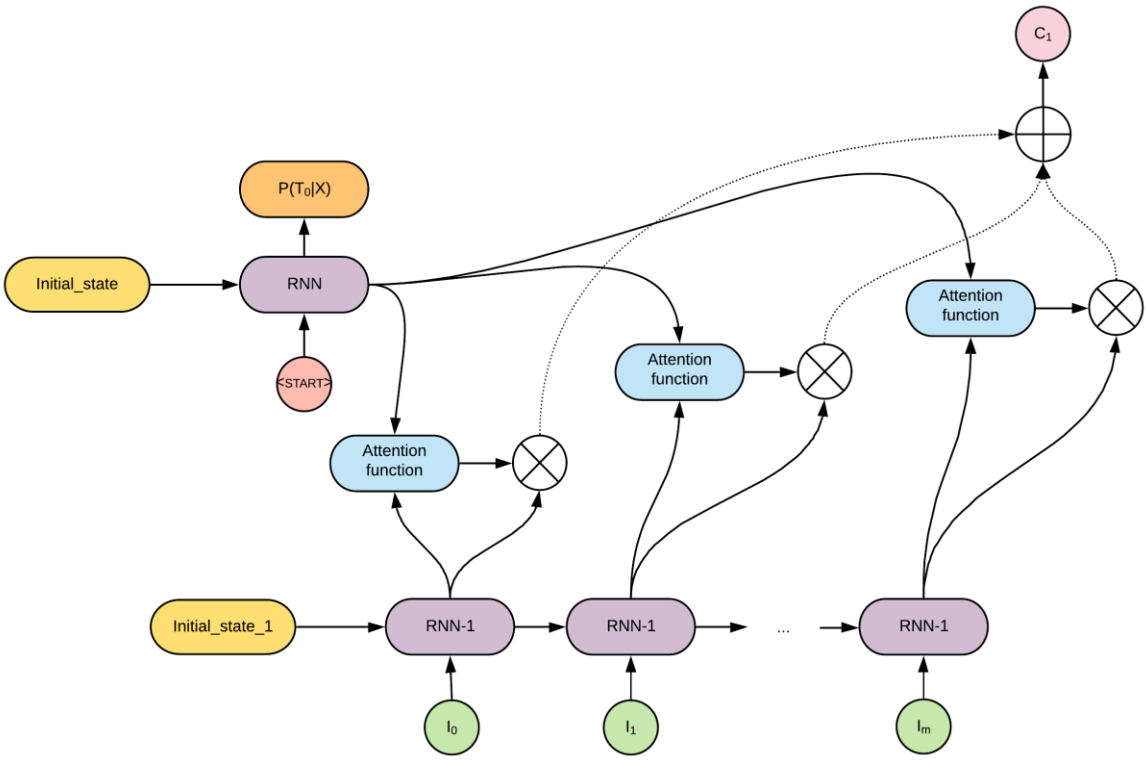


Fig. 3.4.18: Definition of the target sentence's second word's attention vector. The decoder's previous state is combined with every encoder outputs in a perceptron to compute attention weights, which are then normalized using a softmax function. These weights are then used as coefficients for a linear combination of all the encoder's outputs, thus creating a specific context vector for a given word in the target sentence

Finally, now that a way of creating these individual context vectors, remains the question of actually injecting them into the recurrent neural network that decodes the sentence. Several ways of doing so have been proposed over the year, as for instance by adding the context vector to the decoder's last recurrent state. This is directly possible if the encoder and decoder have the same dimensionality, and can be adapted in case they differ by simply injecting the context vector into a one-layer perceptron prior to vector addition. To sum up, an encoder decoder architecture with attention mechanism can be used to perform neural machine translation as displayed in figure 3.4.19 by following the following steps:

- Have an encoder recurrent neural network sweep through the source sentence's sequence of word embeddings,
- Start the autoregressive process with an unconnected initial state for the decoder (which can be either fixed to 0 or considered as additional model parameters), use this state and the encoder's representations to compute attention weights and the first context vector
- Add the context vector to the initial state, and use it as state input to the decoder recurrent neural network to estimate the target sentence's first word's probability distribution
- Repeat the process until the token <STOP> is reached

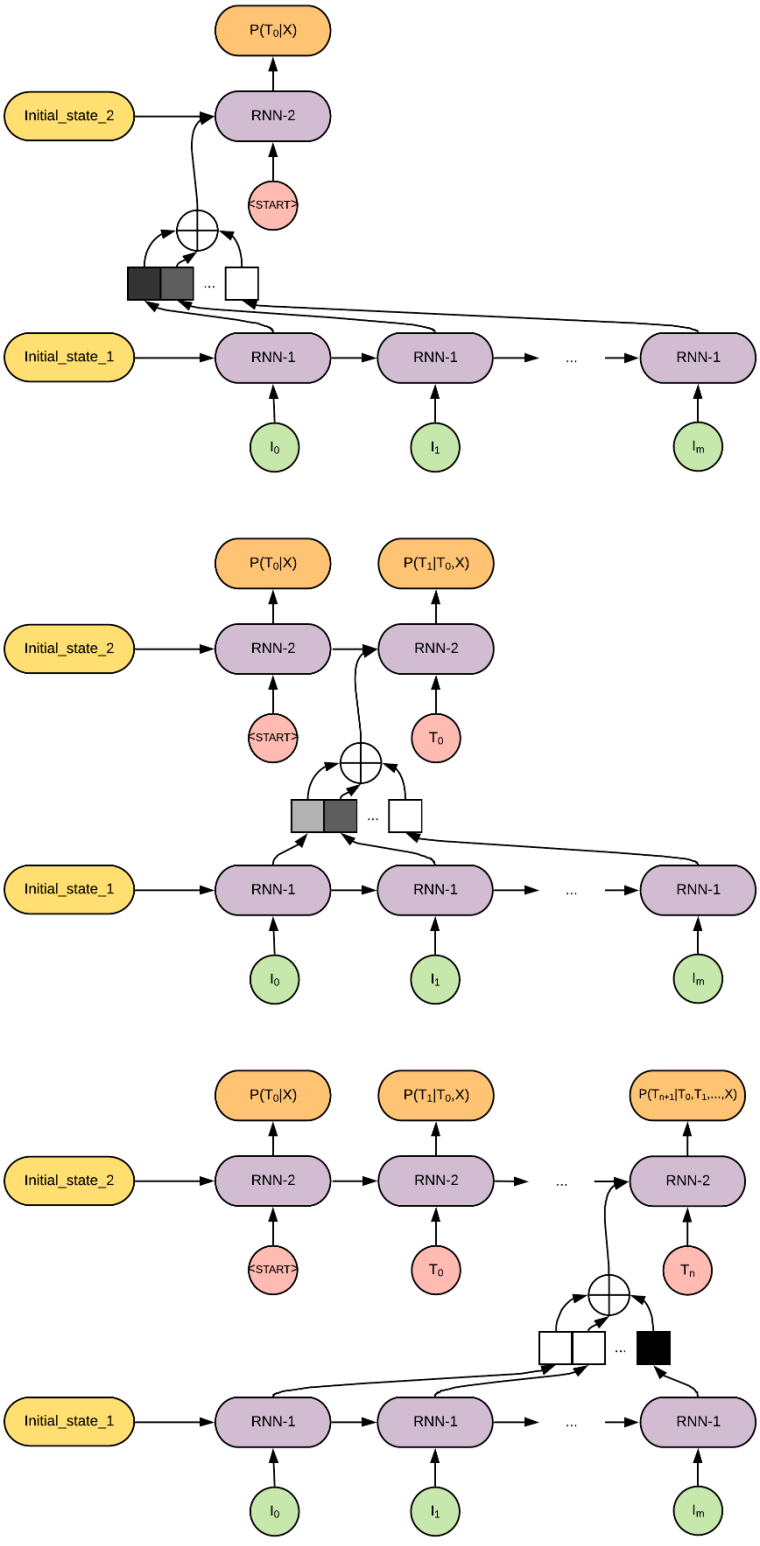


Fig. 3.4.19 Encoder-decoder architecture with attention mechanism. At each autoregressive step, the RNN's previous state is combined with a context vector built as a linear combination of each of the encoder's outputs whose coefficient are learnt from the data

One interesting property of attention mechanisms lies in the fact that they allow for some interpretability of the overall model, which is still a rare thing in modern deep learning based architectures. Indeed, by collecting the attention weights derived to translate a given sentence, one can observe which source words had more attention weights in order to predict a given word in the target sentence and by extension which source words had an impact in the decision process of predicting every word in the target sentence, as can be seen in figure 3.4.20.

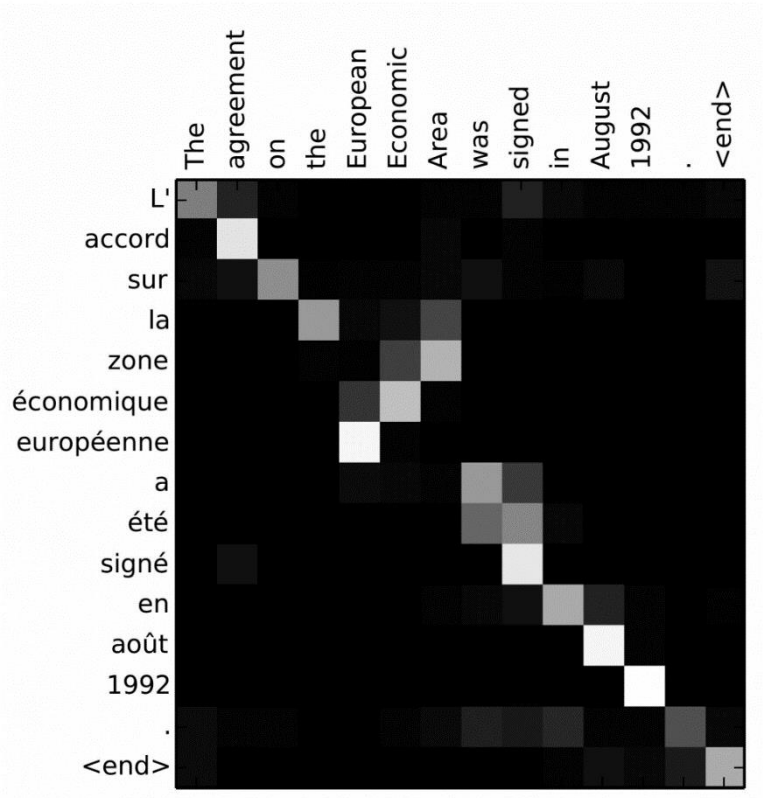


Fig. 3.4.20 Collecting the attention vectors for each of the target sentence's words allows for some interpretability of the encoder-decoder architecture[50]. Notice how the attention matrix is mainly diagonal, except for an inverted submatrix denoting how the English and French languages invert word order for "European economic area", that translates to "Zone économique éuropeenne"

Finally, the architecture mechanism presented in this section is only one among many other introduced over the years. There are potentially countless manners of creating these context vectors and

incorporating them into the decoder's state. The additive attention introduced here was chosen for its historical significance (being the first attention mechanism ever proposed) as well as for its interpretability.

3.4.6 Transformer architecture

The incorporation of attention mechanisms in encoder decoder architectures led to the rise of neural networks as the state of the art in the field of machine translation. However, these encoder-decoder architectures still suffered from some significant issues cause by their recurrent neural network base:

- Even with attention mechanisms, they tend to suffer from the vanishing gradient problem arising with longer sentences
- They are notoriously slow to train and infer with, due to the RNN's poor parallelism

In fact, it was shown as early as 2016 that recurrent neural network were typically outperformed by carefully chosen convolutional architectures (such as causal dilated convolutional networks, for instance) on most Natural Language Processing tasks[51], which led to their downfall in the field. This empirical observation was verified as well in machine translation, where recurrent neural networks in the encoder and decoder architecture can simply be replaced by convolutional networks to yield better and more efficient neural translation models. However, this approach was quickly outperformed by an idea that, at least at first sight, might appear surprising. The idea of self-attention, and more broadly speaking purely attentional models. Looking back at the attention mechanism introduced above, its main purpose was to build for each word predicted in the target sentence a specific vectorial representation of the source sentence's context, derived from weighted means of each of the encoder recurrent neural network's outputs. These outputs were themselves considered as representations of the source sentence's word, conditioned on each of the sentence's past words. From this line of thinking, a question arises. Why not use attention to build representations of the source sentence's

words, contextualized on itself? This idea, called self-attention, led to the introduction of the Transformer architecture, which is the current state of the art in neural machine translation, and proposes to replace recurrent neural networks in both the encoder and decoder by self-attention mechanisms, (while of course retaining the encoder-to-decoder attention).

3.4.6.1 Overview

The transformer architecture shares some fundamental similarities with its recurrent neural network based counterpart. They both make use of word embeddings, for both the source and target language. They are both composed of an encoder that learns a representation of the source sentence, and a decoder that learns to predict the target sentence using the encoder's outputs through teacher forcing, and make predictions through beam search. The word level probability distributions the decoder outputs are both obtained by using the word embeddings' transpose as an invert transformation whose output are injected into a softmax function. To sum up, they both follow the exact same approach to learn to find good translation candidate of a source sentence in a given target language. They mainly differ in the functional form that take both the encoder and the decoder. Where they were defined as simple, recurrent neural network so far, their definition in the transformer is a bit more complex, and constituted of stacks of several elements, as can be seen on figure 3.4.21, that can be summed up as follows:

- Positional encodings added to both the source and target sentence's word embeddings prior to their injection into the encoder and decoder, respectively
- A form of attention called Multi-head dot product attention used in a self-attention setting for both the encoder and decoder (with a masking option for the decoder, that will be explained further below)
- The same attention mechanism used in a traditional encoder to decoder attention setting in the decoder

- An element-wise feedforward network for both the encoder and decoder
- Residual connections, layer normalization and drop-out elements in order to improve training and reduce overfitting

These stacks are cumulated sequentially, in a similar fashion to layers in a traditional feedforward neural network, to further expand the model's predictive power. The number of stacks per encoder and decoder can be considered a model hyper-parameter, but is typically kept to the default value of 6 (as recommended in the model's seminal paper).

The transformer architecture has as a peculiarity to keep its representation dimension constants (meaning the dimensionality of each of the network's part remain constant), from word embeddings to encoder and decoder outputs. It also relies on having either one shared embedding for both the source and target language (to make use of potential similarities between languages, such as can be observed in French to English translation, for instance), or requires the distinct word embeddings to have same dimensionality. This constant dimensionality value is usually denoted as d_{model}, the model's dimensionality, and can be considered as one of the model's major hyper-parameter (with typical values ranging from 64 to 1024 in Natural Language Processing settings).

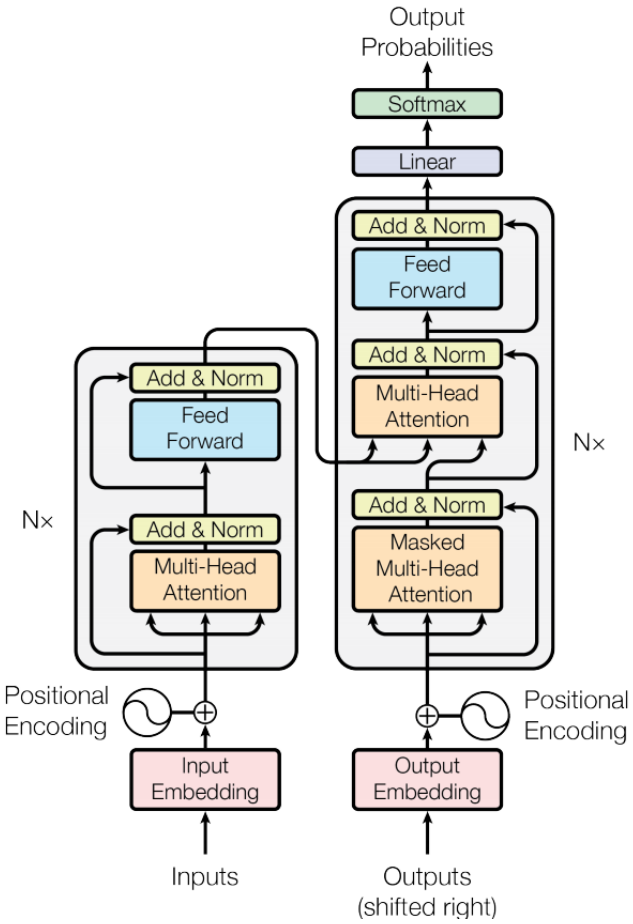


Fig. 3.4.21 Overall Transformer architecture[52]. The encoder and decoder are only built upon attention and element-wise perceptron, no recurrent or convolutional neural networks are used

3.4.6.2 Positional encoding

Although the deletion of recurrent neural networks within the encoder-decoder architecture comes with all the aforementioned advantages, it leaves one significant drawback that requires to be addressed. Indeed, the attention mechanism, as is, is perfectly invariant to word placements in the sentence. This phenomenon becomes clear when remarking, in the recurrent neural network version of the encoder-decoder architecture, that the function computing the attention weight only looks at one word in the source and target sentence at a time. This did not constitute any problem, as the

recurrent neural network could implicitly account for the word's position in its representation. In a fully attentional architecture, however, this information will need to be explicitly expressed, which is done through addition of a predefined position encoding vector to each of the sentences' word embeddings (for both the source and target sentences). These position encoding vectors can be visualized in figure 3.4.22 and were defined arbitrarily as:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \quad (3.4.10),$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \quad (3.4.11),$$

with:

- $d_{model} \in \mathbb{N}$ the model's dimensionality
- $PE \in \mathbb{R}^{d_{model}}$ position embedding vector's value for the sentence's word at position pos
- $pos \in \mathbb{N}$ the given word's position in the sentence
- $i \in \mathbb{N}$ an index denoting the positional embedding vector's coordinate

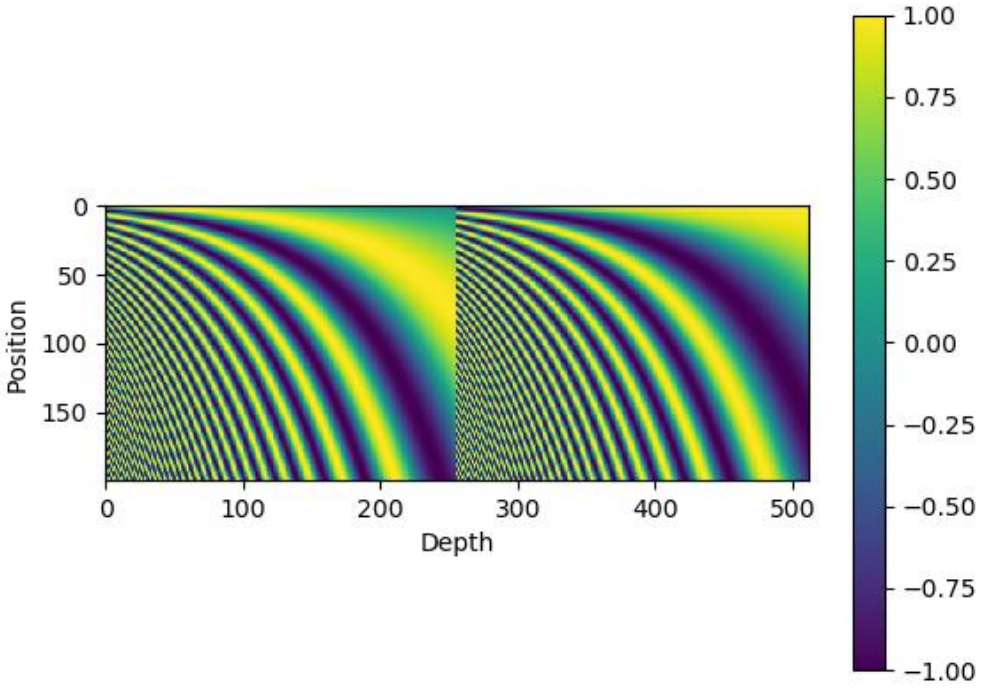


Fig. 3.4.22: Visualization of the transformer's positional encoding for an input sentence of length 200 and a model's hidden size of 512

3.4.6.3 Scaled dot product attention

The attention mechanism used in the Transformer, called scaled dot product attention differs slightly from the additive concept presented before, mostly in the way attention weights are derived. First, for a given target word to be predicted at time step t, each attention weight was computed as a function of the encoder's corresponding output and the decoder's state at time step $t-1$. This cannot apply here, as recurrent neural network are absent from the decoder. This vector can however simply be replaced by the vectorial representation of the target word at time step $t-1$. Secondly, the attention function was defined as a one-layer perceptron that took as input the encoder's output and the previous state. Scaled dot product attention replaces that one-layer perceptron by a much simpler function, being a dot product. This however constrains the encoder's output and the decoder's target words to share their dimensionality (which, as was mentioned previously, is always the case in the transformer architecture). Finally, going back to the original definition of the attention mechanism, the main rationale was to build weighted means of the encoder's outputs, whose weights would depend on these exact same vectorial representations of the source sentence. However, there is absolutely no guarantee that representations fit to yield good context vectors from a weighted sum can also be used as input to the attention function and provide good attention weights, and reciprocally. In other words, the vectorial representations of the source sentence derived by the encoder might, in a sense, be "overloaded". Consequently, building two distinct representations of the input sequence, one specialized in building attention weights, and the other dedicated to building context vectors, might be desirable to ensure this "representation overloading" phenomenon does not impair the attention mechanism's performance. This is exactly the idea behind the Transformer's attention mechanism, which takes as inputs three distinct sequences called queries, keys and values, of which keys and values share the same sequence length (thus typically being distinct representations of the same sequence) and can be written as follows (in matrix notation)

$$Attention(Q, K, V) = softmax(\frac{Q\ K^T}{\sqrt{d_k}})\ V \quad (3.4.12),$$

with:

- $Q \in \mathbb{R}^{n \times d_k}$ with $n, m \in \mathbb{N}$ the queries sequence
- $K \in \mathbb{R}^{m \times d_k}$ with $n, m \in \mathbb{N}$ the keys sequence
- $V \in \mathbb{R}^{m \times d_v}$ with $n, m \in \mathbb{N}$ the values sequence
- $n \in \mathbb{N}$ the length of the queries sequence
- $m \in \mathbb{N}$ the length of the keys and values sequences
- $d_k \in \mathbb{N}$ the dimensionality of the queries and keys sequences' elements, used as a normalization constant within the dot product attention mechanisms as a heuristic to improve gradient behavior
- $d_v \in \mathbb{N}$ the dimensionality of the values sequences' elements

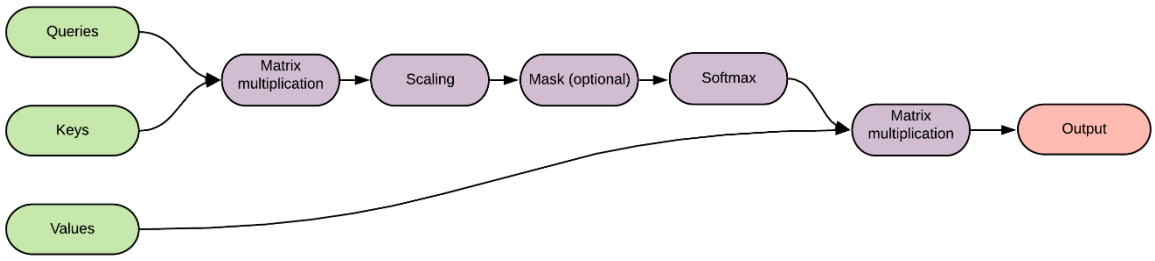


Fig. 3.4.23 visualization of the scaled dot product attention mechanism

The reader might notice that this attention mechanism's definition introduces three distinct sequences (two sharing their dimensionality and two sharing their sequence length), which might at first appear surprising. Indeed, the attention mechanism defined so far focused on building one context vector from the encoder sequence. One of these additional sequences obviously comes from the fact that

this attention mechanism is defined from matrix multiplications (instead of dot product in traditional multiplicative attention mechanisms) to allow for all attention vectors to be derived in parallel during teacher forcing based training. This however leaves an additional sequence that requires some explaining. Going back to the original definition of the attention mechanism, the main rationale was to build weighted means of the encoder's outputs, whose weights would depend on these exact same vectorial representations of the source sentence. The idea of using three distinct sequences comes from noticing that the encoder's outputs have this dual purpose that might overload them. This is exactly why these three sequences are introduced in the matrix formulation of this scaled dot product attention mechanism, which is why it allows the constraint of having the sequences Q and V of same length. These will typically be taken as distinct representations of the same sequence, as will be seen in the following part.

3.4.6.4 Multi-head scaled dot product attention

For computational efficiency, the transformer doesn't make use of the dot product directly as is. Instead, each of the attention element in the model are composed of several distinct attention mechanisms that are all ran in parallel, the results being concatenated afterwards. This idea is called multi-head attention and follows the following steps:

1. Each of the three sequences is injected element wise into h element-wise learnable linear transformation ($h \in \mathbb{N}$ is considered a model hyperparameter, and is typically set to 8)
2. The three sequences' projections are all injected into a dot-product attention mechanism in parallel
3. The resulting sequences' elements are all concatenated in an element-wise fashion (the elements are concatenated alongside the vector's depth, not as a longer sequence of vectors)
4. The resulting sequence is then injected into a final linear transformation

Formally, for three vectorial sequences $(Q, K, V) \in \mathbb{R}^{n \times d_{model}} \times \mathbb{R}^{m \times d_{model}} \times \mathbb{R}^{m \times d_{model}}$ with $n, m \in \mathbb{N}$ the length of sequences Q and K, V respectively, the multi-head scaled dot product attention mechanism can be written as follows:

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O \quad (3.4.13),$$
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (3.4.14),$$

with :

- $W^O \in \mathbb{R}^{hd_v \times d_{model}}$ a linear transformation that brings the concatenated output sequence back to the model's dimensionality
- $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $i \in [\![1, h]\!]$ the queries' linear transformations
- $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $i \in [\![1, h]\!]$ the keys' linear transformations
- $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, $i \in [\![1, h]\!]$ the values' linear transformations

The values of h, d_v and d_k can be considered as model hyper-parameter as well, but are typically chosen so that $h = 8$ and $d_v = \ d_k = \frac{d_{model}}{h}$.

Now that the transformer's attention mechanism has been fully defined, remains the task of applying it to the context of neural translation:

- For encoder to decoder attention, the keys are chosen as the previous decoder layer's output and queries and values as the encoder's output sequence, as can be seen at the top of figure 3.4.24
- For self-attention, all sequences are chosen as the block's previous layer's output, as can be seen at the bottom of figure 3.4.24
- Note that although some of the three input sequences are chosen as identical, the linear transformations applied to each of these versions of the same sequence are not.

Consequently, the three sequences injected into the actual attention mechanisms have no reasons to be identical

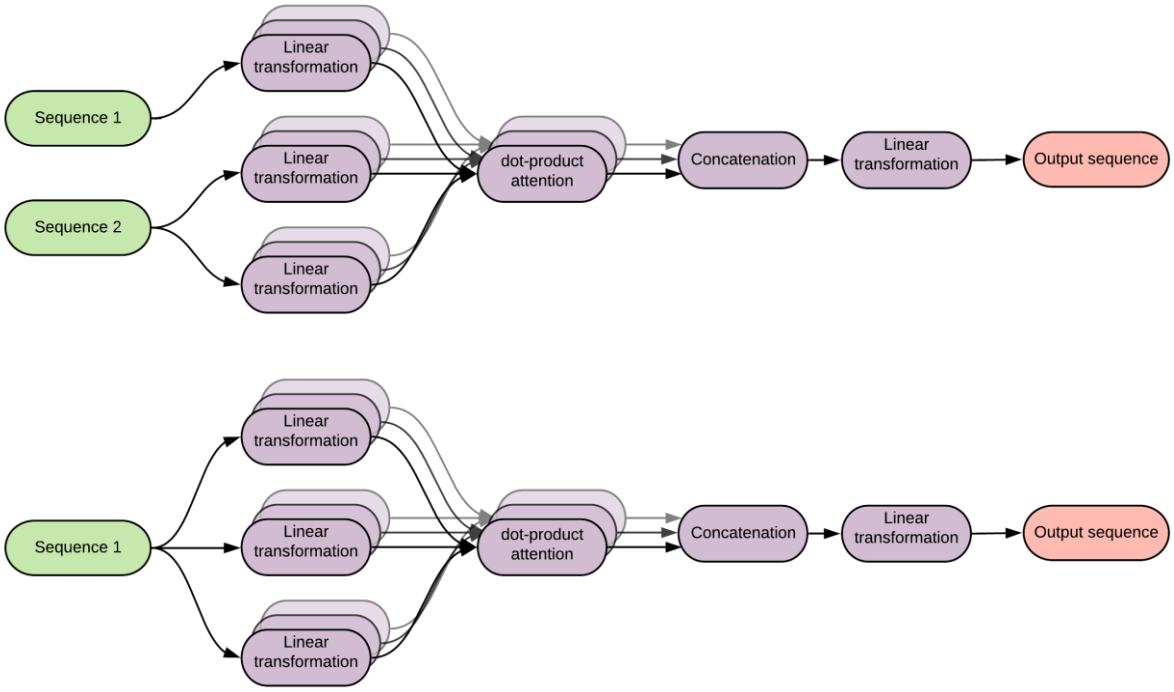


Fig. 3.4.24: Top: Multi-head attention in an encoder-to-decoder setting, where the queries and values in the attention mechanisms are linear transformations of the encoder's output and the attention mechanisms' keys are linear transformations of the decoder's previous layer's output. Bottom: Multi-head attention in a self-attention setting, where all the input sequences in the attention mechanisms are linear transformations of the same sequence, the attention block's input

A problem immediately arises with the decoder's self-attention mechanism, at least in a training setting. Indeed, in teacher forcing setting, the entire target sentence is usually fed to the model in order to speed up the training process. This did not cause any issue when the decoder was a recurrent neural network, as they prevent any conditioning on any word located after the one currently injected into the model. Attention mechanisms, however, can condition any token in their input sequence with any other present in the sequence. As a consequence, at a given time-step during training, the decoder has for input variable the exact target variable it is supposed to predict, as can be seen in figure 3.4.25.

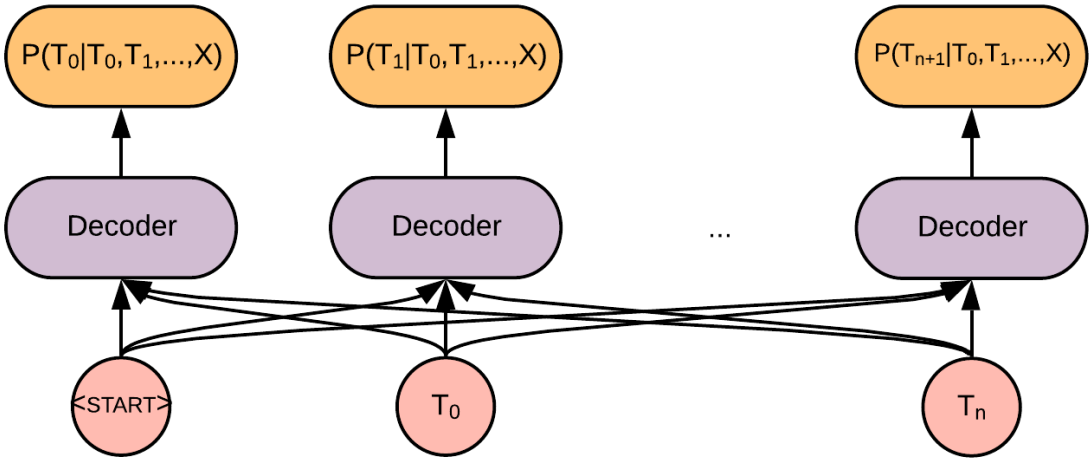


Fig. 3.4.25 Visualization of the problem of decoder self-attention during the training process (with teacher forcing). Even without any encoder information, the decoder can perfectly predict every word in the target sentence, as all words are now considered as input variables (instead of only previous words for a given time step in recurrent neural networks)

In order to counteract this undesirable phenomenon a simple mask process similar to what was done in the ordinal regression example is applied to the attention weights obtained in the decoder's self-attention block:

- The derived attention weights corresponding to future words are set to 0,
- The entire attention vector is renormalized prior to the attention final matrix multiplication.

3.4.6.5 Element-wise multilayer perceptron

Finally, in order to increase the model's depth and non-linear explanatory power, each of the encoder's stack ends with an element-wise two-layer perceptron, element-wise meaning the perceptron takes as input only one vectorial element of the input sequence, and is applied to all of them simultaneously. Such a neural network can be interpreted as a kind of convolutional neural network whose kernel size would be set to 1, and is defined as follows:

$$FFN(x) = \max(0, xW_1 + b_1)\, W_2 + b_2 \quad (3.4.15),$$

with:

- $x \in \mathbb{R}^{d_{model}}$ one element of the input sequence
- $W_1 \in \mathbb{R}^{d_{model} \times d_{ff}}$ the layer's first weight matrix, with $d_{ff} \in \mathbb{N}$ it's dimensionality
- $b_1 \in \mathbb{R}^{d_{ff}}$ the layer's first bias vector
- $W_2 \in \mathbb{R}^{d_{ff} \times d_{model}}$ the layer's second weight matrix
- $b_1 \in \mathbb{R}^{d_{model}}$ the layer's second bias vector
-

3.4.6.6 Overall architecture

Now that all the Transformer architecture's basic blocks have been formally described, the definition of both the encoder and decoder is quite straightforward. The encoder's output is obtained applying the following steps:

1. Create the vectorial source sequence using word embeddings,
2. Add the positional encoding to the vectorial source sentence,
3. Inject the resulting sequence into a multi-head dot product self-attention layer with a residual connection,
4. Normalize its output following the layer normalization methodology,
5. Inject the resulting sequence into an element-wise multilayer perceptron with a residual connection,
6. Normalize its output following the layer normalization methodology,
7. Repeat steps 3 to 6 N times

Once the encoder's outputs have been obtained, model prediction (either while training or predicting) can be derived as follows:

1. Create the vectorial target sequence using word embeddings (the target sequence being either just the <START> token when predicting or the actual target sequence with <START> and <STOP> tokens appended at its beginning and end respectively when training with teacher forcing),
2. Add the positional encoding to the vectorial target sentence,
3. Inject the resulting sequence into a masked multi-head dot product self-attention layer with a residual connection,
4. Normalize its output following the layer normalization methodology,
5. Inject the resulting sequence and the encoder's final output into a multi-head dot product attention layer with a residual connection,
6. Normalize its output following the layer normalization methodology,
7. Inject the resulting sequence into an element-wise multilayer perceptron with a residual connection,
8. Normalize its output following the layer normalization methodology,
9. Repeat steps 3 through 8 N times.

3.5 Conclusion

In this first section, we introduced the concept of artificial neural network, starting from its most primitive form, the multilayer perceptron, which shares many similarities with generalized linear models. We showed how to adapt this simple, yet fairly anecdotal family of models to perform analysis on non-structured data, using for instance recurrent or convolutional neural networks. In particular, we demonstrated that these models can perfectly handle the modelling of variable sequence length target variables, for instance using an encoder-decoder based architecture using either recurrent

neural networks or attention mechanisms. Although this technique is primarily used in machine translation, we showed that its potential in other problematics is quite promising as well, as it can be applied to any modelling problem that can be decomposed in a series of simpler decision, such as was presented in the ordinal regression method introduced in this thesis.

Now that we defined a spectrum of deep artificial neural network based predictive models, and showcased how they can be applied to a wide variety of problems, a question remains: can these methods be applied in the context of electronic health database analysis?

4 Deep neural network for epidemiology and their applications in cause of death statistics

4.1 Introduction

The availability of up-to-date, reliable data on causes of death is a matter of significant importance in public health related disciplines. As an example, the monitoring of leading causes of deaths is an important tool for public health practitioner and has a considerable impact on health policy related decision making processes[53–58]. The collection of said data, however, is a complex, time consuming process that usually involves the coordination of many different actors, starting from medical practitioners writing death certificates, to the final statistics' diffusion by public institutions. One example of non-trivial task involved in this process is the identification of the underlying cause of death from the chain of event reported by the medical practitioner in the death certificate[59]. According to the International Statistical Classification of Diseases and Related Health Problems, the underlying cause of death is defined as "(a) the disease or injury which initiated the train of morbid events leading directly to death, or (b) the circumstances of the accident or violence which produced the fatal injury"

[60]. Since the underlying cause of death is adopted for tabulation of mortality statistics, extracting it from death certificates is of paramount importance. The process of identifying the underlying cause of death from a death certificate can be summed up into two major steps, an example of which can be seen in table 4.1.1:

- Converting the causal chain of death written by the medical practitioner that observed the subject's passing from natural language to a medical classification, the International Statistical Classification of Diseases and Related Health Problems (ICD-10 classification),
- Identifying from the causal chain of death, expressed as ICD-10 entities, the subject's underlying cause of death.

	Textual causal chain of death	ICD-10 formatting	UC
Line 1	Cancer indifférencié de la glande thyroide ayant entraîné une compression complète locale (sténose oesophagienne et paralysie bilatérale des cordes vocales)	C73 K222 J380	C73
Line 2	NA		
Line 3	NA		
Line 4	NA		
Part 2	Diabète insulino-requérant, HTA, Dénutrition, Antécédent d'AVC, AOMI	E119 I10 E46 I696 I702	

Table 4.1.1: Example of the underlying cause of death extraction process from a death certificate. First the causal chain of death is converted into sequences of ICD-10 entities, on which are applied of set of WHO defined rules to derive the underlying cause of death used in official mortality statistics

Fast production of these statistics is of paramount importance for their use in epidemiology and public health. Namely because using them as a tool for public health decision making requires fairly up to date data. At the very least, producing one year of mortality statistics should take at most a year, anything slower result on cumulating delays that would in long term make them unusable.

As a consequence, a number of tools have been derived over time to accelerate the production process of these statistics, mostly based on expert system or classical natural language processing approaches.

Unfortunately, these systems fail to handle a significant amount of complex death scenarios, in which human evaluation is required, leading to a time consuming coding process. However, as aforementioned, deep artificial neural network based models are known to outperform significantly expert systems in a variety of settings, including natural language processing. As a consequence, investigating the application of deep learning based predictive models on both predefined tasks necessary to the derive mortality statistics might yield some interesting, improved tool to accelerate and help coders with the production process.

4.2 Neural translation and automated recognition of ICD-10 medical entities from natural language

As aforementioned, death certificates are filled out by the medical practitioner that observes the subject's passing. When it comes to the production of mortality statistics, the most important piece of information present on certificates is the chain of events leading to the subject's passing, written in natural language. Although natural language is extensively present in some health related databases, it is notoriously difficult to handle with traditional statistical methods, and prevents most international comparisons due to language barrier. In order to counter these undesirable properties, several approaches have been devised. For instance, encapsulating most medical entities in a standardized hierarchical tree structure, the ICD-10 classification[61] offers a powerful and expressive way of organizing "analytics friendly" health databases. On the other hand, ICD-10 entities are significantly less intuitive for human users than natural language, and identifying ICD-10 entities from natural language fluently requires years of training and practice. As a consequence, the data production of classification based medical data is usually handmade, expansive and time consuming. Several attempts have been made to design artificial intelligence based systems able to automatically derive medical entities from natural languages, some with quite promising performance[62–64]. However, all of

them fall short in automating the complex production schemes inherent to medical databases, specifically in regard to their high data quality standards.

However, recent innovation in deep artificial neural networks have achieved significant progress in natural language processing[65,66]. In particular, their applications in the field of machine translation[19,41,42], fuelled by increases in both data and computing power, repeatedly bring automated systems closer and closer to human level performances. Several attempts have been made to apply these powerful techniques in an electronic health database setting, most of them with mitigated success. As an example, the current state of the art in ICD-10 entity recognition from natural language in death certificates still remains a combination of expert system and SVM based classical machine learning[62]. Several explanations exist for this discrepancy between traditional machine translation and medical entity recognition. First, deep artificial neural network based methods are known to require huge amount of data for optimal performances. However, most experiment were either performed with slightly out-of-date neural architectures, or with dataset sizes at least an order of magnitude under what would be typically required[67]. On the other hand, the "Centre for Epidemiology on Medical Causes of Death" (CépiDc) has been storing French death certificates at the national scale since 2011 in both natural language and ICD-10 converted format. The entire database amounts to just under 3 million death examples, thus providing with sensibly better settings to investigate the potential applications of deep neural networks in medical entities recognition.

The following chapter formulates the process of ICD-10 entity recognition from natural language as a sequence to sequence statistical modelling problem (better known as seq2seq models in the academic literature) and proposes to solve it with a variation one of the state of the art machine translation neural architecture, the Transformer. The following section focuses on describing the aforementioned statistical modelling problem and overall methodology. Section 4.2.2 reports the result of experiments done on the French CépiDc dataset as well as a comparison with the current state of the art. Section

4.2.3 presents a discussion on the model's potential limitation through an error analysis and describes potential leads for improvement.

4.2.1 Material and methods

4.2.1.1 Related work

The task of identifying ICD-10 medical entities from natural language, whether in French or in any other language, is a well investigated problem, where several promising approaches have already been proposed. Most of this solution were published at the CLEF (Conference and Labs of the Evaluation Forums) eHealth[62,63,67], a competition held annually where teams compete to solve natural language processing tasks on medical textual data. For instance, the task of recognizing ICD-10 entities from death certificates (in several languages including French) have been addressed several times over the years in this competition. So far, when it comes to the task of extracting ICD-10 entities from French death certificates, the state of the art is held by the "Laboratoire d'Informatique et de Mécanique pour les Sciences de l'Ingénieur" (LIMSI)lab, that used a hybrid approach that combined data-based dictionaries for feature engineering and linear support vector machines. However, most natural language processing tasks are nowadays typically better handled by neural network based architectures, provided that the learning corpus is of sufficient size. These deep learning based approaches have been applied to the problem at hand in this chapter, mainly through a range of sequence to sequence architectures:

- RNN based encoder decoder architectures (either with or without attention) [68],
- Convolutional based encoder-decoder architectures [69,70],
- Fully attentional (although pre-trained) architectures using a BERT model and transfer learning [71,72].

However, all those techniques, at least when applied to French data, fail to outperform the LIMSI's feature engineering based approach. A possible explanation for this observation might lie in the dataset the teams are given. Indeed, their sample sizes fall usually shy of 200K observations[52], which is usually far from enough for advanced deep learning models to train properly, as modern neural architectures in the neural translation academic literature usually train on datasets with up to tens of millions of observations[41]. This might as well explain why teams using fully attentional models (which are the state of the art nowadays in neural translation) used pre-trained architectures and transfer learning with BERT instead of training a full neural architecture end to end in a purely supervised fashion. The latter is exactly what this chapter sets out to investigate, and constitutes, at least to the author's knowledge, the first attempt at training a modern fully attentional, end-to-end trained model on a dataset with a sample size coherent with the requirements of modern deep learning methods.

4.2.1.2 Material

The dataset used during this study consists of every available death certificate found in the CépiDc database for the years 2011 to 2016, representing just under 3 million training examples. These documents record various information about their subjects, including the chain of events leading to the subject's death, written by a medical practitioner.

Causal chain of death

The causal chain of death constitutes the main source of information available on a death certificate in order to devise mortality statistics. It typically sums up the sequence of events that led to the subject's death, starting from immediate causes (such as cardiac arrest) and progressively expanding into the individual's past to the underlying causes of death. WHO provides countries with a standardized causal chain of events format, which France follows, alongside most developed countries.

This WHO standard asks of the medical practitioner in charge of reporting the events leading to the subject's passing to fill out a two-part form in natural language. The first part is comprised of 4 lines, in which the practitioner is asked to report the chain of events, from immediate to underlying cause, in inverse causal order (immediate causes are reported on the first lines, and underlying causes on the last lines). Although 4 lines are available for reporting, they need not all be filled. In fact, the last available lines are rarely used by the practitioner. The second part is comprised of two lines in which the practitioner is asked to report "any other significant conditions contributing to death but not related to the disease or condition causing it" [73] that the subject may have been suffering from.

In order to counter the language dependent variability of death certificates across countries, a pre-processing step is typically applied to the causal chain of events leading to the individual's death, where each natural language based line on the certificate is converted into a sequence of codes defined by the 10th revision of the International Statistical Classification of Diseases and Related Health Problems (ICD-10)[61]. ICD-10 is a medical classification created by WHO defining 14199 medical entities (e.g. diseases, signs and symptoms…) distributed over 22 chapters and encoded with 3 or 4 alpha decimal symbols (one letter and 2 or 3 digits), 5615 of which are present in the investigated dataset. Table 4.2.1 shows an example of a causal chain of events taken from an American death certificate, in both natural language and ICD-10 formats.

Line	Natural language	ICD-10 encoding
1	STROKE IN SEPTEMBER LEFT HEMIPARESIS	I64 G819
2	FALL SCALP LACERATION FRACTURE HUMERUS	S010 W19 S423
3	CORONARY ARTERY DISEASE	I251
4	ACUTE INTRACRANIAL HEMORRHAGE	I629
Part 2	DEMENTIA DEPRESSION HYPERTENSION	F03 F329 I10

Table 4.2.1: Example of cause chain of death, in natural language and as ICD-10 codes. Some natural language lines correspond to several ICD-10 codes, whose orders matter in the overall coding process

As aforementioned, the process of converting the natural language based causal chain of events leading to death in an ICD-10 format is the main focus of this chapter. Consequently, the latter will be selected as target variable and the former as the main explanatory variable for the neural network based predictive model defined further.

For reasons related to the underlying cause of death production process, the natural language based chain of events and its ICD-10 encoded counterpart suffer from alignment errors at the line level, as shown in table 4.2.2. Although qualitatively deemed quite rare, this misalignment phenomenon brings sufficient noise in the dataset to prevent model convergence while fitting models with line level sentence pairs.

Line	Natural language	ICD-10 encoding
1	STROKE IN SEPTEMBER LEFT HEMIPARESIS	I64 G819
2	FALL SCALP LACERATION FRACTURE HUMERUS	S010 W19 S423
3	CORONARY ARTERY DISEASE	I629 I251
4	ACUTE INTRACRANIAL HEMORRHAGE	
Part 2	DEMENTIA DEPRESSION HYPERTENSION	F03 F329 I10

Table 4.2.2: Same certificate as displayed in table 4.1.1 showcasing the misalignment phenomenon. The ICD-10 code related to line 4 (both in red) has been moved to line 3 by a human coder. Concatenating lines in a backward fashion restores alignment while preserving ordering

In order to bypass this critical flaw in the investigated dataset, a decision was chosen to consider as input and target variables the certificates lines concatenated in a backward fashion (from line 6 to line 1), as can be seen in figure 4.2.1. This slight change in data format does not significantly alter the problematic at hand, as the investigated model is still trained to recognize ICD-10 encoded medical entities from natural language. If anything, the modified modelling problem can be expected to be more difficult, as both the variance and dimensionality of both input and target variables have increased. Several methods are available to retrieve line level aligned predictions from a model trained in such a configuration, for instance using a combination of transfer learning and pruned tree search.

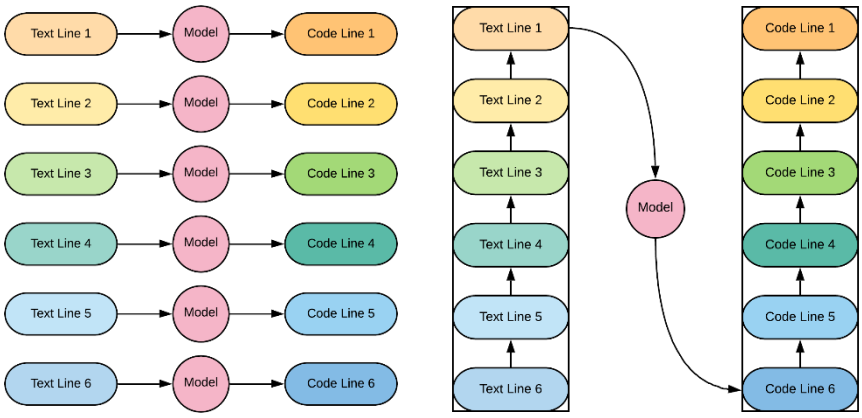


Fig. 4.2.1 Left: the original modelling problem. Each certificate line is taken as an input variable to predict its corresponding ICD10 code line. Right: The modified investigated problem. All certificate lines are concatenated and taken as an input variable to predict the corresponding concatenated ICD10 code line

Miscellaneous variables

From gender to place of birth, a death certificate contains various additional information on its subject besides the chain of events leading to death. As some of these items are typically used by both expert systems and human coders to detect ICD-10 entities in the chain of events, they present an interest as

explanatory variable for the investigated predictive model. After consultation with expert coders, the following items available on French death certificate were selected as additional exogenous variables:

- gender (2 states categorical variables),
- year of death (6 states categorical variables),
- age, factorized into 5 years' intervals with the exception of subject less than one-year-old, which were divided into two classes following whether they were more than 28-day-old,
- origin of the death certificate (2 states categorical variables, either from the electronic or paper based death certification pipeline).

Strictly speaking, the subject's year of passing should only have a limited effect on the relationship between natural language and its contained medical entities. However, the WHO defined coding rules, as well as their interpretations by human coders slightly evolve over the years. As a consequence, the model should benefit, in term of predictive performance, from being able to differentiate between different years.

Similarly, the impact of the certificate's origin on the model's predictive power is not entirely obvious at first sight. However, the paper based certificates data entry process is handled by human through speech recognition technology. In addition, the entry clerks are asked to apply a small set of normalization rules to the natural language. Electronic death certificates, however, are received directly from the medical practitioner as is. As a consequence, distribution shifts are to be expected from paper to electronic based chain of events, and including this information as an explanatory variable might be beneficial to the model's predictive power.

4.2.1.3 Method

With both the explanatory and target variables well defined, the investigated modelling problem can be defined as follows:

$$P(ICD|NL, A, G, Y, E, \theta) = f_\theta(NL, A, Y, G, E) \quad (4.2.1),$$

with:

- $P(X)$ the probability density of discrete random variable X
- $ICD \in [\![0, 1]\!]^{5616^I}$ the sequence of ICD-10 codes present on the death certificate concatenated on a single line of sequence length I
- $NL \in [\![0, 1]\!]^{V^L}$ the line in natural language, tokenized with a vocabulary V and of sequence length L
- $A \in [\![0, 1]\!]^{25}$ the categorized age
- $Y \in [\![0, 1]\!]^{6}$ the year of death
- $G \in [\![0, 1]\!]$ the gender
- $E \in [\![0, 1]\!]$ the death certificate's origin
- f_θ a mapping from the problem's input space to its output space, parameterized in $\theta \in \mathbb{R}^n$ a real-valued vector (typically a neural network) of dimensionality $n \in \mathbb{N}$ the model's dimensionality

Theoretically, the derived modelling problem is typical of traditional statistical modelling problems, and could be solved using multinomial logistic regression. In practice, however, this approach presents a significant drawback. In this setting, the investigated target variable constitutes a categorical variable with 5616^{20} (death certificates in the dataset have at most 20 ICD-10 codes in them, each of which can take 5616 distinct values) distinct states, thus rendering the analysis untractable both in term of computational expanses and sample size requirements. This type of approach, however, makes no use

of the data's inherent sequential nature, which allows to rewrite the investigated modelling problem as follows:

$$P(ICD|NL, A, G, Y, E, \theta) = P\big((ICD_1, -, ICD_n)\big|NL, A, G, Y, E, \theta\big), n \in [\![1, 20]\!] \quad (4.2.2),$$

$$= \prod_{i=1}^{n} P(ICD_i|(ICD_1, -, ICD_{i-1}), NL, A, G, Y, E, \theta), n \in [\![1, 20]\!] \quad (4.2.3),$$

with:

- $ICD_i \in [\![0, 1]\!]^{5616}$, $i \in [\![1, n]\!]$ the i^{th}code present on the code line,
- $n \in [\![1, 20]\!]$ The total number of ICD codes on the death certificate

Factors in the right hand side of equation 2 can be interpreted as distinct predictive modelling problem, all with an output variable distributed across all ICD-10 codes. Although still highly dimensional, predicting output variables of such dimensionality is typically tractable with modern machine learning techniques[19]. They present however two significant drawbacks for traditional modelling techniques:

- The number n of output variables to predict varies across observations in the dataset (not all death certificates have 20 ICD-10 codes),
- The output variables' distributions are conditioned on previous ones.

This particular formulation is known in the deep artificial neural network community as a sequence to sequence modelling problem[19], and has been an active area of research for the past few years. As one of the state of the art neural architecture devised in the field, the Transformer (see chapter 2.4.6) was chosen as the predictive model investigated in the following experiments. It was recently outperformed by the Evolved Transformer[74], a variation on the former. However, both approaches were investigated and yielded similar results. The Transformer architecture was retained due to its availability of official and maintained implementations, and the final results further displayed were

obtained using an ensemble of 7 such models, with a detailed description of the ensembling method used being available in the annex.

Several specificities in the aforedefined modelling problem required small adaptations to the Transformer architecture. However, the authors feel their technicity fall outside the scope of this chapter. The interested reader will however find a complete description of these modifications in the annex documents.

Finally, the authors are aware that many other approaches to sequential learning architectures are available (and already been used) in order to address the problem investigated in this chapter. The current state of the art on French death certificates, for instance uses a multi-label classification approach. However, these methods were deemed unsuited for the task, and this for several reasons:

- The task of extracting ICD10 codes from natural language on death certificates is only a preliminary step in the production of mortality statistics pipeline. The final task in this process is to derive the underlying cause of death from these ICD10 codes, following a set of rules defined by the World Health Organization. The choice of the underlying cause of death from this set of rules heavily depends on the codes' order in the certificate. As a consequence, it is of paramount importance that the model is able to output these codes in the proper order, which is simply unachievable with a multiclass classification approach, and makes the problem a sequential learning problem (as our output is indeed a sequence of variable length tags taken from a set of well-defined class). However, several approaches other than seq2seq are still available to solve such problems (such as connectionist temporal classification, typically used in Optical Character Recognition tasks),
- The ICD10 codes that the model needs to output are not necessarily independent. For instance, the presence of a given code in the outputted sequence can significantly alter other codes present in the sequence. As an example, given by our expert coder, hematoma related codes can be found in several ICD10 chapters. First, in chapter 9 of the ICD10 classification

(codes related to circulatory diseases, beginning by an “I”) and also in chapter 19 (codes related to injury, poisoning and certain other consequences of external causes, beginning with an “S” or a “T”). The choice of attributing the entity “hematoma” present on a death certificate to the first or second possible chapter depends on whether an external cause (meaning an ICD10 code from chapter 20) has already been outputted previously while converting the death certificate into codes. In order to account for such dependencies, we are compelled to model the joint distribution of the output sequence conditioned on the input variables. Which is exactly what seq2seq is about. So the choice of using seq2seq approaches to solving the modelling problem investigated in this chapter becomes not only natural, but almost compulsory. In addition, due to the data-driven tokenization used in order to make use of the ICD-10 classification’s hierarchical nature, some token that the model is allowed to predict are not valid ICD-10 codes. For instance, the code “I659” could be decomposed into a sequence of two codes, [“I65”, “9-“] (with the “-“ character at the end used to keep track of spaces between codes). It appears clear here that when the model needs to output a “I659” code, predicting “9-“ in itself is not possible without any conditioning on “I65” appearing before.

4.2.1.4 Training and evaluation methodology

The investigated model was trained using all French death certificates from years 2011 to 2016. 5000 certificates were randomly excluded from each year and distributed into a validation set for hyper-parameter fine-tuning, and a test dataset for unbiased prediction performance estimation (2500 each), resulting in three datasets with following sample sizes:

- Training dataset: 3 240 109 records
- Validation and test dataset: 30 000 records each

The model was adapted from Tensorflow's (a python-based distributed machine learning framework) official Transformer implementation. Training was performed on three NVidia RTX 2070 GPUs simultaneously using a mirrored distribution strategy using a variant of stochastic gradient descent, the Adam optimization algorithm (see chapter 3.3.3.2).

Hyper-parameters were first initialized following the Transformer's authors in their base setting. Further fine tuning of a selected number of hyper-parameters was performed using a random search guided on the validation set. The interested reader will find a complete description of the training process and hyper-parameter values defining this model in annex.

After training, the model's predictive performance was assessed on the test dataset (excluded prior to training, as mentioned earlier), and compared to the current state of the art, obtained by the LIMSI during the 2017 CLEF eHealth challenge[62]. As the CLEF eHealth challenge only provided electronic certificates to the contestants, and in order to ensure comparability, the model's performances were assessed on paper and electronic certificates separately. For the same reason, the performance metrics used for model evaluation were selected as follows:

$$Precision = \frac{True\ positives}{True\ positives + False\ positives} \quad (4.2.4),$$

$$Recall = \frac{True\ positives}{True\ positives + False\ negatives} \quad (4.2.5),$$

$$F\text{-}measure\ = \frac{2\ \cdot\ Precision\ \cdot Recall}{Precision\ +\ Recall} \quad (4.2.6),$$

with :

- True positives the number of codes predicted by the model that are present in the test set's true output target,

- False positives the number of codes predicted by the model that are not present in the test set's true output target,
- False negatives the number of codes not predicted by the model that are present in the test set's true output.

Note that predictions are considered as true positives only for exact code matches, up to the fourth character. Table 4.2.3 shows an example of how this can affect the reported performances, by focusing on a line of the causal chain of death reported in table 4.2.1 and fictional examples of predictions:

- The first prediction example outputs two incorrect codes. The number of true positives is thus 0, leading to all metrics being evaluated to 0.
- The second prediction example correctly outputs the first code (I64 – "Stroke") but fails to correctly outputs the second character's fourth character (G81 – "Hemiplegia" is predicted instead of the ground truth value G819 – "Hemiplegia, unspecified"). Although the prediction and ground truth are quite similar (they share the three first characters), this code is considered incorrect, which leads to counts of both one false positive (the code was predicted incorrectly) and one false negative (the correct G819 code was not predicted) leading to all metrics being evaluated to .5
- The third prediction example correctly outputs the first code but fails to recognize any additional code from the textual input, leading to a precision of 1 (all predicted codes are indeed true positives) and a recall of .5 (one code present in the ground truth was not predicted), leading to an F-measure of .66. Note that in this context, the F-measure is higher than in example 2
- The fourth prediction example correctly outputs both codes, but also outputs two additional and completely unrelated codes, leading to a precision of .5 (only half of the predicted codes are present in the ground truth), and a recall of 1. (all codes present in the ground truth were correctly predicted), leading to an F-measure of .66.

- The fifth prediction example correctly outputs both codes, and doesn't predict any additional codes (perfect prediction), leading to all metrics being evaluated to 1.
- The sixth prediction example correctly outputs both codes, and doesn't predict any additional codes. The codes are however in the wrong order, but this isn't penalized in any way in the metrics definitions, so this prediction is associated with metrics being all evaluated to 1.

	Value	Precision	Recall	F-measure
Input Text	STROKE IN SEPTEMBER LEFT HEMIPARESIS	-	-	-
True ICD-10 encoding	I64 G819	-	-	-
Prediction example 1	B189 H155	.0	.0	.0
Prediction example 2	I64 G81	.5	.5	.5
Prediction example 3	I64	1.	.5	.66
Prediction example 4	I64 G819 A338 B87	.5	1.	.66
Prediction example 5	I64 G819	1.	1.	1.
Prediction example 6	G819 I64	1.	1.	1.

Table 4.2.3: Examples of how the selected performance metrics behave for different predictions.

The informed reader might find these metrics stray away from common machine translation system benchmarking metrics such as BLEU or negative log perplexity scores[19,41,42,75], but the former were the only ones used in comparable work. As BLEU and negative log perplexity have close to no absolute interpretability without comparisons to alternative methods, their use was discarded from the experiment. In order to present the reader with a more comprehensive view of the proposed approaches' performances, these accuracy metrics were also derived on a per chapter basis, again on the same test set, and confidence intervals were computed using bootstrap.

4.2.2 Results

Performance evaluation

The ensemble of transformer models was trained as aforedescribed for approximately 3 weeks, and the final ensemble's predictive performance as well as the current state of the arts' are reported in Table 4.2.4. As previously mentioned, the current state of the arts' performances was assessed on electronic certificates only, and should as a consequence be compared to the proposed approach performance on a similar situation. Because paper based certificates are still sensibly more common than their electronic counterparts in France (approximately 90% of certificates in the dataset are paper based), overall and paper specific performances are also displayed.

Approach	F-measure	Precision	Recall
Current state of the art (LIMSI)	.825	.872	.784
Proposed approach (electronic certificates)	**.952 [.946, .957]**	**.955 [.95, .96]**	**.948 [.943, .954]**
Proposed approach (paper certificates)	.942 [.941, .944]	.949 [.947, .95]	.936 [.934, .937]
Proposed approach (all certificates)	.943 [.941, .944]	.949 [.948, .951]	.937 [.935, .938]

Table 4.2.4: F-measure of the current state of the art and the proposed approach, with their corresponding 95% confidence intervals, derived by bootstrap. Confidence intervals were not provided in the LIMSI's publication and are therefore not displayed.

The proposed approach shows an F measure 73% closer to a perfect score when compared to the current state of the art. In addition to its substantial improvement in F-measure, the proposed approach displays significantly more balanced precision and recall scores than the LIMSI's method (from 5% relative difference to less than 1%).

A surprising result, however, lies in the model's lower performances on paper certificates. Indeed, the standardization they receive due to their voice based data collecting process considerably reduces variance and prevents any misspelling of words in the data potentially present in electronic based certificates. As a consequence, model performance on the former should be expected to be higher. A potential explanation for this phenomenon lies in the potential for missing data in paper based

certificates. Indeed, when confronted to poorly written words, data clerks are allowed to replace them with a "!" symbol when the word is estimated unreadable (present in approximately 10% of paper based certificates). Medical coders, however, are usually more efficient in guessing the words from the written certificates (typically with the addition of contextual clues). A purely text based approach however, is then limited to pure guess on those observations with missing data, logically leading to poorer performance. This phenomenon being absent from electronic based certificates; it constitutes a promising candidate in explaining this unexpected difference of performance. In addition, model performances on paper certificates not containing any "!" symbol in the test set led to 96.2% F-measure, thus providing strong evidence to support this hypothesis.

Per-chapter quantitative analysis

Although the proposed approach significantly outperforms the current state of the art, neural network based methods are known to present several drawbacks that can significantly limit their application in some situations. Typically, the current lack of systematic methods to interpret and understand neural network based model and their decision processes can lead the former to perform catastrophically on mispredicted cases, independently from their high predictive performances. As a consequence, the proposed model behaviour in mispredicted cases require careful analysis. In addition, such an investigation can lead to significant insights potentially relevant when applying the derived model in practical applications.

One simple, straightforward approach to understanding the model's weakness, lies in assessing its performance on a finer grain level, for instance by identifying false positives and negatives not only at the global level, but per ICD-10 chapters, as can be seen in table 4.2.5.

It appears from these graphs that although the most prevalent medical entities are associated with low false positive and negative rates, some rarer chapters are associated with unreasonably high error

rates. Depending on their prevalence and accuracies, these chapters can be classified into two distinct categories:

- Chapters associated with unreasonably high error rates but extremely low prevalence such as “diseases for the ear and mastoid process” or “pregnancy, childbirth and the puerperium”. However, these entity groups remain rare enough within the dataset to allow for alternative treatments, like manual evaluation, for instance.
- Chapters associated with high error rates (although lower than the former) but with significant prevalence such as “External causes of morbidity and mortality” or “Injury, poisoning and certain other consequences of external causes”.

The task of identifying these potential mistakes, however, is not entirely trivial depending on whether mistakes are of false positive or false negative types. Indeed, potential false positives errors are directly identifiable within the predicted ICD-10 code sequences. As a consequence, coding quality control for this mistake type should be fairly straightforward to implement (one could for instance manually review all code sequences containing codes related to “Pregnancy, childbirth and the puerperium” systematically. Potential false negative errors, however, are inherently significantly harder to identify, and require further investigation, for instance through association rules analysis.

A number of promising leads are already available and should reasonably improve upon the proposed approach:

- Training methods adapted to imbalanced datasets such as up sampling or loss weighting,
- Data augmentation for rare medical entities,
- Addition of information to the model (prenatal related death, for instance, are explicitly defined as such on certificates),
- Hybrid approach with traditional NLP approaches, typically less expensive in term of sample size requirements.

ICD-10 chapter	False positives (%)	False negatives (%)	Prevalence (%)
Diseases of the circulatory system	3.75	4.98	22.4
Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified	3.87	4.12	21.8
Neoplasms	4.07	5.07	15.9
Diseases of the respiratory system	3.02	4.00	8.76
Endocrine, nutritional and metabolic diseases	2.17	3.44	4.83
Diseases of the nervous system	2.70	4.12	3.89
Mental and behavioural disorders	2.88	4.14	3.58
Diseases of the digestive system	5.72	8.10	3.53
Factors influencing health status and contact with health services	19.2	19.6	3.08
Diseases of the genitourinary system	5.45	7.59	2.71
External causes of morbidity and mortality	16.6	23.5	2.57
Certain infectious and parasitic diseases	7.98	9.23	2.55
Injury, poisoning and certain other consequences of external causes	14.0	19.8	2.07
Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism	6.72	12.2	0.77
Diseases of the musculoskeletal system and connective tissue	12.2	17.3	0.62
Diseases of the skin and subcutaneous tissue	8.72	8.16	0.51
Certain conditions originating in the perinatal period	14.5	20.5	0.16
Congenital malformations, deformations and chromosomal abnormalities	22.4	25.6	0.15
Diseases of the eye and adnexa	4.93	13.6	0.076
Codes for special purposes	24.0	34.0	0.047
Diseases of the ear and mastoid process	5.6	33.3	0.017
Pregnancy, childbirth and the puerperium	50	33.3	0.0056

Table 4.2.5: Prevalence, false positives and negatives rates for each ICD-10 chapter, sorted in descending order by prevalence

Score calibration fitness assessment

The model being fit in a similar fashion to multinomial logistic regression, it not only yields a prediction, but an associated score similar to a confidence probability. If properly calibrated, this score can offer powerful insights regarding the prediction's quality at the individual level. Typically, a "good" score would be expected to show higher values in cases where the predicted ICD-10 sequence is correctly predicted and a low one when mispredicting. Such a well calibrated score could for instance allow for real-world applications of semi-autonomous systems where:

- A threshold value for the model's score is defined,
- All certificates whose predictions are associated with confidence scores above the threshold level are accepted without any additional human supervision,
- All certificates whose predictions are associated with confidence scores below the threshold level are systematically reviewed by a human expert, and modified manually if required.

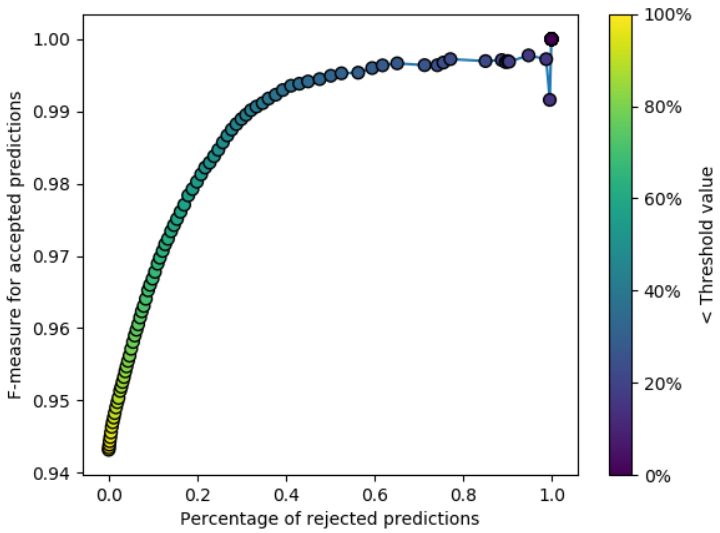


Fig. 4.2.2 Percentage of rejected predictions versus F-measure on accepted ones. The score threshold value defining the accepted predictions are displayed as point colours

Being able to properly filter the model's predictions according to a well calibrated confidence score would thus allow to get the best of both worlds. Most of the certificates would be automatically coded by the autonomous system, leaving for human coders only the most complex cases.

Efficient assessment of such scores in traditional machine learning problems is typically done through visualization of ROC curves. However, the sequential, multinomial nature of the investigated problem renders this approach ill defined. The plot found in figure 4.2.2, while conceptually similar to a ROC curve, was derived following a slightly different approach in order to efficiently appreciate the model score's quality:

- A grid of score threshold values was defined (uniform grid with 0.01 intervals), corresponding to the threshold defined above, filtering between model predictions that would require human examination or not,
- For every given threshold value were computed the percentage of predictions with inferior or equal scores (considered as rejected requiring human examination due to poor score), as well as the F-measure performance on the predictions with high enough scores (that would be accepted without any human intervention following the above example),
- Percentage of accepted certificates and F-measurement were scatter plotted against each other, with threshold value displayed as points' colour

By showing a clean, increasing relationship between the number of rejected predictions and the F-measure evaluated on the remaining certificates, figure 4.2.2 strongly indicates good score calibration. As an example, by considering that only predictions associated with a confidence score lower than .5 as not requiring any additional human supervision, the system is able to code approximately 80% of all certificates present in the test set with a F-measure of .98, significantly higher than the .94 obtained on all test certificates.

4.2.3 Discussion

The error analysis carried on so far allowed for the assessment of the model's strength and weaknesses on the global level. They however fail to yield any interesting insight regarding potential model biases, for instance towards specific coding rules. Indeed, the coding of medical entity from natural language, especially with regard to mortality statistics, is subject to a number of coding rules depending on context or pathology, with a level of specificity oftentimes reaching casuistry[61].

In addition, all results presented so far with a model error defined as a disagreement between the model's output and the information contained in the database. However, building a medical database

is a complex, mostly human-based process. As such, an inevitable amount of noise is to be expected in the ICD-10 codes presents in the database, in two main forms:

- Simple human errors in the ICD-10,
- The presence of unreadable text in paper certificates. Unreadable words on paper certificates are denoted as an exclamation point in the textual data fed to the model. However, human coders usually take additional time in order to infer these words, for instance from contextual cues. This leads to death certificates in the database where the ICD-10 sequences contain additional codes compared to the textual data available. As such, not predicting these codes would result in a drop in performance metrics, while the model has no way of predicting them. An example of such a death certificate can be found in table 7.3 of the annex.

These phenomena have the potential to negatively bias the proposed model's performance estimations, and should be the object of further investigation.

One straightforward, although fairly time-consuming approach to address these two considerations can be obtained by having a ICD-10 coding expert manually examine some of the death certificates where the model's predictions do not match the ICD-10 codes present in the database. Two experiments were conducted following this idea.

In a first experiment, 99 certificates where the model's predictions did not exactly match with the database's ICD-10 variables (meaning that the ICD10 sequences differed by at least one code) were selected at random from the test set and shown to the medical practitioner referent and final decision-maker on ICD-10 mortality coding in France, who was asked for each certificates to:

- Manually recode all the ICD-10 medical entities present on each death certificates by herself, from the information the proposed model had access to, without access to the dataset and model's ICD-10 sequences proposal,

- Give a qualitative comment on the investigated model and database's outputs compared to hers.

The ICD-10 sequences derived from the medical expert and national referent for ICD-10 coding in France being significantly more reliable than the ones coming from the traditional data production process (using a combination of expert system and human coders), they can be considered as exempt of any potential human error. As a consequence, comparing them to both the proposed model's output and the ICD-10 values contained in the dataset would allow for an estimation of the potential negative biases described above. This can be done for instance by estimating the performance metrics selected for the previous experiments, considering both the model's predictions and the database's values as predictions, and the medical expert's outputs as the ground truth. Depending on the resulting values, several interpretations can be made ranging from two extreme cases:

- If perfect agreement (meaning an F-measure of 1.) is reached between the database's ICD-10 sequences and the medical expert, suggesting that the database does not have coding mistakes, then the performance metrics reported in the results section can safely be considered unbiased
- If perfect agreement is reached between the model's prediction's ICD-10 sequences and the medical expert, suggesting that the model does not do any mistakes, then the performance metrics reported in the results section should be considered significantly underestimated

Before estimating the performance metrics following this methodology, a slight pre-processing step is however required. Indeed, on the death certificates sampled for the experiment, F-measure estimation between the model's prediction and the database's ICD-10 sequences yields a value of .81. This is explained by the sampling process that selects death certificates where at least one code differs in both ICD-10 sequences. As a consequence, and because of the model's performances, most ICD-10 codes present on both sequences are identical, as can be seen on error examples presented in the annex' tables 7.3, 7.4 and 7.5. The authors felt that this might lead to artificially high values of the

estimated metrics in the experiment, and consequently decided to delete all common codes on both the model's outputs and database's values, prior to metrics estimation, as can be shown in table 4.2.6.

	Before pre-processing	After pre-processing
Predicted ICD-10	I259 Z951 I719 C679 I10 R092	Z951
Database ICD-10	I259 I251 I719 C679 I10 R092	I251
Medical expert ICD-10	I259 I251 I719 C679 I10 R092	I251

Table 4.2.6: Example of preprocessing used for the experiment on a real error example. The predicted and database ICD-10 sequences only differ by one code, while they share 5 codes. All these shared codes are deleted from all ICD-10 sequences prior to performance metrics estimation

For better comparability, these statistics are reported both on:

- Certificates without missing data in the natural language based causal chain of death (by excluding certificates containing a "!" symbol) in table 4.2.7,
- All certificates in table 4.2.8.

Database or prediction	F-measure	Precision	Recall
Database against medical expert	.483 [.383, .589]	.443 [.341, .555]	.531 [.425, .636]
Prediction against medical expert	.431 [.316, .542]	.458 [.338, .580]	.407 [.295, .519]

Table 4.2.7: F-measure, precision and recall (with their 95% confidence intervals) of both the database and the model's predictions against the medical expert for sampled certificates without missing data

Database or prediction	F-measure	Precision	Recall
Database against medical expert	.613 [.486, .733]	.630 [.492, .761]	.596 [.471, .721]
Prediction against medical expert	.370 [.237, .504]	.392 [.250, .54]	.351 [.222, .482]

Table 4.2.8: F-measure, precision and recall (with their 95% confidence intervals) of both the database and the model's predictions against the medical expert for all sampled certificates

Tables 4.2.7 and 4.2.8, show no significant difference in prediction performance between the proposed approach and the current data production process (based on a combination of expert system and human coders), although the database's ICD-10 values have better performance metrics on both cases. When including certificates containing missing text, the proposed model agreement with the medical expert increases considerably to an F-measure of above .96, further confirming the hypothesis that the performance metrics reported in the result section are negatively biased.

From the qualitative comments made by the medical experts, three major types of model errors could be defined:

- In 16% of cases, disagreement between the current data production process and the proposed approach was due to missing information in the input text. On these specific cases, the F-measure between model output and medical expert decision was measured at .974 (an example of such error case can be seen in the annex, in table 7.3),
- In 14% of cases, the correct ICD-10 sequence is dependent on highly contextual clues or external knowledge of world behaviour (e.g. someone found dead at the bottom of stairs is quite likely to have suffered a fall. An example of such error case can be seen in the annex, in table 7.4),
- In 12% of cases, the correct ICD-10 sequence is dependent on highly nonlinear, almost casuistic rules and are typical examples of scenarios where a hybridized deep learning and expert-based

system should be beneficial (an example of such error case can be seen in the annex, in table 7.5)

- The remaining cases did not elicit any comment from the medical expert.

Finally, in a second experiment, the medical expert's ability to discriminate between human coding and the proposed approach was assessed, in a Turing test-like approach. To do so, a hundred additional certificates where the model's output differed from the database's ICD-10 sequences were sampled at random from the test set. The medical expert was shown their corresponding input features (text and auxiliary variables), as well as the two ICD-10 sequences (with their provenance, either from the model or the database, masked) as can be seen in table 4.2.9.

Sex	Year	Age	Text	Proposition 1	Proposition 2
2	2013	90	90 ans, péritonite, perforation grêle, occlusion, chirurgie digestive, infection pulmonaire, arrêt respiratoire	R54, K566, K659, K631, Y839, J958, R092	R54, K659, K631, K566, Y839, J189, R092

Table 4.2.9: Example of death certificate format given to the medical expert for the second experiment. The medical expert is asked from the information available in the line to guess which of proposition 1 or proposition 2 was produced by a human coder, the other being the proposed model's output

After exclusion of certificates containing missing text data (where the human coder was easily identifiable due to the apparently out of context additional codes as seen in table 7.3), the medical expert was able to correctly identify the human in 62.0% [50.7, 73.2] of cases, which is significantly better than random guessing (although barely).

4.2.4 Conclusion

In this section, the task of automatic recognition of ICD-10 medical entities from natural language in French was presented as a seq2seq modelling problem, well known in the deep artificial neural

network academic literature. From this consideration, the performances of a well-known approach in the field, consisting of an ensemble of Tranformer models, was investigated using the CépiDc database and shown to obtain a new state of the art. The derived model's behaviour was thoroughly assessed following different approaches in order to identify potential weaknesses and leads for improvements. Although the proposed approach significantly outperforms any other existing automated ICD-10 recognition systems on French free-text, the question of method transferability to other languages require more investigations.

The substantial performances reported in this chapter open an entire range of promising applications in various medical related fields, from medical act automated coding to advanced natural language based analysis for epidemiology. However, these interesting opportunities are oftentimes prohibited by these methods' massive drawbacks, mostly their requirement for millions of annotated observations to perform well. Mortality datasets, in spite of their specificity, provide researchers with huge, clean and multilingual medical text data perfectly fit for the application of deep neural networks. As a consequence, and keeping in mind neural network's strong transfer learning capability, the authors firmly believe that mortality data constitutes one of the most promising point of entry into modern natural language processing methods applications in the biomedical sciences.

4.3 A deep artificial neural network based model for underlying cause of death prediction from death certificates

As aforementioned, the process of deriving the underlying cause of death from raw death certificates is divided into two main steps, the first one being the identification of pathologies from the textual death certificate, which was addressed in the previous chapter, and the final one being the identification of the actual underlying cause of death from the ICD-10 encoded causal chain of death.

Nowadays, in order to preserve spatial and temporal comparability, the underlying cause of death is usually identified from an expert system[76] (such as the Iris software[77]), a form of artificial intelligence that encodes a series of WHO-defined coding rules. Unfortunately, these decision systems fail to handle a significant amount of more complex death scenarios, typically including multiple morbidities or disease interactions. These cases then require human evaluation, consequently leading to a time consuming coding process potentially subject to distributional shift across both countries and years, sensibly impairing the statistics' comparability.

The following chapter formulates the process of extracting the underlying cause of death from death certificate as a statistical predictive modelling problem, and proposes to solve it with a deep artificial neural network. The following section focuses on describing the structured information contained in a death certificate. Section 3 introduces the neural network architecture used for the task of predicting the underlying cause of death. Section 4 reports the results obtained from training the neural network on French death certificate from the years 2000 to 2015 (about 8 million training examples) as well as a comparison with prediction performances obtained using the Iris software, current state of the art for this predictive task and solution used in numerous countries for underlying cause of death coding. Finally, section 5 shows an application of the derived model on opioid overdose related deaths in France.

4.3.1 Material and method

4.3.1.1 Dataset

The dataset used during this study consists of every available death certificate found in the CépiDc database for the years 2000 to 2015, and their associated cause of death, representing over 8 million training examples. These documents record various information about their subjects, with varying predictive power with regard to the underlying cause of death. The following chapter aims to derive a

deep neural network based predictive model explaining the underlying cause of death from the information contained within death certificates:

$$P(UCD|DC) = f(DC) \quad (4.3.1),$$

with:

- DC the information contained in a French death certificate
- UCD its corresponding underlying cause of death
- f a neural network based predictive function

In order to model the underlying cause of death from these information, the following items were selected as explanatory variables:

- The causal chain of events leading death
- Age
- Gender
- Year of death

Causal chain of death

As aforementioned, the causal chain of death constitutes the main source of information available on a death certificate in order to devise its corresponding underlying cause of death. The latter being the target of the investigated predictive model, the information contained in the causal chain of death is of paramount importance to decision process leading to the underlying cause of death's establishment. In order to enforce death statistics comparability across countries, the coding of the underlying cause of death from the causal chain of events is defined from a number of WHO issued rules bases, oftentimes reaching casuistry on more complex situations[78].

As seen in the previous chapter, and this to counter language dependent variability of death certificates across countries, the decision rules governing the underlying cause of death process are actually defined from this ICD-10 converted causal chain, and the former is to be reported as a unique ICD-10 code.

The processed causal chain of death, in its encoded format, can be assimilated as a sequence of 6 varying length sequences of ICD-10 codes. In order to simplify both the model and computations, this hierarchical data structure will hereon be assimilated, as seen in figure 4.3.1, as a padded 6 by 20 grid of ICD-10 codes, with rows and columns denoting a code's line and rank in line, respectively, 20 being the maximal number of ICD-10 codes found on a causal chain line in all certificates present in the investigated dataset. Several, subtler approaches to this grid like assimilation were explored prior to the experiment reported in this chapter, but all yielded models with significantly inferior predictive power. Although this encoding scheme apparently prevents the encoding to handle death certificates with at least one line containing more than 20 codes, the model introduced further itself sees no such limitation. Bigger certificates can be processed without trouble with an appropriately larger code matrix encoding, with theoretically no significant loss in performance, since the model is translation invariant[79].

The question of encoding ICD-10 codes in a statistically exploitable format is another challenge in itself. A straightforward approach would be to factor each ICD-10 code as a 7404 dimensional dummy variable. This simple encoding scheme might however be improved upon, typically by exploiting the ICD-10 hierarchical structure by considering codes as sequences of character. This approach was investigated, but yielded significantly lower results. As a consequence, the results reported in this chapter only concern the dummy variable encoding scheme.

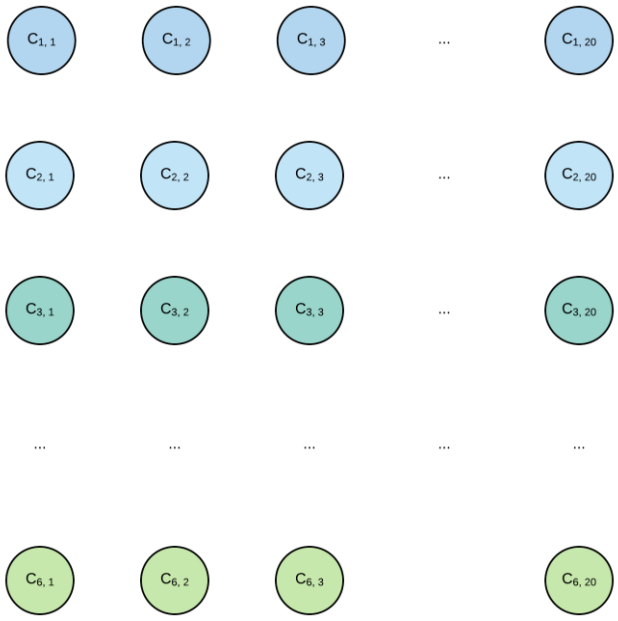


Fig. 4.3.1 Causal chain of death encoded as a 3 dimensional tensor. Each node represents an ICD-10 code as a 7404 dimensional dummy variable. Its row and column position respectively denotes the corresponding code's line and rank in the corresponding certificate

Miscellaneous variables

From gender to birth town, a death certificate contains various additional information on its subject besides the chain of events leading to death. As some of these items are typically used by both Iris and human coders to decide the underlying cause of death, they present an interest as explanatory variable for the investigated predictive model. After consultation with expert coders, the following items available on French death certificate were selected as additional exogenous variables:

- gender (2 states categorical variables),
- year of death (16 states categorical variables),
- age, factorized into 5 years' intervals from subject less than one year old, which were divided into two classes.

Strictly speaking, the subject's year of passing should only have a limited effect on the underlying cause of death However, the WHO defined coding rules, as well as their interpretations by human coders slightly evolve over the years. As a consequence, the model should benefit, in term of predictive performance, from being able to differentiate between different years.

4.3.1.2 Neural architecture

With the death certificate and its selected variables converted into a format enabling analysis, the underlying cause of death extraction task can be solved by estimating its corresponding ICD-10 code's probability density, conditioned on the aforedefined explanatory variables:

$$P(UCD|CCD, A, Y, G, \theta) = f_\theta(CCD, A, Y, G) \quad (4.3.2),$$

with:

- $UCD \in \mathbb{R}^{7404}$ the underlying cause of death
- $CCD \in \mathbb{R}^6 \times \mathbb{R}^{20} \times \mathbb{R}^{7404}$ the ICD-10 grid encoded causal chain of death
- $A \in \mathbb{R}^{25}$ the categorized age
- $Y \in \mathbb{R}^{16}$ the year of death
- $G \in \mathbb{R}^2$ the gender
- f_θ a mapping from the problem's input space to its output space, parameterized in θ a real-valued vector (typically a neural network)

This approach can be yet improved upon by stacking several neural layer between the input variable and the final linear model output, progressively increasing the model's number of degrees of freedom and non-linear explanatory power[80] and leading to the creation of a wide family of predictive model, with state of the art performance in a wide variety of tasks, typically in computer vision and natural language processing. Although the currently investigated modelling problem does not fall into one of

these categories, recent advances in both deeply inspired the neural architecture presented in this chapter, which can be seen in figure 4.3.2 and can be decomposed as follows:

- Linear projections are applied to each one-hot encoded categorical variable[65] (one linear projection is shared for all ICD-10 codes present in the causal chain of death), with all linear projections sharing the same output space dimension,
- The miscellaneous variables' projections are added to all of the projected grid's elements,
- The resulting grid is used as input to a convolutional neural network[81],
- A multinomial logistic regression (softmax regression) targeting the underlying cause of death is performed on the convolutional neural network's output[82],
- All model parameters (from both the linear projections and the convolutional network) are adjusted by minimizing a cross-entropy objective using gradient based optimization. The model's gradients are computed using the backpropagation method[83].

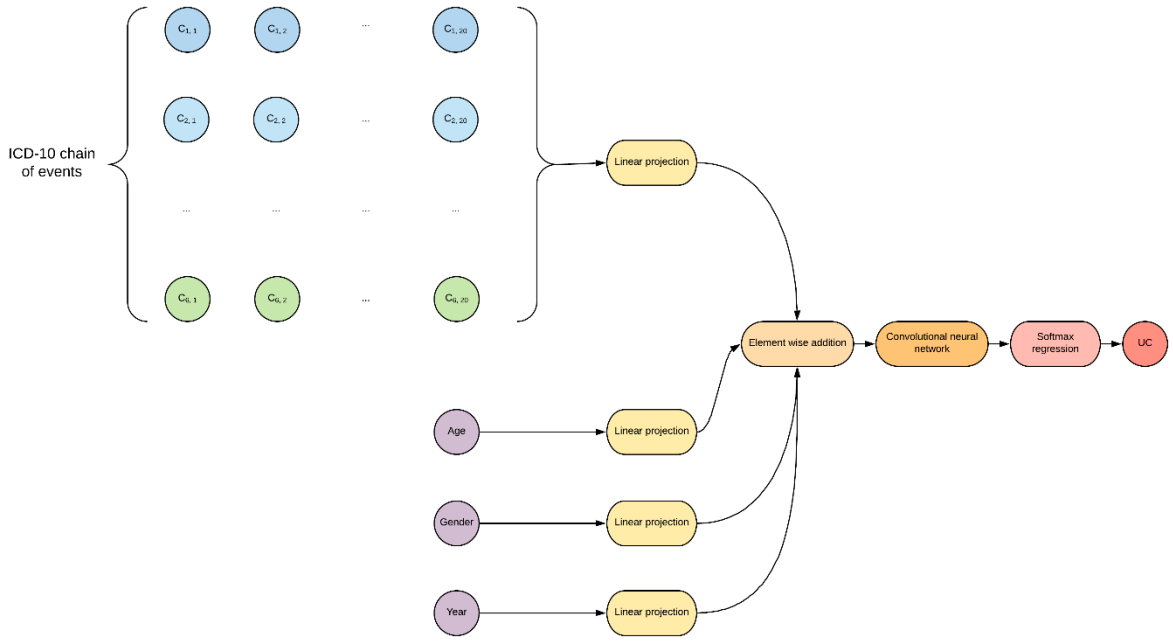


Fig. 4.3.2 Overall model architecture

4.3.1.3 Training and evaluation methodology

The investigated model was trained using all French death certificates from years 2000 to 2015. 10,000 certificates were excluded from each year and spread into a validation set for hyper-parameter fine-tuning, and a test dataset for unbiased prediction performance estimation (5000 each), resulting in three datasets with following sample sizes:

- Training dataset: 8553705 records,
- Validation and test dataset: 80000 records each.

The model was implemented with Tensorflow, a python-based distributed machine learning framework, on two NVidia RTX 2070 GPUs simultaneously using a mirrored distribution strategy. Training was performed using a variant of stochastic gradient descent, the Adam optimization algorithm.

The numerous hyper-parameters involved in the model and optimization process definition were tuned using a random search process. However, due to the significant amount of time required to reach convergence on the different versions of the model trained for the experiment (around 1 week per model) only three models were trained, the results displayed below being reported from the best of them, in term of prediction accuracy on the validation set. The interested reader will find a complete list of the hyper-parameters defining this model in annex. Given the considerably small hyper-parameter exploration performed for the experiment reported in this chapter, the authors expect that better settings might provide with a slight increase in prediction performance. However, given the successful results obtained and the computational cost of a finer tuning, a decision was taken to not further the exploration.

After training, the model's predictive performance was assessed on the test dataset (excluded prior to training, as mentioned earlier), and compared to the Iris software's, nowadays considered as the state

of the art in automated coding and internationally used. In order to ensure a fair comparison between the two systems, Iris' performances were assessed on the test set as well and given the same explanatory variables. As is done traditionally in the machine learning academic literature, the predictive performance is reported in term of prediction accuracy, namely the fraction of correctly predicted codes in the entire test dataset.

The Iris software' automatic coding accuracy was assessed with two distinct values resulting from the software's ability to automatically reject cases considered as too complex to be handled by the decision system. As a consequence, a first accuracy measurement (the lowest one) was assessed considering rejects as ill-predicted cases, while the second one excluded these rejects from the accuracy computation, thus yielding an improved estimate. In order to present the reader with a more comprehensive view of both approaches' performances these accuracy metrics were also derived on a per chapter basis, again on the same test set.

4.3.2 Results

The neural network based model was trained as aforedescribed for approximately 5 days and 18 hours, and its predictive performance as well as Iris' are reported in table 4.3.1.

The neural network based approach to underlying cause of death automated coding significantly outperforms both metrics. Indeed, even when compared to Iris' performance on non-rejected cases, the error rate offered by the proposed approach is 3.4 times lower. This performance difference increases to an eleven-fold decrease when including rejected cases in Iris performance.

Selected approach	Prediction accuracy
Iris overall accuracy	$0.745\ [0.740, 0.750]$
Iris on non-rejected certificates	$0.925\ [0.921, 0.928]$
Proposed approach	$\mathbf{0.978}\ [\mathbf{0.977}, \mathbf{0.979}]$

Table 4.3.1 Prediction accuracy of Iris and the best derived predictive model, with their corresponding 95% confidence intervals, derived by bootstrap

In addition, Figure 4.3.3 shows the model's error rates per ICD-10 chapter, alongside the latter's prevalence. In this plot chapter VII, diseases of the eye and adnexa, appears as a strong outlier in term of error rate. Also not statistically significant (only 3 death certificates among the 80 thousand sampled for the test set have a chapter VII related underlying cause of death), this observation might indicate that the training set does not have a big enough sample size to allow the model to handle extremely rare cases such as chapter VII related death certificates, which might better be handled by a hand crafted, rule based decision system.

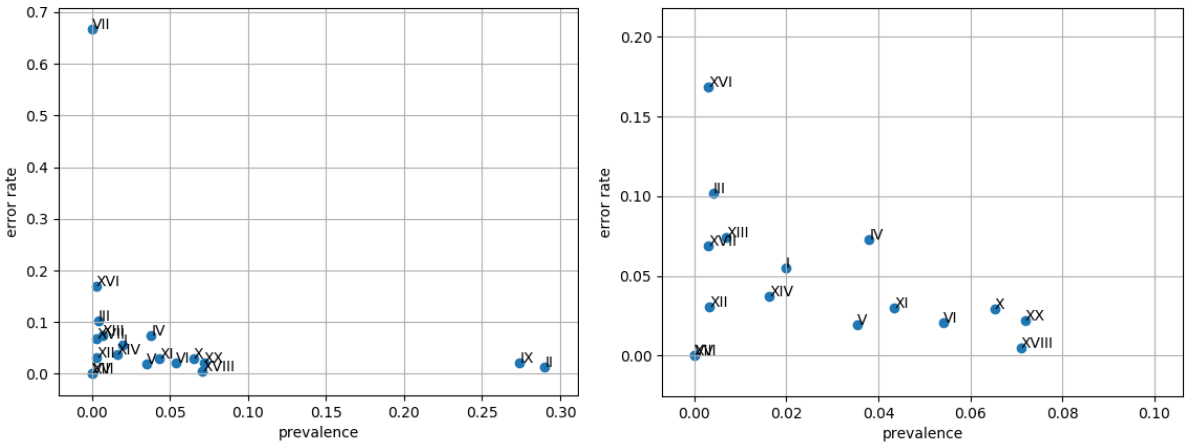


Fig. 4.3.3 Left: Prevalence of underlying causes (by ICD-10 chapter) against ICD-10 chapter level model error rate. Right: Zoom on the previous plot's bottom left hand corner

Finally, figure 4.3.4 shows the per chapter difference in error rate between the proposed neural network approach and the Iris software (on non-rejected certificates). As previously hypothesized, the Iris software outperforms the deep learning approach on diseases of the eyes and adnexa related

death certificates (chapter VII), although still not significantly. Even if the Iris software is beaten in every other chapter, a case should be made from never appearing chapters. Indeed, a number of chapters (namely chapters XIX, XXI and XXII) are not observed as underlying causes in the test dataset, strongly indicating that they might benefit from a set of hand crafted rules as do chapter VII related certificates, if they were to appear in extremely rare cases.

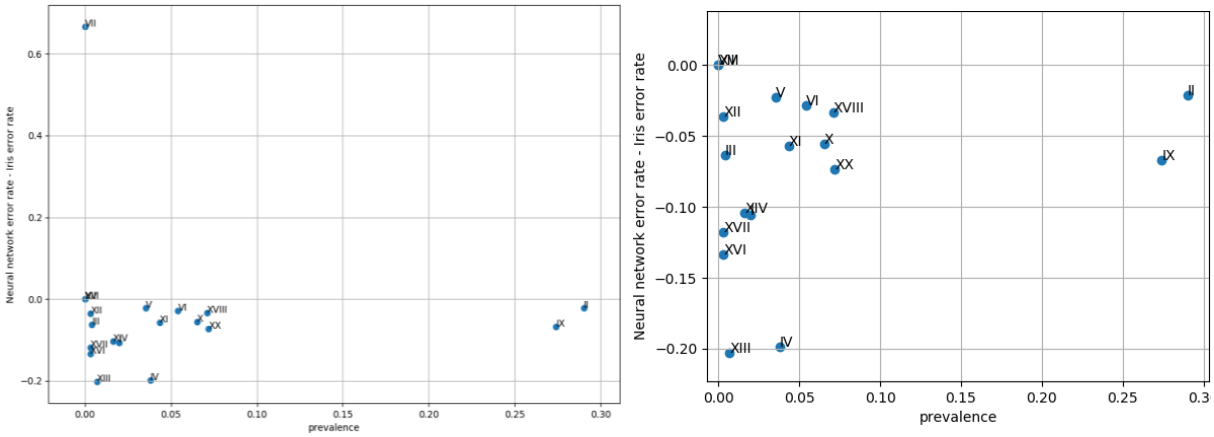


Fig. 4.3.4 Left: Difference in error rate between the proposed model and the Iris software versus chapter prevalence as underlying cause. Right: Zoom on the previous plot's bottom left hand corner

4.3.3 Error analysis

Although the proposed approach significantly outperforms the current state of the art that is the Iris software, neural network based methods are known to present several drawbacks that can significantly limit their application in some situations. Typically, the current lack of systematic methods to interpret and understand neural network based model and their decision processes can lead the former to perform catastrophically on ill predicted cases, independently from their high predictive performances.

As a consequence, the proposed model behaviour in ill predicted cases require careful analysis. In addition, the system's performance can potentially benefit from such an investigation. For instance, although the model outperforms Iris on average, there might some highly non-linear exceptions that

are better fit to rule-based decision systems, in which case a hybrid approach could, by using the best of both world, again yield performance gains.

Although assessing per chapter error rates, as previously shown, constitutes a simple, straightforward approach to understanding the model's weakness, much more can be done to gain insight into the model's behaviour. As an example, it only feels natural, after identifying cases incorrectly predicted by the investigated model, to assess the nature of errors made by the latter. As aforementioned, neural network based classifiers tend to, in misprediction cases, output answer unreasonably far from the ground truth. One should however expect from a good predictive model to, in error cases, output predictions as close as possible to the correct answer. Figure 4.3.5 displays an ICD-10 chapter level confusion matrix built from ill predicted test cases, and shows that beside chapter VII, most of the errors remain in the same chapter as the ground truth, indicating some degree of model robustness.

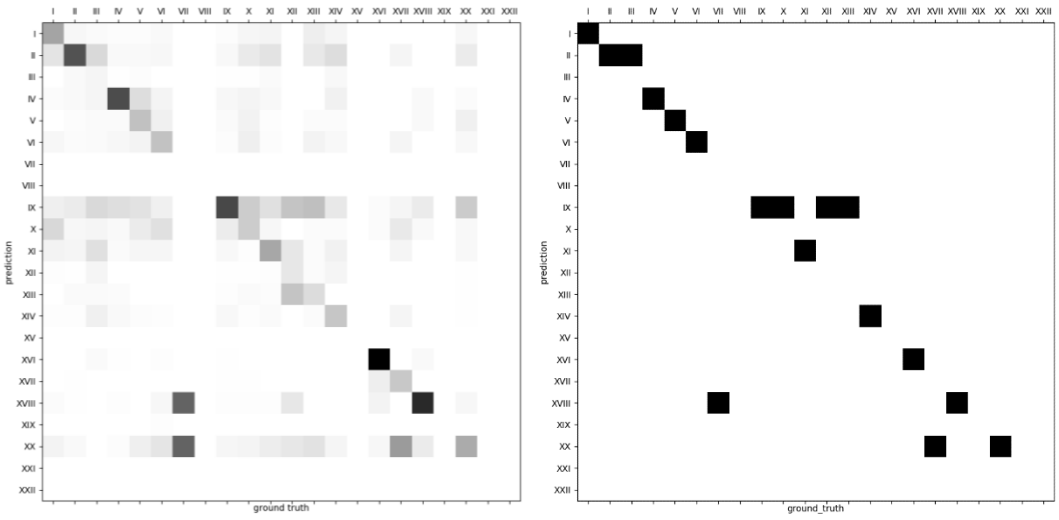


Fig. 4.3.5 Left: distribution of wrong predictions per ICD-10 chapter versus their ground truth (the lighter the rarer). Right: Same distribution's modes. Apparent missing values in both figure correspond to chapters either not represented in the test dataset or on which no mistakes were made

The model's error behaviour can also be investigated from a calibration fitness perspective. As aforementioned, some artificial neural network based models have been known to behave quite poorly in ill predicted cases, which could constitute a highly undesirable phenomenon when handling health data. The model being fit in a similar fashion to multinomial logistic regression, it not only yields a prediction, but an entire probability distribution over all possible ICD-10 codes. Assessing this distribution can offer powerful insight for such considerations. Typically, a "good" predictive model would be expected to show high confidence in cases where the prediction is correct, and a low one when mispredicting. Bar plots of said prediction confidences can be found in figure 4.3.6, and clearly show a strong tendency for the model to be more confident in its prediction in correctly predicted cases.

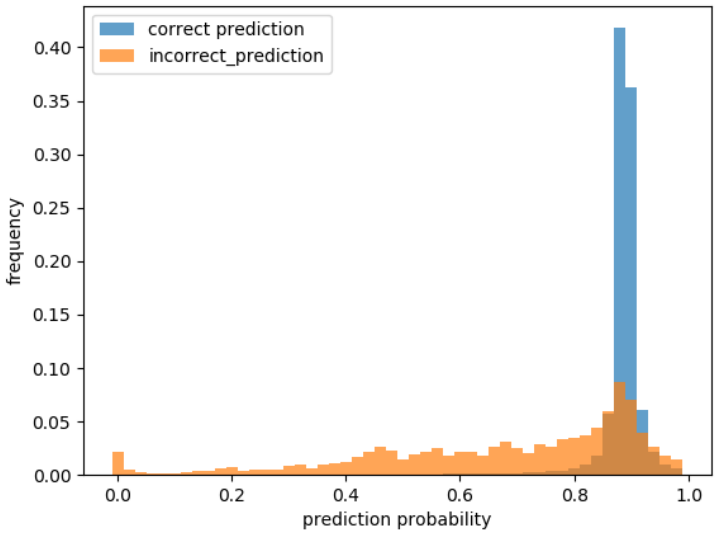


Fig. 4.3.6 Histograms of prediction confidence in correct (blue) and incorrect (orange) predictions. The model typically predicts correct values with high confidence, and incorrect values with lower confidence

If predicting incorrect values with low confidence is a desirable behaviour for a predictive model, associating the ground truth with high probabilities, even in misprediction cases, should be of equal importance. This is typically assessed by evaluating whether each test set subject's corresponding ground truth is contained in the $k \in \mathbb{N}^*$ most probable values present in the model's corresponding

outputted distribution. This type of metric is typically denoted as the model's top-k accuracy, and helps assessing a model's ability to give high confidence to correct values, even when mispredicting. Although the academic machine learning literature typically makes use of the top-5 accuracy in such cases, the investigated model was investigated with a top-2 accuracy only. Indeed, most death certificates present in the dataset display causal chains of events with 5 or less ICD-10 codes, with the underlying cause of death being one of them. It is consequently reasonable to expect the model to output these 5 codes as most probable, thus leading to a high but meaningless top-5 accuracy. The assessed top-2 accuracy can be found in table 4.3.2, and strongly indicates that the model consistently associates correct underlying causes of death with higher probabilities, even in ill predicted cases.

Second most probable code prediction accuracy on ill predicted certificates	$0.663\ [0.641, 0.685]$
Proposed model Top-2 accuracy	$0.993\ [0.992, 0.993]$

Table 4.3.2 Accuracies on codes wrongly predicted by the proposed model, and model top-2 accuracy

A richer, although more time consuming, error analysis can be derived from human observation of each error cases by an underlying cause of death coding specialist. To do so, 96 of the 1777 ill predicted death certificates in the test set were selected at random and shown to the medical practitioner referent and final decision-maker on underlying cause of death coding in France, who gave for each of the selected certificates:

- Her personal opinion of what each certificate's corresponding underlying cause should be,
- A qualitative comment on the investigated model's error.

The aforementioned underlying causes obtained were then confronted with both the actual values contained in the dataset and those predicted by the derived model, leading to the following observations:

- In 41% of cases, the referent agreed with the model's predictions,
- In 37% of cases, the referent agreed with the underlying cause present in the dataset,
- In 22% of cases, the referent agreed with neither of them.

Consequently, the derived predictive model's coding can be considered as comparable in quality to the actual process responsible for the production of the investigated dataset's. In addition, a qualitative analysis of the medical practitioner's comments on the model's mistakes showed that 30% of errors committed by the predictive model are related to casuistic exceptions in coding rules, such as non-acceptable codes as underlying cause of death. Such an observation strongly reinforces the hypothesis that a hybrid, expert system deep learning approach should improve the presented system's coding accuracy.

4.3.4 Practical application: Recoding the 2012 French overdose anomaly

The topic of overdose related death monitoring has recently drawn attention of public health agencies around the world, specifically in light of the opioid related sanitary crisis recently witnessed in the US. Causes of death data constitutes an information source of choice to investigate such topics. In France, the CépiDc database was used to assess the evolution of overdose related deaths from 2000 to 2015, by counting for each year the number of deaths associated with the following underlying causes:

- Opioid and Cannabis related disorders (ICD-10 codes beginning with F11 and F12),
- Cocaine, hallucinogen and other stimulant related disorders (F14 to F16),
- Other psychoactive substance related disorders (F19),
- Accidental poisoning by and exposure to narcotics and psychodysleptics, not elsewhere classified (X42),
- Intentional self-poisoning by and exposure to narcotics and psychodysleptics, not elsewhere classified (X62),

- Poisoning by and exposure to narcotics and psychodysleptics, not elsewhere classified, undetermined intent (Y12).

The resulting trajectory can be found in figure 4.3.8, and shows a significant decline in overdose related deaths in 2011 and 2012. Some explanations were found for this punctual reduction, such as decrease in heroin purity[84] and heroin overdose related deaths[85], in the same time period.

Although this punctual reduction can be at least partially explained by observed decreases in both heroin purity[84] and heroin overdose related deaths[85] in the same time period, confrontation with results obtained from an independent source, the DRAMES dataset, suggests another hypothesis. The DRAMES study constitutes a non-exhaustive inventory of overdose related deaths detected in French Legal Medicine Institutes. As a non-exhaustive database, its death count should not exceed the value obtained from the CépiDc database. As can be seen in figure 4.3.7, this logical assertion is true for all years from 2009 to 2013 but the notable exception of 2012. This discrepancy might be explained by a coding process deficiency, a hypothesis that can easily be verified by recoding every certificate from 2012, and comparing the number of overdose related deaths in both situations.

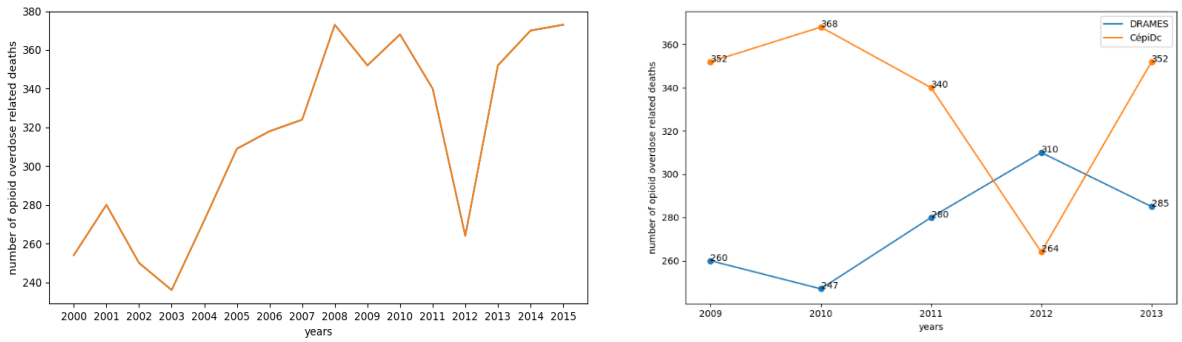


Fig. 4.3.7 Left: Evolution of overdose related deaths from 2000 to 2015 in France. The sudden decrease in 2012 appears anomalous. Right: Comparison with DRAMES data, a non-exhaustive, independent data source, which finds more deaths in 2012 than the exhaustive CépiDc database

The model derived in the previous experiment was used to recode every French death certificate from 2000 to 2015, with the year of coding set to 2015 to prevent any discrepancy related to coding rule

variation. The overdose related death were then selected from the predicted underlying cause of deaths following the aforementioned methodology.

The resulting curve can be seen in figure 4.3.8, alongside the official curve, and clearly shows a smoother decrease in opioid related deaths. The discrepancy with the DRAMES database, in addition, disappears when considering the recoded underlying causes of deaths.

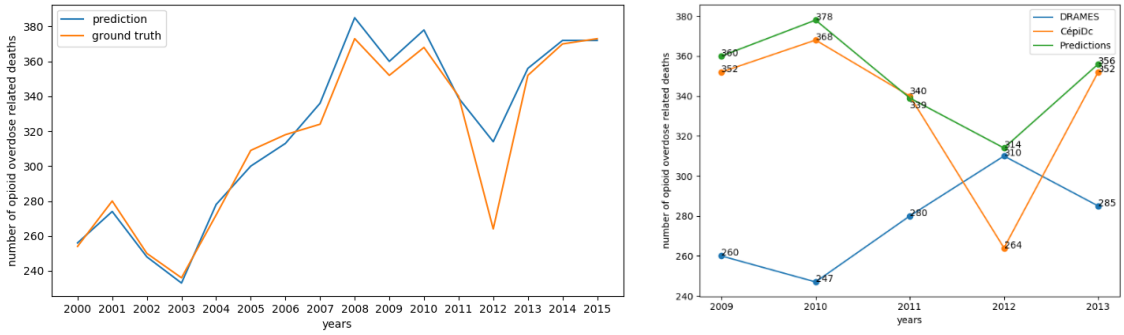


Fig. 4.3.8 Left: Evolution of opioid overdose related deaths from 2000 to 2015 in France either coded with Iris and human coders (orange) or with the proposed approach (blue). The 2012 gap, although still present, is much smoother when using predicted underlying causes. Right: Comparison with DRAMES data. The contradiction with the CépiDc database is entirely corrected with the predicted causes.

4.3.5 Conclusion

In this section was presented a formulation of the underlying cause of death coding from death certificate as a statistical modelling problem, who was then addressed with a deep artificial neural network, setting a new state of the art. The derived model's behaviour was thoroughly assessed following different approaches in order to identify potentially harmful biases and assess the potential of a hybrid approach mixing rule based decision system and statistical modelling. Although the proposed solution significantly outperforms any other existing automated coding approaches on French death certificates, the question of model transferability to other countries requires more

investigation. Indeed, the problem of distribution shift is well known in the machine learning community, and can significantly impair the model's quality[86].

The authors feel confident the model should perform with similar predictive power on other countries' death certificate with little to no supplementary effort necessary, even though this claim requires some experimental validation, unrealisable without international cooperation. To conclude, this chapter shows that deep artificial neural networks are perfectly suited to the analysis of electronic health records, and can learn directly from voluminous dataset a complex set of medical rules, without any explicit prior knowledge. Although not entirely free from mistakes, the derived algorithm constitutes a powerful decision making tool able to handled structured, medical data with unprecedented performances, and we strongly believe that the methods developed in this chapter are highly reusable in a variety of settings related to epidemiology, biostatistics, and the medical sciences in general.

4.4 On the comparability of international cause of death statistics: A deep artificial neural network based study

4.4.1 Introduction

As aforementioned, the current need for human contribution in the production of mortality statistics can lead them to be subject to distributional shift across both countries and years, which in turn raises the question of whether, and how much, are cause of death statistics comparable at the international level. However, the predictive model introduced in the previous chapter leads to an autonomous coding system with performances comparable to human coders in term of performance. Although they still require further investigation to be proposed as part of the data production process included or in interaction with the well-established Iris software, their predictive power might be able to provide some insight on this comparability question. This is exactly what this chapter investigates, by fitting one deep learning based model for the prediction of underlying cause of death from death certificates

on a massive and multi-country set of records, and trying to isolate coding patterns specific to each country included in the experiment from the derived model.

4.4.2 Material and method

4.4.2.1 Material

The dataset used for the experiments presented in this part consists of death certificates from 4 countries, namely England, the United States of America (U.S.A.), France and Italy, and their associated causes of death, representing over 60 million training examples. As the availability and number of these documents varies across countries, each of the datasets at the national scale level have some specificities:

- The English dataset consists of all English and Welsh death certificates from years 2005 to 2018, with the exception of certificates that were certified by a coroner, for a total of 5.1 million observations, 280 000 of which were reserved for the creation of a validation and test dataset for hyper-parameter searching and performance evaluation, respectively,
- The Italian dataset consists of all Italy's death certificates from years 2014 to 2016, with the exception of certificates related to external causes of deaths (chapter XX of the ICD-10 classification, related to violent or accidental death among other things), for a total of 1.8 million observations, 120 000 of which were reserved for the creation of a validation and test dataset,
- The American dataset consists of all U.S. death certificates from years 2000 to 2017, for a total of 45.5 million observations, 900 000 of which were reserved for the creation of a validation and test dataset,
- The French dataset consists of all French death certificates from years 2000 to 2016, for a total of 9.3 million observations, 340 000 of which were reserved for the creation of a validation and test dataset.

These documents record various information about their subjects, with varying predictive power with regard to the underlying cause of death. The following part aims to derive a deep neural network based predictive model explaining the underlying cause of death from the information contained within death certificates:

$$P(UCD|DC) = f(DC) \quad (4.4.1),$$

with:

- DC the information contained in a death certificate
- UCD its corresponding underlying cause of death
- f a neural network based predictive function

In order to model the underlying cause of death from these information, the following items were selected as explanatory variables:

- The causal chain of events leading death
- age
- gender
- year of death
- country of origin

4.4.2.2 Method

The variables' formatting and model definition follow the exact same methodology as was done in our previous study with the addition of one variable in the model, the certificate's country of origin. This

variable will be considered as a qualitative variable with 4 distinct states (one for each country included in the study), which will be treated in the same exact manner as all other miscellaneous variables, meaning all these categorical variables are projected into a dense highly dimensional space, mean aggregated, and then added element wise to all ICD-10 codes present in the certificate's causal chain of death's own embeddings.

This additional variable constitutes the cornerstone of the experiment presented in this chapter, the main rationale behind its inclusion being that if mortality statistics were to be perfectly comparable across countries, this variable should not impact the model's predictions. As an example, for a given certificate, setting this variable's state to "Italy" or "France" should result in the exact same model prediction. Consequently, making this variable state vary for all certificates included in the test set and investigating whether they lead to variation in the model's prediction might yield some leads as to whether death statistics are actually comparable across countries, at least in an exploratory fashion. This can be interpreted as an experiment where human coders from a given country were given certificates from another country, for an inter-rater agreement study. Due to the time consuming nature of the coding process, this approach would be unrealistic for human coders, but can be done fairly simply using a neural network that learnt each country's way of coding.

As a consequence, after model fitting, test sets from all countries were predicted with the country of origin's information being set to both the proper one, and all other countries included in the experiment, resulting in 4 distinct predictions for each death certificate. The model's performances were then evaluated for all these different predictions separately, in order to assess whether variation in the country of origin variable results in a variation in predictive performances. To sum up, for a given country, after model fitting:

- The model predicted the country's test dataset its country of origin variable set to its correct value,

- The model predicted the country's test dataset its country of origin variable set to all other countries,
- All the resulting predictions' performances were estimated in term of accuracy at the 4 characters ICD-10 code level (with 95% confidence intervals being computed using bootstrap), leading to 4 distinct performance evaluations that would be expected to be equal, were mortality statistics perfectly comparable at the international scale.

In order to provide the reader with a finer grained picture of country of origin variable's impact on the model's predictive power, its accuracy was also assessed at the Eurostat causes of death shortlist level in two distinct approaches:

- Exact code match (at the 4 characters ICD-10 code level, as was done for the model's overall accuracy)
- Eurostat shortlist level match (were an incorrect prediction at the ICD-10 level might become correct, if both the certificate's actual cause of death and the model's prediction lie in the same Eurostat item). This estimation is performed in order to assess whether prediction errors are mostly due to subtle, 4th character changes (for instance predicting E106, "Type 1 diabetes mellitus with other specified complications" instead of E108 "Type 1 diabetes mellitus with unspecified complications"), or to bigger, more systemic differences in coding

4.4.2.3 Model fitting

The model was fit on the entire dataset following the same method as was presented in part 4.3, with one notable difference. As the significant unbalance in dataset sample size between the countries included in the experiment (ranging from 1.8 million certificates for Italy to 45 million certificates emanating from the United States of America) might lead to a risk of potential under-fitting for countries associated to smaller sample sized, a simple random up-sampling strategy was applied to the

dataset in order to ensure balance between different country classes. In addition, a hyper-parameter search was conducted using random searching guided by the model's accuracy on a preliminary excluded validation set dedicated to this task.

4.4.3 Results

The experiment results can be found in table 4.4.1. In term of overall accuracy, the English and Welsh dataset is best predicted, followed by the American, Italian and French datasets. The reader might notice that the performances obtained for France are significantly lower than what was reported in chapter 4.3. This might be explained by the fact that the model cannot only focus its predictive power on French certificates solely, and might be solved by using a bigger model. This was unfortunately not practical due to a lack of computational resources.

	Coded as France	Coded as Italy	Coded as U.S.A.	Coded as England
From France	97.3 [97.2, 97.4]	90.6 [90.4, 90.7]	94.1 [94.0, 94.2]	94.4 [94.3, 94.5]
From Italy	90.0 [89.8, 90.3]	97.6 [97.5, 97.7]	91.1 [90.9, 91.3]	89.7 [89.4, 90.0]
From the U.S.A.	94.8 [94.7, 94.9]	90.5 [90.4, 90.6]	98.6 [98.6, 98.6]	96.8 [96.7, 96.8]
From England	96.9 [96.8, 97.0]	96.3 [96.2, 96.4]	97.0 [96.9, 97.1]	99.1 [99.1, 99.2]

Table 4.4.1: Model performances for each country, in term of accuracy, with the country of origin country set to all included countries. The diagonal gives the actual model performances for each country

In term of sensibility to changes in the state of the country of origin variable, a decrease in predictive performances can be observed for all countries, the biggest ones being observed for Italian certificates, with observed decreases of up to 8% when they are coded with the country of origin variable set to England. For every country apart from Italy, the biggest drop in predictive power can be observed when their death certificates are fed to the model as if they were originating from Italy.

The Eurostat level performance evaluation can be seen in figures 1, 2, 3 and 4 for French, English, American and Italian certificates, respectively. For better readability, only Eurostat items where at least one of the four prediction set yielded an accuracy measurement lesser than 90% are displayed. The entire Eurostat level accuracy results can however be found in the annex.

Category	France	US	Italy	England	Prevalence (%)
1.2 AIDS (HIV-disease)	0.784	0.759	0.784	0.784	0.136
1.3 Viral hepatitis	0.929	0.898	0.898	0.893	0.132
3. Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism	0.875	0.891	0.882	0.881	0.404
4.1 Diabetes mellitus	0.911	0.455	0.449	0.439	2.106
5.1 Dementia	0.984	0.896	0.935	0.884	2.375
5.4 Other mental and behavioural disorders	0.918	0.875	0.897	0.884	0.546
8.2 Pneumonia	0.958	0.921	0.895	0.920	2.024
8.3.2 Other chronic lower respiratory diseases	0.970	0.917	0.871	0.959	1.507
10. Diseases of the skin and subcutaneous tissue	0.945	0.894	0.900	0.933	0.299
11.1 Rheumatoid arthritis and osteoarthrosis	0.945	0.907	0.901	0.890	0.107
11.2 Other diseases of the musculoskeletal system/connective tissue	0.922	0.892	0.879	0.898	0.576
13. Complications of pregnancy, childbirth and puerperium	0.706	0.765	0.588	0.706	0.010
14. Certain conditions originating in the perinatal period	0.756	0.722	0.718	0.724	0.282
15. Congenital malformations and chromosomal abnormalities	0.908	0.862	0.856	0.871	0.282
17.1.1 Transport accidents	0.987	0.986	0.051	0.979	0.938
17.1.2 Accidental falls	0.972	0.959	0.355	0.950	1.097
17.1.3 Drowning and accidental submersion	0.966	0.959	0.810	0.959	0.174
17.1.4 Accidental poisoning	0.926	0.905	0.623	0.903	0.278
17.1.5 Other accidents	0.963	0.894	0.267	0.923	2.339
17.2 Suicide and intentional self-harm	0.990	0.988	0.801	0.988	1.859
17.3 Homicide, assault	0.900	0.894	0.381	0.875	0.094
17.4 Events of undetermined intent	0.901	0.895	0.361	0.890	0.112
17.5 Other external causes of injury and poisoning	0.812	0.436	0.146	0.470	0.213

Category	France	US	Italy	England	Prevalence (%)
3. Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism	0.885	0.905	0.897	0.894	0.404
5.1 Dementia	0.986	0.904	0.947	0.888	2.375
5.4 Other mental and behavioural disorders	0.936	0.896	0.921	0.906	0.546
8.2 Pneumonia	0.959	0.922	0.896	0.922	2.024
13. Complications of pregnancy, childbirth and puerperium	0.765	0.882	0.647	0.765	0.010
15. Congenital malformations and chromosomal abnormalities	0.929	0.885	0.885	0.898	0.282
17.1.1 Transport accidents	0.991	0.990	0.051	0.982	0.938
17.1.2 Accidental falls	0.973	0.960	0.355	0.951	1.097
17.1.3 Drowning and accidental submersion	0.966	0.959	0.810	0.959	0.174
17.1.4 Accidental poisoning	0.934	0.915	0.629	0.911	0.278
17.1.5 Other accidents	0.974	0.902	0.268	0.948	2.339
17.2 Suicide and intentional self-harm	0.992	0.991	0.805	0.990	1.859
17.3 Homicide, assault	0.912	0.900	0.394	0.887	0.094
17.4 Events of undetermined intent	0.901	0.895	0.361	0.890	0.112
17.5 Other external causes of injury and poisoning	0.820	0.448	0.155	0.478	0.213

Fig. 4.4.1 French certificates, Top: Code level accuracy per Eurostat items, Bottom: Eurostat shortlist level accuracy per Eurostat items

Category	France	US	Italy	England	Prevalence (%)
1.2 AIDS (HIV-disease)	0.775	0.775	0.825	0.875	0.029
3. Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism	0.860	0.892	0.871	0.957	0.199
4.1 Diabetes mellitus	0.589	0.907	0.841	0.978	1.155
5.2 Alcohol abuse (including alcoholic psychosis)	0.909	0.727	0.773	0.909	0.016
8.3.2 Other chronic lower respiratory diseases	0.981	0.894	0.831	0.992	5.816
10. Diseases of the skin and subcutaneous tissue	0.624	0.921	0.893	0.977	0.369
12.2 Other diseases of the genitourinary system	0.973	0.897	0.898	0.983	1.430
15. Congenital malformations and chromosomal abnormalities	0.919	0.891	0.875	0.931	0.177
17.1.2 Accidental falls	0.909	0.939	0.485	0.970	0.024
17.1.4 Accidental poisoning	0.000	0.000	0.000	0.000	0.001
17.1.5 Other accidents	0.671	0.709	0.025	0.990	0.282
17.5 Other external causes of injury and poisoning	0.333	0.410	0.154	0.744	0.028

Category	France	US	Italy	England	Prevalence (%)
3. Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism	0.860	0.892	0.871	0.957	0.199
5.2 Alcohol abuse (including alcoholic psychosis)	0.909	0.727	0.773	0.909	0.016
10. Diseases of the skin and subcutaneous tissue	0.640	0.926	0.899	0.979	0.369
12.2 Other diseases of the genitourinary system	0.973	0.898	0.898	0.984	1.430
15. Congenital malformations and chromosomal abnormalities	0.919	0.891	0.875	0.931	0.177
17.1.2 Accidental falls	0.909	0.939	0.485	0.970	0.024
17.1.4 Accidental poisoning	0.000	0.000	0.000	0.000	0.001
17.1.5 Other accidents	0.684	0.868	0.025	0.992	0.282
17.5 Other external causes of injury and poisoning	0.333	0.410	0.154	0.744	0.028

Fig. 4.4.2 English certificates, Top: Code level accuracy per Eurostat items, Bottom: Eurostat shortlist level accuracy per Eurostat items

Category	France	US	Italy	England	Prevalence (%)
1.2 AIDS (HIV-disease)	0.779	0.888	0.848	0.817	0.424
3. Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism	0.869	0.933	0.905	0.897	0.400
4.1 Diabetes mellitus	0.417	0.988	0.829	0.926	2.905
5.3 Drug dependence, toxicomania	0.883	0.907	0.922	0.898	0.091
5.4 Other mental and behavioural disorders	0.877	0.942	0.821	0.892	0.186
10. Diseases of the skin and subcutaneous tissue	0.586	0.956	0.883	0.915	0.160
11.1 Rheumatoid arthritis and osteoarthrosis	0.899	0.938	0.915	0.897	0.136
11.2 Other diseases of the musculoskeletal system/connective tissue	0.848	0.927	0.899	0.903	0.442
13. Complications of pregnancy, childbirth and puerperium	0.692	0.713	0.706	0.713	0.032
16.3 Other symptoms, signs, ill-defined causes	0.957	0.990	0.956	0.819	0.733
17.1.1 Transport accidents	0.979	0.982	0.171	0.973	1.740
17.1.2 Accidental falls	0.979	0.994	0.026	0.961	0.974
17.1.3 Drowning and accidental submersion	1.000	0.998	0.000	0.994	0.148
17.1.4 Accidental poisoning	0.990	0.993	0.152	0.987	1.299
17.1.5 Other accidents	0.960	0.971	0.068	0.890	0.831
17.2 Suicide and intentional self-harm	0.998	0.998	0.046	0.998	1.484
17.3 Homicide, assault	0.995	0.997	0.211	0.993	0.681
17.4 Events of undetermined intent	0.986	0.993	0.236	0.971	0.185
17.5 Other external causes of injury and poisoning	0.745	0.679	0.138	0.618	0.135

Category	France	US	Italy	England	Prevalence (%)
10. Diseases of the skin and subcutaneous tissue	0.589	0.957	0.889	0.918	0.160
13. Complications of pregnancy, childbirth and puerperium	0.895	0.874	0.874	0.881	0.032
16.3 Other symptoms, signs, ill-defined causes	0.958	0.991	0.958	0.821	0.733
17.1.1 Transport accidents	0.981	0.994	0.171	0.978	1.740
17.1.2 Accidental falls	0.979	0.994	0.026	0.961	0.974
17.1.3 Drowning and accidental submersion	1.000	0.998	0.000	0.994	0.148
17.1.4 Accidental poisoning	0.992	0.995	0.152	0.989	1.299
17.1.5 Other accidents	0.966	0.972	0.080	0.961	0.831
17.2 Suicide and intentional self-harm	1.000	1.000	0.047	0.999	1.484
17.3 Homicide, assault	0.996	0.997	0.211	0.994	0.681
17.4 Events of undetermined intent	0.992	0.996	0.237	0.976	0.185
17.5 Other external causes of injury and poisoning	0.745	0.682	0.138	0.618	0.135

Fig. 4.4.3 U.S. Certificates, Top: Code level accuracy per Eurostat items, Bottom: Eurostat shortlist level accuracy per Eurostat items

Category	France	US	Italy	England	Prevalence (%)
1.1 Tuberculosis	0.852	0.852	0.926	0.889	0.045
1.2 AIDS (HIV-disease)	0.441	0.746	0.695	0.356	0.098
2.1.22 Other malignant neoplasms	0.785	0.945	0.977	0.827	3.658
4.1 Diabetes mellitus	0.809	0.690	0.977	0.678	3.433
4.2 Other endocrine, nutritional and metabolic diseases	0.865	0.906	0.951	0.883	1.052
5.2 Alcohol abuse (including alcoholic psychosis)	0.913	0.826	0.913	0.913	0.038
5.3 Drug dependence, toxicomania	0.800	0.800	0.800	0.800	0.017
5.4 Other mental and behavioural disorders	0.826	0.870	0.948	0.861	0.192
6.2 Alzheimer's disease	0.889	0.878	0.992	0.881	1.828
6.3 Other diseases of the nervous system and the sense organs	0.897	0.915	0.927	0.891	1.560
8.3.2 Other chronic lower respiratory diseases	0.877	0.831	0.986	0.870	3.543
8.4 Other diseases of the respiratory system	0.863	0.957	0.973	0.897	2.005
11.2 Other diseases of the musculoskeletal system/connective tissue	0.828	0.882	0.912	0.874	0.397
12.1 Diseases of kidney and ureter	0.798	0.853	0.965	0.828	1.617
13. Complications of pregnancy, childbirth and puerperium	0.500	0.500	0.500	0.500	0.003
16.3 Other symptoms, signs, ill-defined causes	0.974	0.980	0.998	0.728	1.703

Category	France	US	Italy	England	Prevalence (%)
1.2 AIDS (HIV-disease)	0.695	0.966	0.932	0.644	0.098
2.1.22 Other malignant neoplasms	0.881	0.986	0.986	0.881	3.658
5.2 Alcohol abuse (including alcoholic psychosis)	0.913	0.826	0.913	0.913	0.038
11.2 Other diseases of the musculoskeletal system/connective tissue	0.836	0.882	0.916	0.887	0.397
12.1 Diseases of kidney and ureter	0.811	0.864	0.972	0.838	1.617
13. Complications of pregnancy, childbirth and puerperium	0.500	0.500	0.500	0.500	0.003
16.3 Other symptoms, signs, ill-defined causes	0.989	0.995	0.999	0.742	1.703

Fig. 4.4.4 Italian Certificates, Top: Code level accuracy per Eurostat items, Bottom: Eurostat shortlist level accuracy per Eurostat items

Several point of interest can be observed from these figures, the main ones being:

- Alcohol abuse related death for English and Welsh certificates, both at the code and Eurostat shortlist level. Curiously, the gap in prediction power only concerns Italian and American predictions, with the French remaining consistent with the English ones
- AIDS related deaths at the code level for all countries, and at the Eurostat level for Italian certificates, where a decrease of accuracy from 93.2% to 64.4% can be observed from Italian to English predictions. Curiously, for this particular Eurostat chapter, the accuracy on Italian chapter is higher for American predictions
- External causes of death for all countries where they were made available
- Diabetes at the code level for French certificates, although this disagreement disappears at the Eurostat level (the worst accuracy then measured being of 95.8% for Italian predictions, as can be seen in the annex)

4.4.4 Discussion

As mentioned before, the biggest drop in predictive power was obtained when recoding certificates as if they were originating from Italy. However, this might have been at least partially explained due to the fact that the Italian dataset was only comprised of certificates from years 2014 to 2016. As a consequence, the model has technically never seen any Italian certificates coded in 2005, while it is artificially given such certificates when predicting France as Italy, for instance. However, when assessing the model's accuracy on death certificates from years to 2014 to 2016 coded as if they were originating from Italy, the accuracy only drops lower (96.1%, 87.8% and 89.2% for English, French and American certificates, respectively), strongly suggesting that this hypothesis does not suffice to explain this phenomenon. Another explanation might lie in the fact that the Italian dataset did not contain any certificates related to external causes of death, which is strongly supported by the fact that when excluding external causes of death related certificates from the English, American and French datasets

and recoding them as if they were originating from Italy, the model's accuracy raises to 96.6%, 96.9% and 94.3% respectively.

The fact that the lowest actual model accuracy is observed for French certificates rather than Italian ones can be explained by this fact too. Indeed, as shown in figure 4.4.1, certificates related to external causes of death are associated with poor accuracy, potentially resulting in artificially higher results for Italian certificates.

Finally, the actual meaning of these variations in predictive power needs to be interpreted carefully. Indeed, even though the model strongly suggests that as the country of origin variable has an impact on predictions, at least to some extent, differences in the coding process exists, the explanation of these differences remains unclear. One could for instance ask whether they come from an actual difference in coding practices, or from model artefacts such as sensibility to a distribution shift. As an example, firearm related deaths being fairly rarer in France than in the US, it might be reasonable to hypothesize that French predictions of US firearm related death certificates are more often mispredicted, due to the sheer rarity of the phenomenon in France. In addition, even in cases where this variation in prediction were to be related to actual differences in the coding process, further investigation would be required in order to properly identify their nature.

4.4.5 Conclusion

In this chapter was introduced a novel, deep artificial neural network based approach to investigate the comparability of cause of death statistics at the national scale. Its application on the 4 countries included in the experiment shows that these statistics are globally fairly comparable, although some cases of potential differences between international coding processes were identified. These results offer interesting leads, but remains exploratory by design, and require further investigation in order to be better understood.

Finally, we would like to invite any other country that would be interested in assessing their own mortality statistics' comparability at the international scale to share a dataset of their own. Indeed, the results exposed in this chapter are but a mere snapshot of the rich complexity of mortality statistics. As aforementioned, the comparability of these statistics is one of their most crucial aspect, and the method presented in this chapter is among the first approaches that enables to formally assess it in a unified manner. Considering the usefulness of reproducing and generalize that kind of treatment, an international body (WHO, Eurostat…) would seem adapted to set an infrastructure to manage international data collection and treatment.

5 Conclusion

During this thesis have been introduced several examples of how deep artificial neural network based models can be applied to electronic health databases, oftentimes with results showing a significant improvement upon the previous state of the art. This models have immediate real world applications, typically in accelerating the production of electronic medical databases (for instance with automated coding of medical acts), whose interest has already been shown with the use of the transformer based medical entity recognition from natural language model, which was used during the first lock down in France to produce real time comorbidity statistics for COVID-19 related deaths, and more indirect ones, as seen with the underlying cause of death predictive model's use to check for potential coding anomalies, or to assess the comparability of mortality statistics at the international level. In addition, these methods in themselves are not specific to the dataset they were used on, nor the predictive modelling problem they were applied to. As such, they make promising leads for application in other electronic medical databases, such as the French "Système National des Données de Santé". On the other hand, the fact that artificial neural network need not be the massive, "black box" type models they have the reputation to be, was partially demonstrated with the introduction of an ordinal

regression method based on recurrent neural networks. Even though a substantial amount of work is still required to better understand the theory behind it, this type of hybrid, linear/neural model might constitute a promising lead toward efficient neural networks models with interpretable parameters, which is still one of their major drawback when applied to fields such as biostatistics and epidemiology. In addition to this lack of interpretability, several challenges still remain in order to see a more widespread use of deep learning methods in electronic health databases. First, although the CépiDc database constitutes a perfectly valid example of such a database, its volumetry remains small in comparison with databases such as the SNDS, which is several orders of magnitude bigger. This phenomenon constitutes a double edged sword. The sample size and sheer complexity of these massive database should allow for the implementation of huge neural network models on a rich variety of different problematics, but with a significant increase of complexity, in terms of logistics, data management, and computational infrastructures. All the experiment presented in this thesis were performed with a single multi-GPU machine, with the entire investigated datasets on disk. Such a feat is unthinkable on a database such as the SNDS, where even the task of shaping the data into a "machine-learning friendly" format constitutes a major technological achievement that ought to require entire teams of specialists from several fields. Besides, the data's quality itself is of paramount importance in itself, and can vary from case to case, as was showcased with the opioid overdose anomaly example. As a consequence, brutally applying modern machine learning techniques to such complex databases, without extensive understanding of their potential biases' and limits, can result in either poor results, or, in the worst case, in erroneous conclusions, without much means of validation.

Focusing on the sole task of pattern analysis, which, in its most basic essence, is what deep learning is, allows practitioner to build outstandingly powerful models with superhuman complexity. However, this is, at least from my observations, done at the price of reducing all the inherent complexity of the latent world to negligible noise. This is perfectly fine in most fields in artificial intelligence, where most

tasks are dedicated to reproducing human cognition. As an example, evaluating that all the information necessary to predict whether a dog is present in a picture is contained in a dataset of pictures of cats and dogs is trivial for the average human being. On the contrary, I find that epidemiology is usually all about understanding what the data means in the context of everything that's already known outside the dataset (what Bayesian statisticians would call the prior, in a sense), whether it applies to the data's quality and reliability to external biases. In my opinion, the fundamental root of the current conflict between modern data scientists and epidemiologists lies in a lack of dialogue between the two sides concerning this issue. I've seen machine learning practitioners feed a dataset to a random forest without taking a second to think about its limits and potential biases, only to be quietly laughed at by epidemiologists, amused by the contrast between the tool's sophistications and the relatively poor interest of their results. On the other hand, I've seen public health practitioner dismiss entire protocols that might have led to interesting results, just based on an (sometimes) underserved mistrust of "blackboxy" tools. However, I strongly believe having specialists of both fields working together while understanding each other could lead to some outstanding results, and hope that this thesis can be a building block toward that goal.

6 Bibliography

1. JMIR Medical Informatics - A Deep Artificial Neural Network–Based Model for Prediction of Underlying Cause of Death From Death Certificates: Algorithm Development and Validation. https://medinform.jmir.org/2020/4/e17125/.
2. Falissard, L. *et al.* Neural translation and automated recognition of ICD10 medical entities from natural language. *ArXiv200413839 Cs Stat* (2020).
3. Falissard, L., Bounebache, K. & Rey, G. Learning a binary search with a recurrent neural network. A novel approach to ordinal regression analysis. *ArXiv210102609 Cs Stat* (2021).
4. Hernán, M. A., Hsu, J. & Healy, B. A Second Chance to Get Causal Inference Right: A Classification of Data Science Tasks. *CHANCE* **32**, 42–49 (2019).
5. Some Studies in Machine Learning Using the Game of Checkers - IBM Journals & Magazine. https://ieeexplore.ieee.org/document/5392560/.
6. Mitchell, T. M. *Machine Learning*. (McGraw-Hill, Inc., 1997).
7. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning*. (MIT Press, 2016).
8. Mohri, M., Rostamizadeh, A. & Talwalkar, A. *Foundations of Machine Learning*. (The MIT Press, 2012).
9. Boser, B. E., Guyon, I. M. & Vapnik, V. N. A Training Algorithm for Optimal Margin Classifiers. in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* 144–152 (ACM, 1992). doi:10.1145/130385.130401.
10. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
11. Statistical models theory and practice 2nd edition | Statistical theory and methods. *Cambridge University Press* http://www.cambridge.org/gb/academic/subjects/statistics-probability/statistical-theory-and-methods/statistical-models-theory-and-practice-2nd-edition.
12. Cox, D. R. The Regression Analysis of Binary Sequences. *J. R. Stat. Soc. Ser. B Methodol.* **20**, 215–242 (1958).

13. Hinton, G. E. *Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair*.
14. Rectifier Nonlinearities Improve Neural Network Acoustic Models - Semantic Scholar. /paper/Rectifier-Nonlinearities-Improve-Neural-Network-Maas-Hannun/367f2c63a6f6a10b3b64b8729d601e69337ee3cc.
15. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).
16. Mozer, M. A Focused Backpropagation Algorithm for Temporal Pattern Recognition. *Complex Syst.* **3**, (1995).
17. Abiodun, O. I. *et al.* State-of-the-art in artificial neural network applications: A survey. *Heliyon* **4**, (2018).
18. Long Short-Term Memory | Neural Computation | MIT Press Journals. http://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735.
19. Cho, K. *et al.* Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *ArXiv14061078 Cs Stat* (2014).
20. Zhang, W., Itoh, K., Tanida, J. & Ichioka, Y. Parallel distributed processing model with local space-invariant interconnections and its optical architecture. *Appl. Opt.* **29**, 4790–4797 (1990).
21. Lecun, Y. & Bengio, Y. Convolutional networks for images, speech, and time-series. *Handb. Brain Theory Neural Netw.* (1995).
22. Oord, A. van den *et al.* WaveNet: A Generative Model for Raw Audio. *ArXiv160903499 Cs* (2016).
23. Dumoulin, V. & Visin, F. A guide to convolution arithmetic for deep learning. *ArXiv160307285 Cs Stat* (2016).
24. Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A. & Bengio, Y. Maxout Networks. *ArXiv13024389 Cs Stat* (2013).
25. Zeiler, M. D. & Fergus, R. Visualizing and Understanding Convolutional Networks. *ArXiv13112901 Cs* (2013).

26. Lea, C., Vidal, R., Reiter, A. & Hager, G. D. Temporal Convolutional Networks: A Unified Approach to Action Segmentation. *ArXiv160808242 Cs* (2016).
27. Hochreiter, S., Bengio, Y., Frasconi, P. & Schmidhuber, J. *Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies*. (2001).
28. Ripley, B. D. Pattern Recognition and Neural Networks by Brian D. Ripley. *Cambridge Core* /core/books/pattern-recognition-and-neural-networks/4E038249C9BAA06C8F4EE6F044D09C5C (1996) doi:10.1017/CBO9780511812651.
29. Kohavi, R. A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection. in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2* 1137–1143 (Morgan Kaufmann Publishers Inc., 1995).
30. Geman, S., Bienenstock, E. & Doursat, R. Neural Networks and the Bias/Variance Dilemma. *Neural Comput* **4**, 1–58 (1992).
31. Ng, A. Y. Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance. in *Proceedings of the Twenty-first International Conference on Machine Learning* 78- (ACM, 2004). doi:10.1145/1015330.1015435.
32. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **15**, 1929–1958 (2014).
33. Hoerl, A. E. & Kennard, R. W. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics* **12**, 55–67 (1970).
34. Regression Shrinkage and Selection via the Lasso on JSTOR. https://www.jstor.org/stable/2346178?seq=1#page_scan_tab_contents.
35. Glorot, X. & Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. in *PMLR* 249–256 (2010).

36. Bengio, Y., Lamblin, P., Popovici, D. & Larochelle, H. Greedy Layer-Wise Training of Deep Networks. in *Advances in Neural Information Processing Systems 19* (eds. Schölkopf, B., Platt, J. C. & Hoffman, T.) 153–160 (MIT Press, 2007).
37. LeCun, Y., Bottou, L., Orr, G. B. & Müller, K.-R. Efficient BackProp. in *Neural Networks: Tricks of the Trade* 9–50 (Springer, Berlin, Heidelberg, 1998). doi:10.1007/3-540-49430-8_2.
38. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. *ArXiv14126980 Cs* (2014).
39. Ioffe, S. & Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ArXiv150203167 Cs* (2015).
40. Hochreiter, S. & Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **9**, 1735–1780 (1997).
41. Vaswani, A. *et al.* Attention Is All You Need. *ArXiv170603762 Cs* (2017).
42. Gehring, J., Auli, M., Grangier, D., Yarats, D. & Dauphin, Y. N. Convolutional Sequence to Sequence Learning. *ArXiv170503122 Cs* (2017).
43. Turkoz, I., Fu, D.-J., Bossie, C. A., Sheehan, J. J. & Alphs, L. Relationship between the clinical global impression of severity for schizoaffective disorder scale and established mood scales for mania and depression. *J. Affect. Disord.* **150**, 17–22 (2013).
44. Winship, C. & Mare, R. D. Regression Models with Ordinal Variables. *Am. Sociol. Rev.* **49**, 512 (1984).
45. Pedregosa-Izquierdo, F. Feature extraction and supervised learning on fMRI : from practice to theory. (Université Pierre et Marie Curie - Paris VI, 2015).
46. A modification to the half-interval search (binary search) method | Proceedings of the 14th annual Southeast regional conference. https://dl.acm.org/doi/10.1145/503561.503582.
47. Sutskever, I., Vinyals, O. & Le, Q. V. Sequence to Sequence Learning with Neural Networks. in *Advances in Neural Information Processing Systems 27* (eds. Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D. & Weinberger, K. Q.) 3104–3112 (Curran Associates, Inc., 2014).

48. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
49. Press, O. & Wolf, L. Using the Output Embedding to Improve Language Models. *ArXiv160805859 Cs* (2016).
50. A Beginner's Guide to Attention Mechanisms and Memory Networks. *Pathmind* http://wiki.pathmind.com/attention-mechanism-memory-network.
51. Bai, S., Kolter, J. Z. & Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *ArXiv180301271 Cs* (2018).
52. Vaswani, A. *et al.* Attention is All you Need. *Adv. Neural Inf. Process. Syst.* **30**, 5998–6008 (2017).
53. Mahapatra, P. *et al.* Civil registration systems and vital statistics: successes and missed opportunities. *Lancet Lond. Engl.* **370**, 1653–1663 (2007).
54. Towards universal civil registration and vital statistics systems: the time is now - The Lancet. https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(15)60170-2/fulltext.
55. Civil registration and vital statistics: progress in the data revolution for counting and accountability - The Lancet. https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(15)60173-8/fulltext.
56. A global assessment of civil registration and vital statistics systems: monitoring data quality and progress - The Lancet. https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(15)60171-4/fulltext.
57. Brolan, C. E., Gouda, H. N., AbouZahr, C. & Lopez, A. D. Beyond health: five global policy metaphors for civil registration and vital statistics. *The Lancet* **389**, 1084–1085 (2017).
58. Principles and Recommendations for a Vital Statistics System, Revision 2 - Global Inventory of Statistical Standards. https://unstats.un.org/unsd/iiss/Principles-and-Recommendations-for-a-Vital-Statistics-System-Revision-2.ashx.
59. *International statistical classification of diseases and related health problems. Vol. 1: Tabular list*. (2004).

60. WHO ICD-10 The International Statistical Classification of Diseases and Related Health Problems 10th Revision - eHealth DSI Semantic Community - CEF Digital. https://ec.europa.eu/cefdigital/wiki/display/EHSEMANTIC/WHO+ICD-10+The+International+Statistical+Classification+of+Diseases+and+Related+Health+Problems+10th+Revision.
61. WHO | ICD-10 online versions. https://www.who.int/classifications/icd/icdonlineversions/en/.
62. Névéol, A. *et al.* CLEF eHealth 2017 Multilingual Information Extraction task Overview: ICD10 Coding of Death Certificates in English and French. in *Workshop of the Cross-Language Evaluation Forum* (ed. CEUR-WS) (CEUR-WS, 2017).
63. Névéol, A. *et al.* Clinical Information Extraction at the CLEF eHealth Evaluation lab 2016. *CEUR Workshop Proc.* **1609**, 28–42 (2016).
64. A Deep Learning Method for ICD-10 Coding of Free-Text Death Certificates | SpringerLink. https://link.springer.com/chapter/10.1007/978-3-319-65340-2_12.
65. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. Distributed Representations of Words and Phrases and Their Compositionality. in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* 3111–3119 (Curran Associates Inc., 2013).
66. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv181004805 Cs* (2019).
67. S˘eva, J., Sanger, M. & Leser, U. WBI at CLEF eHealth 2018 Task 1: Language-independent ICD-10 coding using multi-lingual embeddings and recurrent neural networks. 14.
68. Atutxa, A., de Ilarraza, A. D., Gojenola, K., Oronoz, M. & Perez-de-Viñaspre, O. Interpretable deep learning to map diagnostic texts to ICD-10 codes. *Int. J. Med. Inf.* **129**, 49–59 (2019).
69. Reys, A. D. *et al.* Predicting Multiple ICD-10 Codes from Brazilian-Portuguese Clinical Notes. in *Intelligent Systems* (eds. Cerri, R. & Prati, R. C.) 566–580 (Springer International Publishing, 2020). doi:10.1007/978-3-030-61377-8_39.

70. Cao, P. *et al.* Clinical-Coder: Assigning Interpretable ICD-10 Codes to Chinese Clinical Notes. in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations* 294–301 (Association for Computational Linguistics, 2020). doi:10.18653/v1/2020.acl-demos.33.
71. Schafer, H. & Friedrich, C. M. Multilingual ICD-10 Code Assignment with Transformer Architectures using MIMIC-III Discharge Summaries. 16.
72. Amin, S. *et al. MLT-DFKI at CLEF eHealth 2019: Multi-label Classification of ICD-10 Codes with BERT*. (2019).
73. Completing a medical certificate of cause of death (MCCD). *GOV.UK* https://www.gov.uk/government/publications/guidance-notes-for-completing-a-medical-certificate-of-cause-of-death.
74. So, D. R., Liang, C. & Le, Q. V. The Evolved Transformer. *ArXiv190111117 Cs Stat* (2019).
75. Papineni, K., Roukos, S., Ward, T. & Zhu, W.-J. Bleu: a Method for Automatic Evaluation of Machine Translation. in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* 311–318 (Association for Computational Linguistics, 2002). doi:10.3115/1073083.1073135.
76. Jackson, P. *Introduction to Expert Systems*. (Addison-Wesley Longman Publishing Co., Inc., 1998).
77. Lu, T. Using ACME (Automatic Classification of Medical Entry) software to monitor and improve the quality of cause of death statistics. *J. Epidemiol. Community Health* **57**, 470–471 (2003).
78. WHO | International Classification of Diseases, 11th Revision (ICD-11). *WHO* http://www.who.int/classifications/icd/en/.
79. Backpropagation Applied to Handwritten Zip Code Recognition - MITP Journals & Magazine. https://ieeexplore.ieee.org/document/6795724.
80. Schmidhuber, J. Deep Learning in Neural Networks: An Overview. *Neural Netw.* **61**, 85–117 (2015).

81. Zhang, W., Itoh, K., Tanida, J. & Ichioka, Y. Parallel distributed processing model with local space-invariant interconnections and its optical architecture. *Appl. Opt.* **29**, 4790–4797 (1990).
82. Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1* 1097–1105 (Curran Associates Inc., 2012).
83. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323**, 533 (1986).
84. Héroïne et autres opiacés - Synthèse des connaissances - OFDT. https://www.ofdt.fr/produits-et-addictions/de-z/heroine-et-autres-opiaces/.
85. Décès en relation avec l'abus de médicaments et de substances (DRAMES) - OFDT. https://www.ofdt.fr/statistiques-et-infographie/sources-statistiques/deces-en-relation-avec-l-abus-de-medicaments-et-de-substances-drames/#diff.
86. Quionero-Candela, J., Sugiyama, M., Schwaighofer, A. & Lawrence, N. D. *Dataset Shift in Machine Learning*. (The MIT Press, 2009).
87. Sennrich, R., Haddow, B. & Birch, A. Neural Machine Translation of Rare Words with Subword Units. in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* 1715–1725 (Association for Computational Linguistics, 2016). doi:10.18653/v1/P16-1162.
88. tensorflow/models. *GitHub* https://github.com/tensorflow/models.
89. [1706.03762] Attention Is All You Need. https://arxiv.org/abs/1706.03762.
90. Falissard, L. *et al.* A deep artificial neural network based model for underlying cause of death prediction from death certificates. *ArXiv190809712 Cs Stat* (2019).
91. Oord, A. van den *et al.* WaveNet: A Generative Model for Raw Audio. *ArXiv160903499 Cs* (2016).
92. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. Rethinking the Inception Architecture for Computer Vision. *ArXiv151200567 Cs* (2015).

93. Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv12070580 Cs* (2012).
94. Ba, J. L., Kiros, J. R. & Hinton, G. E. Layer Normalization. *ArXiv160706450 Cs Stat* (2016).
95. He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. *ArXiv151203385 Cs* (2015).

7 Annex

7.1 Learning a binary search with recurrent neural networks, a novel approach to ordinal regression

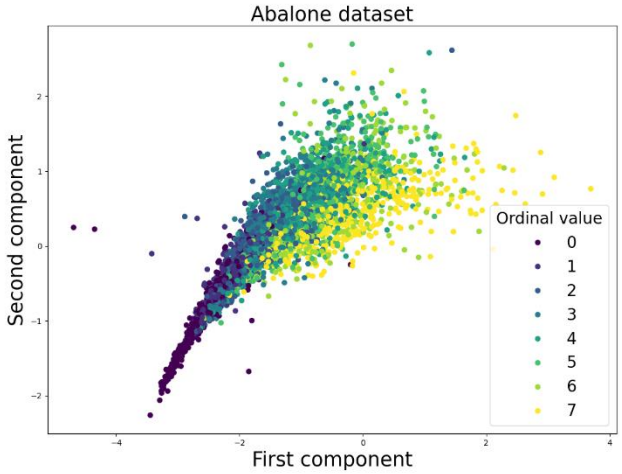


Fig. 7.1 Abalone dataset visualization derived from the proposed method

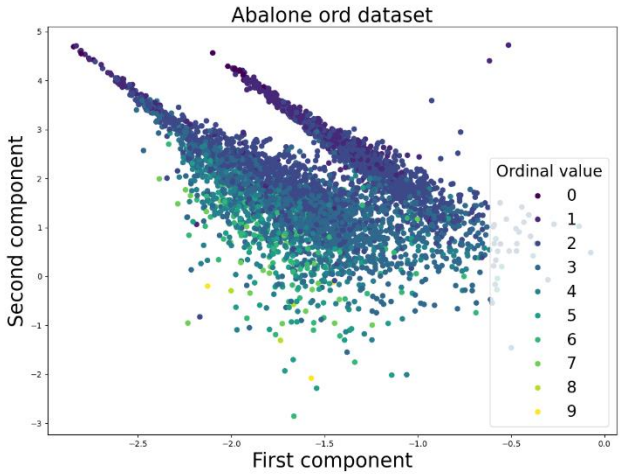


Fig. 7.2 Abalone ord dataset visualization derived from the proposed method

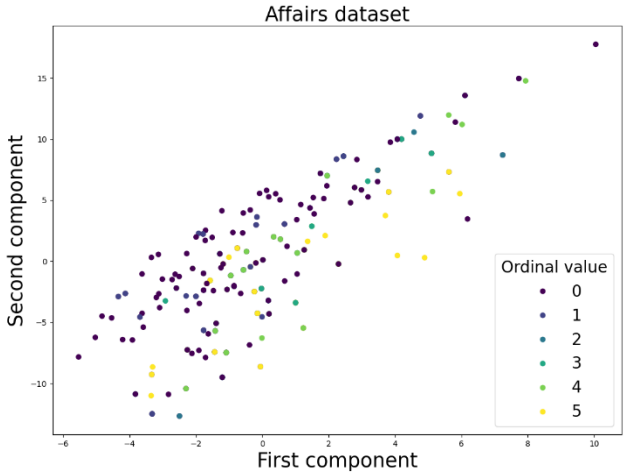


Fig. 7.3 Affairs dataset visualization derived from the proposed method

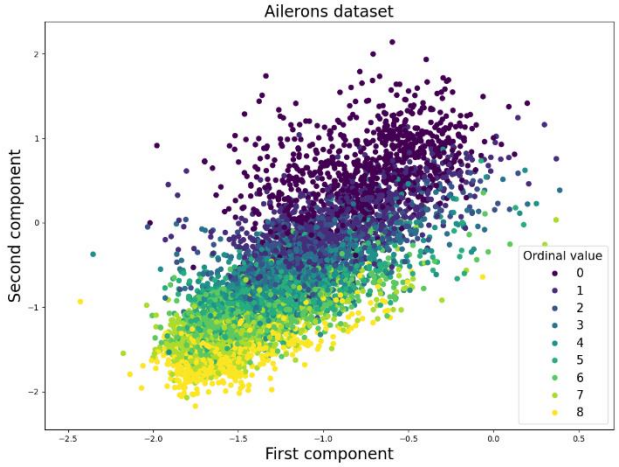


Fig. 7.4 Ailerons dataset visualization derived from the proposed method

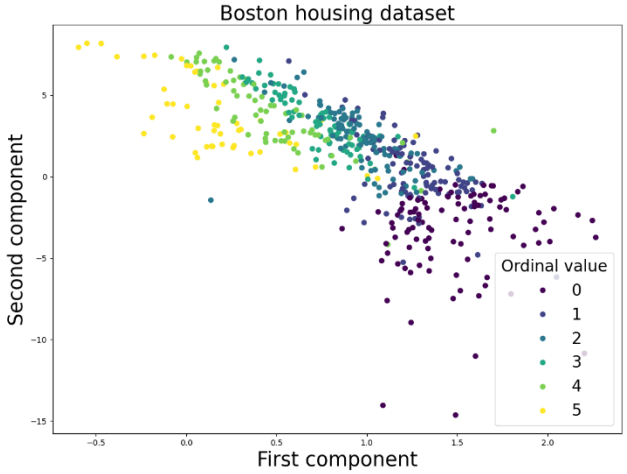


Fig. 7.5 Boston housing dataset visualization derived from the proposed method

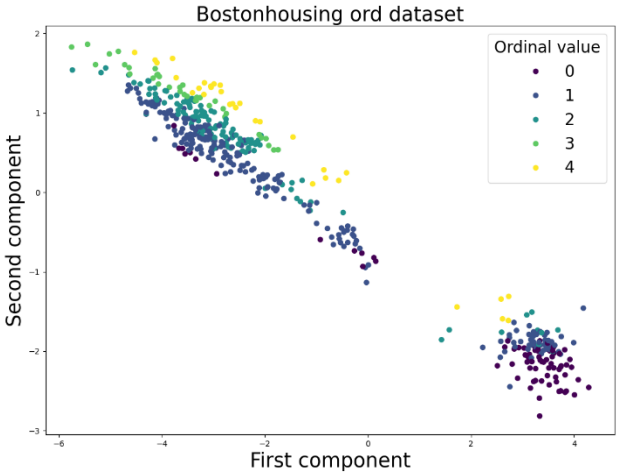


Fig. 7.6 Bostonhousing ord dataset visualization derived from the proposed method

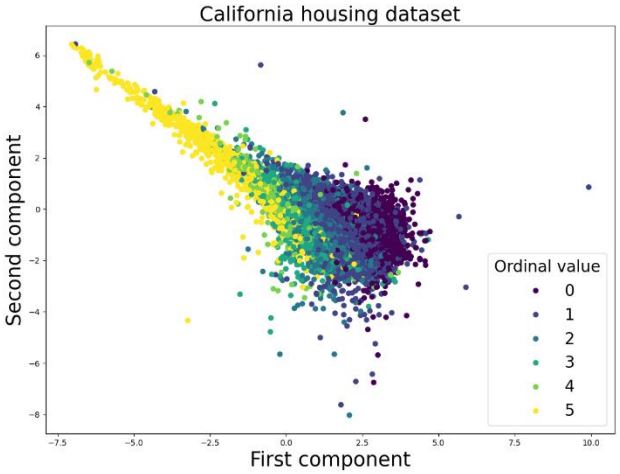


Fig. 7.7 California housing dataset visualization derived from the proposed method

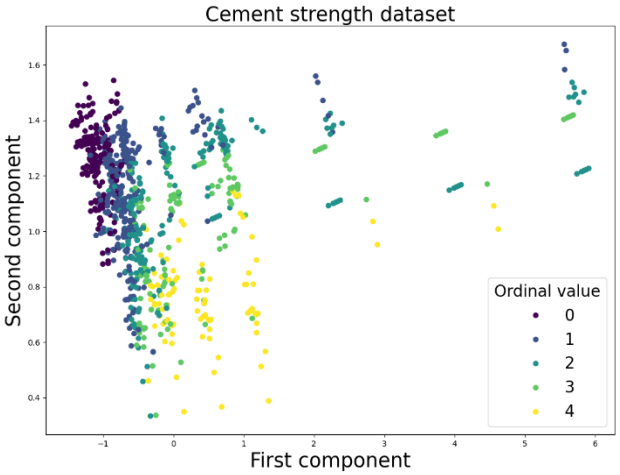


Fig. 7.8 Cement strength dataset visualization derived from the proposed method

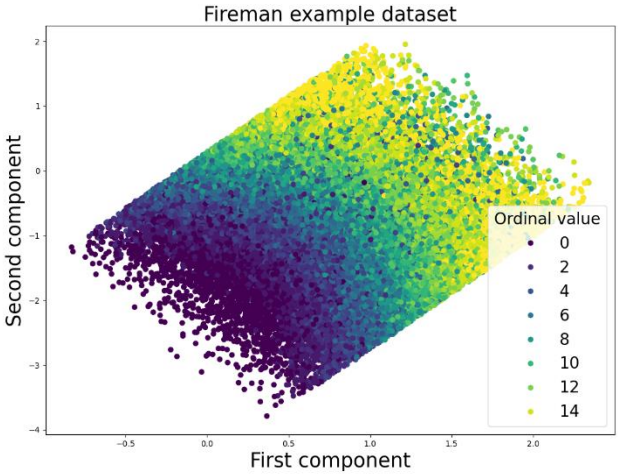


Fig. 7.9 Fireman example dataset visualization derived from the proposed method

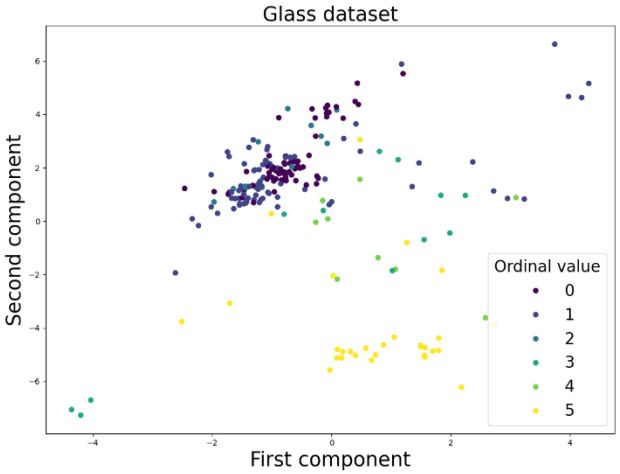


Fig. 7.10 Glass dataset visualization derived from the proposed method

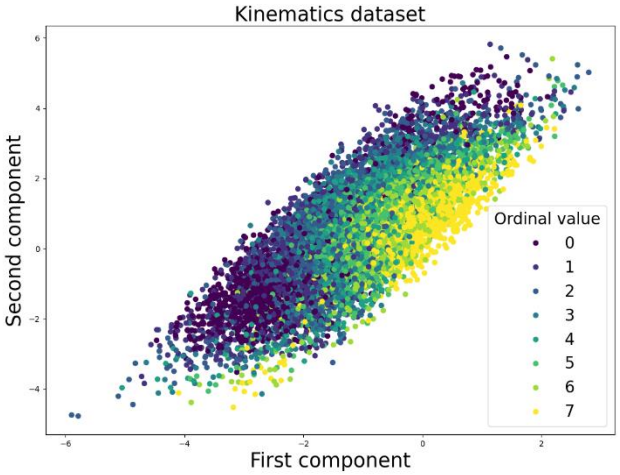


Fig. 7.11 Kinematics dataset visualization derived from the proposed method

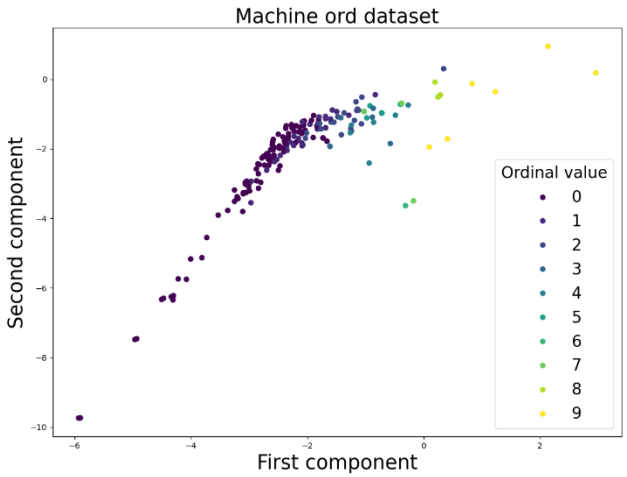


Fig. 7.12 Machine ord dataset visualization derived from the proposed method

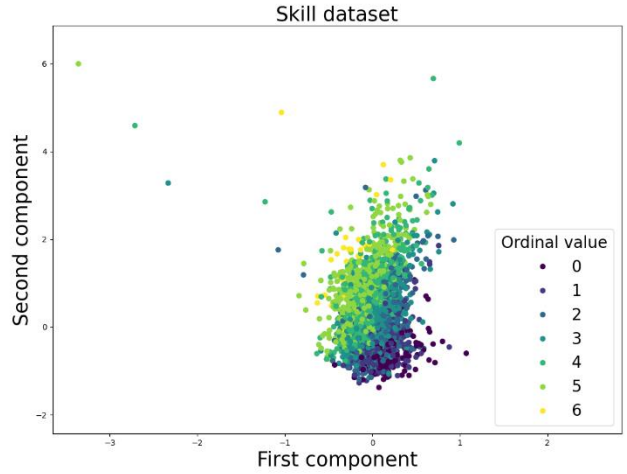


Fig. 7.13 Skill dataset visualization derived from the proposed method

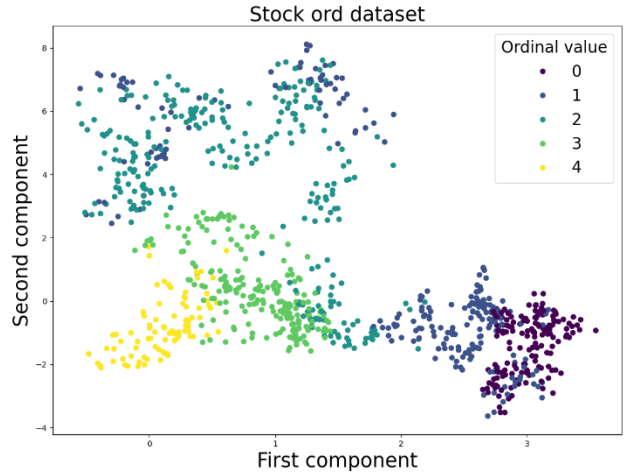


Fig. 7.14 Stock ord dataset visualization derived from the proposed method

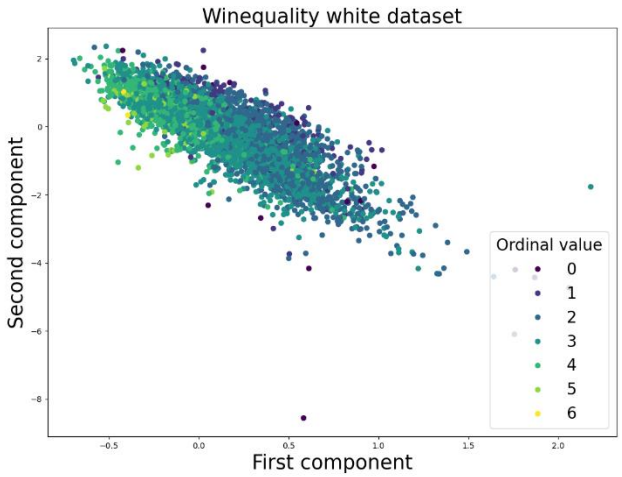


Fig. 7.15 Winequality white dataset visualization derived from the proposed method

7.2 Neural translation and automated recognition of ICD10 medical entities from natural language

7.2.1 Pre-processing

7.2.1.1 Text standardization

Minimal standardization was applied to the text data. The 6 lines present on the death certificates were concatenated with a ", " separator, and the two following steps were applied as the only text cleaning treatments:

- All letters were put to lower case,
- All space based separator were collapsed to " ".

7.2.1.2 Tokenization *for rare words*

In order to reduce the problem's dimensionality and handle rare words in the dataset, Byte pair encoding[87], the standard methodology used in the recent machine translation academic literature, was

used. Its implementation used for the experiments reported in this article can be found in the official Tensorflow Transformer repository[88].

The algorithm was applied on the entire training dataset and the derived tokenization were of cardinal 500 and 2033 for the ICD10 and French corpora, respectively. In standard machine translation problems, it is usually standard procedure to tokenize both corpora with one given tokenization, in order for instance to make use of potential similar prefixes and suffixes in both languages. However, since the ICD-10 and French vocabulary are so different (the ICD-10 classification not being a natural language), the authors decided to use a distinct tokenization for each. Not that the two derived concatenation could be concatenated to get a similar result.

7.2.2 Model definition

The model itself follows the traditional transformer architecture[89]. The model's official Tensorflow implementation was used for the experiments[88]. However, the traditional Transformer model doesn't allow for the treatment of additional conditional variables. In order to include the latter in the model, a similar methodology than that followed in [90] was chosen:

- Each element in both the ICD10 target sequence and the French input sentences (so either an ICD-10 code or a French word) are tokenized following the tokenizations obtained using byte pair encoding. The derived tokens are then fed to a linear embedding (of dimensionality "hidden_size", a model hyperparameter) similar to word2vec, whose parameters are randomly initialized and learned during training. A distinct embedding is created for the ICD-10 tokens and the French tokens
- Each conditional variable are also fed to a linear embedding of same dimensionality that the ICD-10 and French tokens) that are learnt by the model during training. One distinct

embedding is learnt per categorical variable (leading to 4 linear additional embeddings, for the gender, year, age, and origin of certificate variables)

- The mean vector of these linear projections is added in an element-wise fashion to the French embedded token sequence.
- The transformer model is used as defined in its original article on the resulting embedded sequence

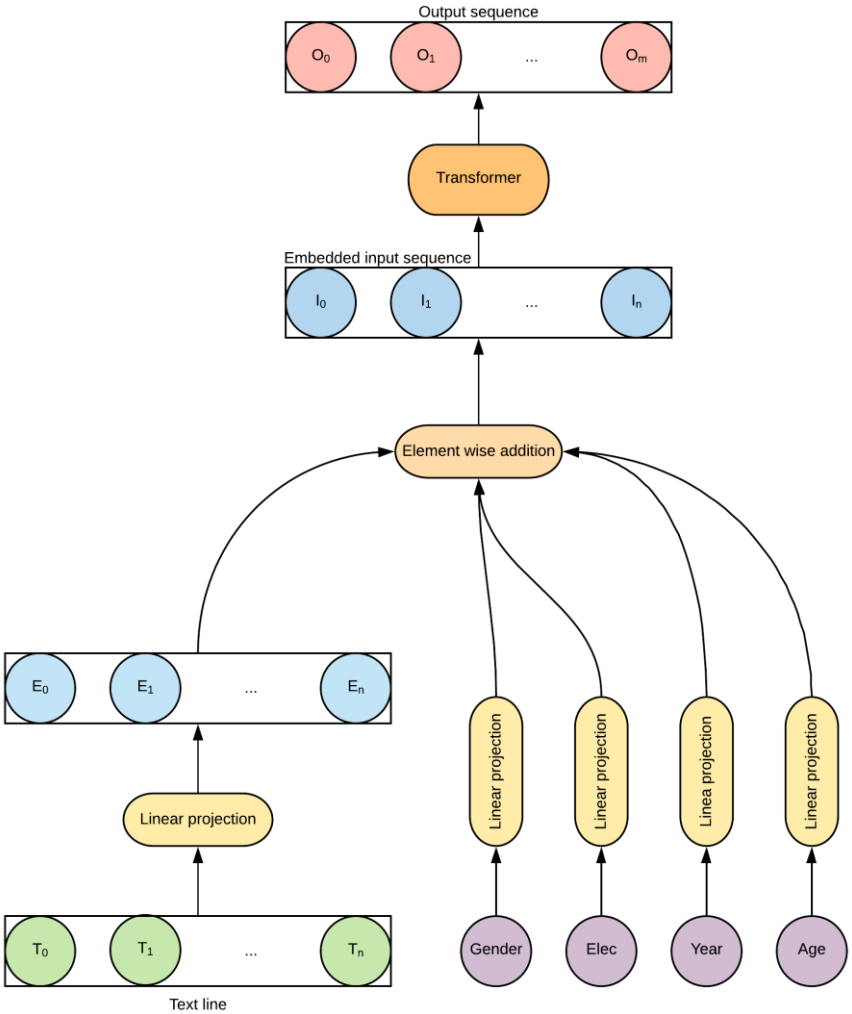


Fig. 7.16 Transformer adaptation for the handling of conditional variables

Several approaches are available in order to incorporate the categorical variable into the Transformer architecture. The authors chose this one for several reasons:

- Element-wise addition of categorical variables into the embedded input sequence has already been performed in the past (both in independent work or by work produced by the authors[90,91])
- The idea of element-wise addition of categorical variables to the embedded input sequence is already implicitly present in the Transformer architecture, in its positional encoding. Indeed, as the model doesn't use any convolutional or recurrent neural architecture, the position of each word in the sentence needs to be explicitely expressed in the input values. To do so, the authors decide to consider the position of each word in the sentence as an auxiliary categorical variable, to embed it into a "hidden_size" vector, and to add it in an element-wise fashion to each of the sentence token. They investigate whether a learnable embedding or one pre-defined is better, and end up choosing the predefined one for simplicity (this being possible because position of a token in a sentence is something reasonably easy to model using only prior knowledge). The authors followed the exact same methodology to incorporate the auxiliary conditional variables in the model, but opted for a learnable embedding, as no clear prior knowledge can be used to manually define the impact these variables have on the decision process leading to the identification of ICD-10 entities.

7.2.3 Hyperparameter search

Hyperparameter tuning was done with a random search guided by validation set's F-measure results.

The following variable were randomly sampled from the further specified probability distributions:

- Model's hidden size: sampled from a uniform random distribution between 256 and 512
- Batch size: For computational reasons, the batch size was defined as (100 * 512 / hidden_size)

- Learning rate: Uniformly sampled from discrete values 1. or 2. (note that this value doesn't constitute the actual learning rate, which is modified by the function "get_learning_rate"
- Layer_postprocess_dropout: sampled from a uniform random distribution between 0 and 0.2
- Attention_dropout: sampled from a uniform random distribution between 0 and 0.2
- Relu_dropout: sampled from a uniform random distribution between 0 and 0.2

All other parameters were fixed as recommended by the BASE_MULTI_GPU settings provided in the tensorflow transformer official implementation.

40 models were trained with different hyperparameters sampled from these distributions, the best set of hyperparameter was then used to train a new set of model for ensembling.

7.2.4 Ensembling method

Due to computational reasons, the traditional method for ensembling neural translation model (logits averaging during the beam search process) could not be used. The following alternative was used instead:

- Get the prediction from each model
- Compute F-measurements between all prediction candidates
- Select the prediction that shows highest F-measurements with other candidates on average

The ensemble of models was selected by a greedy search on all the models trained for the experiment (40 models trained during the hyperparameters search and additional models trained with the best hyperparameter set) guided by the F-measurement on the validation set

The derived score was taken as the prediction scores' average.

7.2.5 Final ensemble hyperparameters

The final ensemble found by greedy exploration consisted of 7 different models, 5 of which were trained with the best set of hyperparameters revealed by the random hyperparameter searchs. The three distinct sets of hyperparameters can be found in table 7.1, and the individual performances of each model can be found in table 7.2

Hyperparameter	Set 1 (best set)	Set 2	Set 3
Batch size	172	152	164
Hidden size	296	336	312
Learning rate	2.	2.	2.
Layer postprocess dropout	.073	.12	.005
Attention dropout	.105	.030	.017
Relu dropout	.173	.030	.20

Table 7.1 Sets of hyperparameters for the different models used in the final ensemble

Model	Precision	Recall	F-measure
Model 1 (hyperparameter set 2)	94.6	93.3	93.9
Model 2 (hyperparameter set 3)	94.5	93.1	93.8
Model 3 (hyperparameter set 1)	94.4	93.4	93.9
Model 4 (hyperparameter set 1)	94.6	93.4	94.0
Model 5 (hyperparameter set 1)	94.6	93.4	94.0
Model 6 (hyperparameter set 1)	94.6	93.5	94.0
Model 7 (hyperparameter set 1)	94.5	93.4	94.0

Overall ensemble	94.9	93.7	94.3

Table 7.2 Performance metrics on the test set for each individual model from the ensemble (on both paper and electronic certificates). The ensemble value reported in the main article is also reported to allow comparison

7.2.6 Error examples

Text	hta, insuffisance cardiaque, anévrisme aorte !, !, asystolie !
Predicted ICD10	I10 I509 I714 I500
Database ICD10	I10 I509 I714 H570 I500 R068
Expert ICD10	I10 I509 I714 I500

Table 7.3 Example of "missing data" type error. The database shows two additional codes that are not present in the text according to the medical expert. These codes are probably associated with the "!" present in the text, and were derived from a human coder reading the handwritten death certificate.

Text	acfa, hta, connu vertige, retrouvé terre bas escalier
Predicted ICD10	I48 I10 R42 R98
Database ICD10	I48 I10 R42 W10
Expert ICD10	I48 I10 R42 W10

Table 7.4 Example of contextual error. The proposed approach converts "retrouvé terre bas escalier" (which roughly translates to "found at the bottom of the stairs") to R98 "unattended death". Both human coders are able to deduce that the subject probably fell down the stairs and use the ICD10 code W10 "Fall on and from steps"

Text	cardiopathie ischemique avec triple pontage, anevrisme aortique, cancer de la vessie, hta, arret cardio - respiratoire
Predicted ICD10	I259 Z951 I719 C679 I10 R092
Database ICD10	I259 I251 I719 C679 I10 R092
Expert ICD10	I259 I251 I719 C679 I10 R092

Table 7.5 Example of error caused by a coding rule. I251 and Z951 are both suitable for "triple pontage" (Coronary artery bypass surgery). However, the M4 mortality coding rule (Special instructions on surgery and other medical procedures) dictates the code choice

7.3 A deep artificial neural network based model for underlying cause of death prediction from death certificates

7.3.1 Model architecture

The model architecture is mostly inspired from the Inception v2 network [92], with the following modifications:

- The Inception v2 network's first stage has been replaced with two temporal blocks [51] with successive dilation rates of 1 and 2
- Drop-out [93] , layer normalization [94] and residual connections [95] were applied to each block as can be seen on Figures 7.17, 7.18 and 7.19
- The Inception maximum pooling operation was limited to the grid's width dimension as can be seen on Figure 7.20
- The final softmax operation is tied to the linear embedding as described in [49]

The model's full structure is described in Table 7.6 and Figures 7.17 through 7.20

Type	Layer size	Dilation rate, stride or remarks	Input size
Linear embedding	512	_	$6 \times 20 \times 7404$
Temporal block [51]	512	1	$6 \times 20 \times 512$
Temporal block	512	2	$6 \times 20 \times 512$
3 x Inception block 1 [92]	512	As in Figure 7.17	$6 \times 20 \times 512$
Inception pooling [92]	1024	As in Figure 7.20	$6 \times 20 \times 512$
5 x Inception block 2 [92]	1024	As in Figure 7.18	$6 \times 9 \times 1024$
Inception pooling	1536	As in Figure 7.20	$6 \times 9 \times 1024$

2 x Inception block 3 [92]	1536	As in Figure 7.19	$6 \times 4 \times 1536$
Full maximum pooling	1536	6×4	$6 \times 4 \times 1536$
Linear layer	512	_	$1 \times 1 \times 1536$
Tied linear embedding [49]	7404	Transpose of the first linear embedding matrix	$1 \times 1 \times 512$

Table 7.6 Model architecture and corresponding hyperparameters

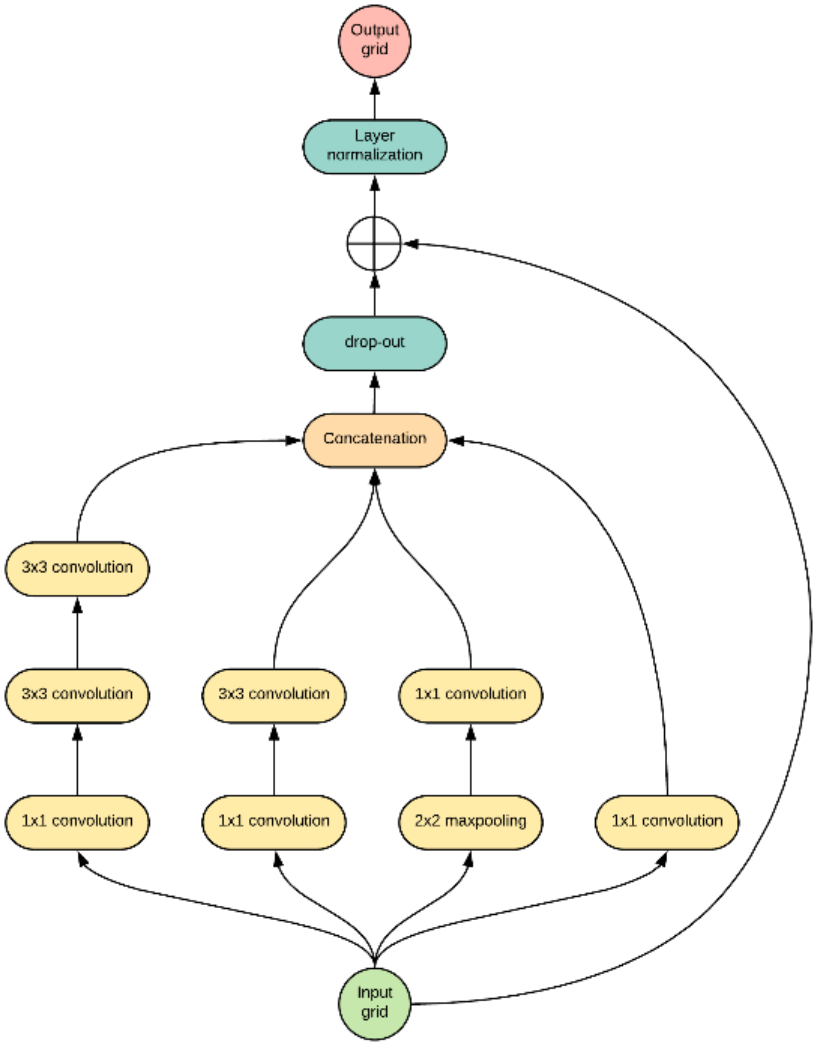


Fig. 7.17 Inception block 1 as described in [92] with additional drop-out, layer normalization and residual connection mechanisms

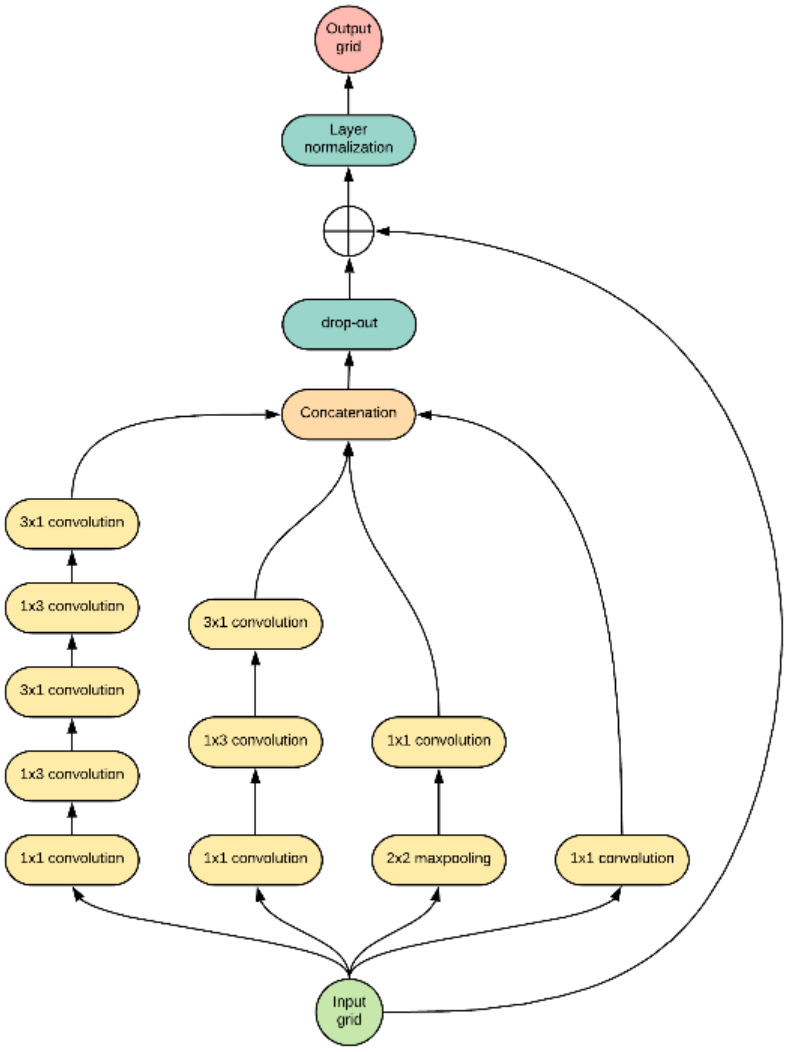


Fig. 7.18 Inception block 2 as described in [92] with additional drop-out, layer normalization and residual connection mechanisms

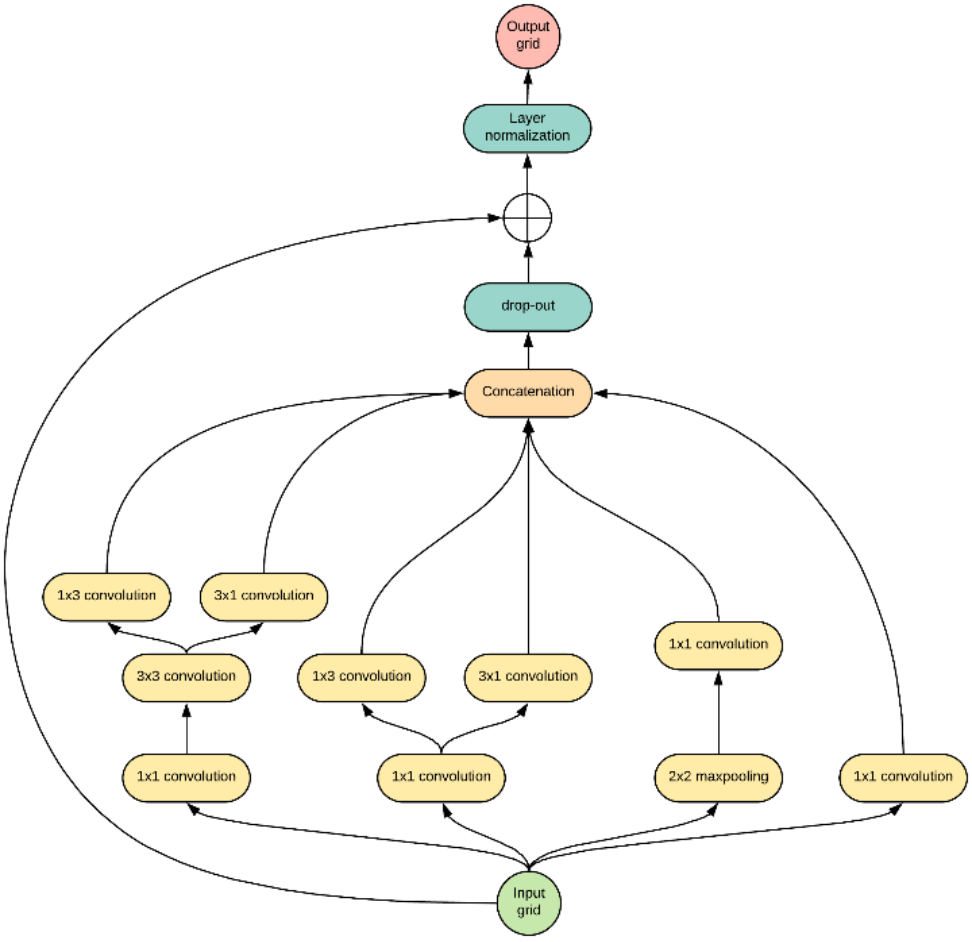


Fig. 7.19 Inception block 3 as described in [92] with additional drop-out, layer normalization and residual connection mechanisms

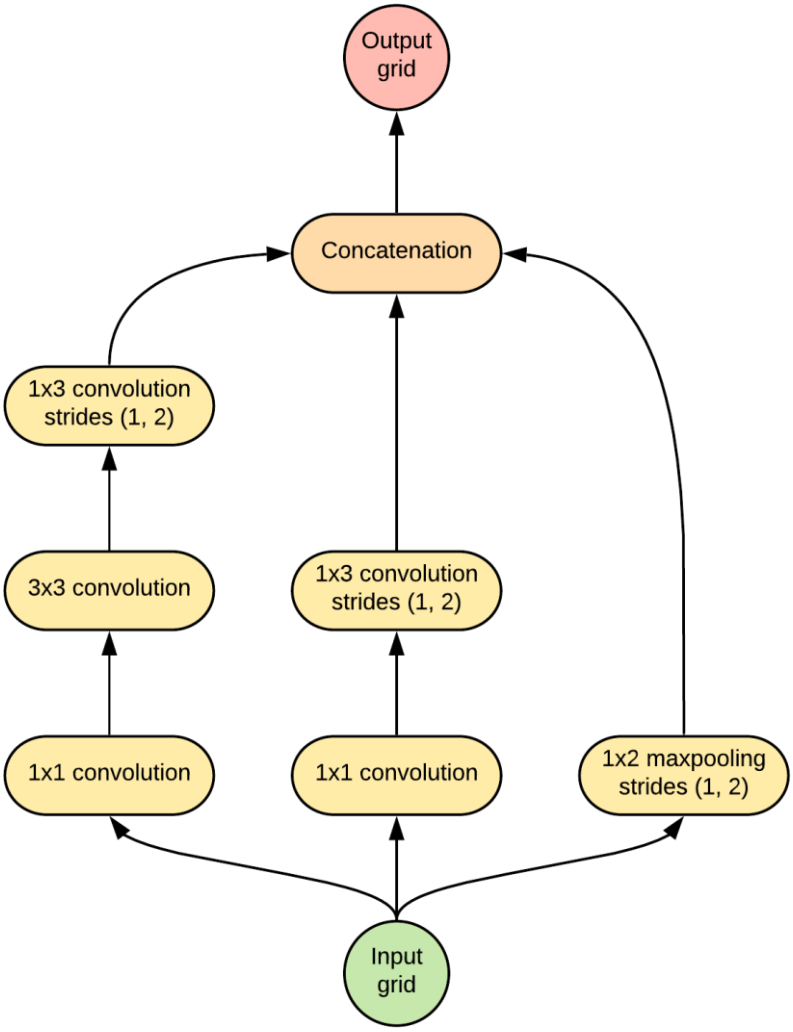


Fig. 7.20 Inception pooling as described in [92]. All grid-reducing operations are limited to the width dimension

7.3.2 Training methodology

The model was implemented with Tensorflow, a python-based distributed machine learning framework, on two NVidia RTX 2070 GPUs simultaneously using a mirrored distribution strategy. Training was performed using a variant of stochastic gradient descent, the Adam optimization algorithm.

The descent's step size (also called learning rate in the machine learning academic literature) was updated in real time during training according a rule defined in [89], that can be seen in figure 7.21 and is defined according to the formula:

$$Step\_size(t) = \alpha \cdot \min(t^{-0.5}, t \cdot warmup\_steps^{-1.5}) \qquad \forall t \in \mathbb{R}^{+}$$

With:

- $\alpha \in \mathbb{R}$ a constant considered as a model hyperparameter and defining the learning rate's overall amplitude
- $warmup\_steps \in \mathbb{N}$ another hyperparameter defining the learning rate's linear warmup phase length

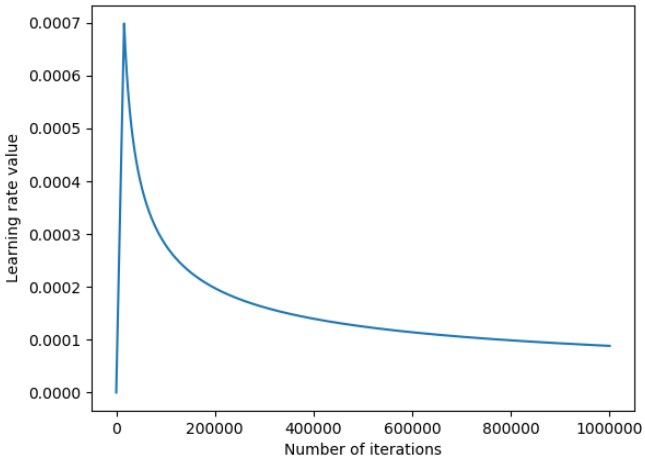


Fig. 7.21 learning rate evolution with gradient descent iterations. The learning rate follows a first linear increase warmup phase followed by an inverse root square decay

In order to limit gradient explosion phenomena typically encountered in deep neural network, the optimization was in addition controlled using gradient clipping. Essentially, the norm of all gradients computed during the descent were normalized to be of global norm equal or less than 0.1.

In addition, label smoothing was applied to the cross entropy loss to further regularize the model.

The final hyper parameters were chosen from a random search selection process with the following values:

- Batch size: 250
- Drop-out selection rate: 0.1 for all layers
- Label smoothing parameter: 0.1
- Initial learning rate constant: $\frac{2}{\sqrt{512}} \approx 0.088$
- Learning rate warmup steps: 16000
- Trainable variable initialization: Uniform variance scaling initializing

7.3.3 Example of mispredicted certificates

Line	Content		
1	I501 Left ventricular failure		
2	J690 Acute respiratory failure		
6	I509 Heart failure, unspecified		
Source	Medical expert	Prediction	Dataset
UCD	I501	J690	I501

Table 7.7 Example of death certificate where the prediction differs from the underlying cause of death presented in the dataset. The medical expert agreed with the code present in the dataset, and commented that J690 is not a valid underlying cause of death code

Line	Content
1	R688 Other specified general symptoms and signs
2	N19 Unspecified kidney failure E148 Unspecified diabetes mellitus with unspecified complications
3	I499 Cardiac arrhythmia, unspecified I519 Heart disease, unspecified I501 Left ventricular failure E46 Unspecific protein-energy malnutrition
4	Z896 Acquired absence of leg above knee
5	I702 Atherosclerosis of arteries of extremities
6	Z740 Need for assistance due to reduced mobility I694 Sequelae of stroke, not specified as haemorrhage or infarction

Source	Medical expert	Prediction	Dataset
UCD	I501	I702	I501

Table 7.8 Example of death certificate where the prediction differs from the underlying cause of death presented in the dataset. The medical expert agreed with the code present in the dataset, and commented that this certificate is subject to "linked causes" a set of casuistic exceptions

Line	Content
1	Q300 Choanal atresia
2	Q878 Other congenital malformation syndromes, not elsewhere classified
3	Q213 Tetralogy of Fallot Q165 Congenital malformation of inner ear I678 Other specified cerebrovascular diseases
6	P013 Fetus and newborn affected by polyhydramnios

Source	Medical expert	Prediction	Dataset
UCD	Q897 Multiple congenital malformations, not elsewhere classified	Q878	Q300

Table 7.9 Example of death certificate where the prediction differs from the underlying cause of death presented in the dataset. The medical expert disagreed with both values, and commented that this certificate constitutes a "rare case requiring the medical referent's expertise"

Line	Content
1	I509 Heart failure, unspecified
2	I259 Chronic ischaemic heart disease, unspecified
3	E109 Type 1 Diabetes mellitus without complications

Source	Medical expert	Prediction	Dataset
UCD	E108 Type 1 diabetes mellitus with unspecified complications	E106 Type 1 diabetes mellitus with other specified complications	E109

Table 7.10 Example of death certificate where the prediction differs from the underlying cause of death presented in the dataset. The medical expert disagreed with

both values, and commented that, when coding diabetes related certificates, the underlying cause of death's fourth character is often subject to interpretation

7.4 On the comparability of international cause of death statistics: A deep artificial neural network based study

7.4.1 Tables of predictive power variation at the Eurostat level

Category	France	US	Italy	England	Prevalence (%)
1.1 Tuberculosis	0.951	0.946	0.951	0.951	0.131
1.2 AIDS (HIV-disease)	0.957	0.974	0.966	0.944	0.136
1.3 Viral hepatitis	0.938	0.907	0.907	0.902	0.132
1.4 Other infectious and parasitic diseases	0.954	0.945	0.945	0.938	1.562
2.1.1 Malignant neoplasm of lip, oral cavity, pharynx	0.983	0.944	0.948	0.955	0.792
2.1.2 Malignant neoplasm of oesophagus	0.989	0.970	0.969	0.974	0.713
2.1.3 Malignant neoplasm of stomach	0.996	0.974	0.978	0.983	0.911
2.1.4 Malignant neoplasm of colon, rectum and anus	0.991	0.967	0.970	0.974	3.042
2.1.5 Malignant neoplasm of liver and intrahepatic bile ducts	0.982	0.974	0.975	0.975	1.388
2.1.6 Malignant neoplasm of pancreas	0.996	0.980	0.983	0.988	1.682
2.1.7 Malignant neoplasm of larynx	0.981	0.934	0.939	0.947	0.242
2.1.8 Malignant neoplasm of trachea, bronchus, lung	0.991	0.982	0.981	0.986	5.211
2.1.9 Malignant melanoma of skin	0.998	0.994	0.994	0.994	0.282
2.1.10 Malignant neoplasm of breast	0.994	0.968	0.970	0.974	2.135
2.1.11 Malignant neoplasm of cervix uteri	0.988	0.980	0.976	0.976	0.148
2.1.12 Malignant neoplasm of other and unspecified parts of uterus	0.991	0.965	0.973	0.975	0.454
2.1.13 Malignant neoplasm of ovary	0.997	0.981	0.982	0.991	0.644
2.1.14 Malignant neoplasm of prostate	0.990	0.954	0.957	0.963	1.706
2.1.15 Malignant neoplasm of kidney	0.991	0.955	0.956	0.963	0.649
2.1.16 Malignant neoplasm of bladder	0.990	0.950	0.954	0.958	0.903
2.1.17 Malignant neoplasm of brain and central nervous system	0.990	0.979	0.980	0.978	0.622
2.1.18 Malignant neoplasm of thyroid	1.000	0.958	0.967	0.967	0.071
2.1.19 Hodgkin disease and lymphomas	0.986	0.954	0.955	0.951	0.868
2.1.20 Leukaemia	0.988	0.968	0.966	0.969	1.095
2.1.21 Other malignant neoplasm of lymphoid and haematopoietic tissue	0.985	0.972	0.965	0.968	0.527
2.1.22 Other malignant neoplasms	0.982	0.981	0.979	0.979	4.031
2.2 Non-malignant neoplasms (benign and uncertain)	0.973	0.965	0.970	0.970	1.216
3. Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism	0.885	0.905	0.897	0.894	0.404
4.1 Diabetes mellitus	0.975	0.969	0.958	0.961	2.106
4.2 Other endocrine, nutritional and metabolic diseases	0.955	0.915	0.916	0.939	1.569
5.1 Dementia	0.986	0.904	0.947	0.888	2.375
5.2 Alcohol abuse (including alcoholic psychosis)	0.969	0.949	0.952	0.957	0.566
5.3 Drug dependence, toxicomania	0.923	0.923	0.923	0.923	0.031
5.4 Other mental and behavioural disorders	0.936	0.896	0.921	0.906	0.546
6.1 Parkinson's disease	0.988	0.961	0.976	0.972	0.872

Category	France	US	Italy	England	Prevalence (%)
6.2 Alzheimer's disease	0.992	0.961	0.976	0.960	2.936
6.3 Other diseases of the nervous system and the sense organs	0.963	0.944	0.945	0.946	1.721
7.1.1 Acute myocardial infarction	0.992	0.986	0.987	0.990	3.594
7.1.2 Other ischaemic heart diseases	0.970	0.959	0.962	0.964	3.475
7.2 Other heart diseases	0.984	0.973	0.974	0.978	9.179
7.3 Cerebrovascular diseases	0.986	0.981	0.980	0.980	6.267
7.4 Other diseases of the circulatory system	0.972	0.940	0.936	0.962	4.897
8.1 Influenza	0.984	0.967	0.973	0.962	0.108
8.2 Pneumonia	0.959	0.922	0.896	0.922	2.024
8.3.1 Asthma	0.976	0.970	0.970	0.979	0.195
8.3.2 Other chronic lower respiratory diseases	0.980	0.963	0.965	0.972	1.507
8.4 Other diseases of the respiratory system	0.967	0.964	0.967	0.959	2.534
9.1 Ulcer of stomach, duodenum and jejunum	0.949	0.925	0.929	0.933	0.149
9.2 Cirrhosis, fibrosis and chronic hepatitis	0.979	0.973	0.975	0.973	1.355
9.3 Other diseases of the digestive system	0.970	0.963	0.963	0.966	2.809
10. Diseases of the skin and subcutaneous tissue	0.947	0.906	0.904	0.941	0.299
11.1 Rheumatoid arthritis and osteoarthrosis	0.962	0.923	0.923	0.907	0.107
11.2 Other diseases of the musculoskeletal system/connective tissue	0.941	0.924	0.911	0.922	0.576
12.1 Diseases of kidney and ureter	0.968	0.956	0.954	0.953	1.235
12.2 Other diseases of the genitourinary system	0.962	0.935	0.935	0.950	0.353
13. Complications of pregnancy, childbirth and puerperium	0.765	0.882	0.647	0.765	0.010
14. Certain conditions originating in the perinatal period	0.942	0.923	0.927	0.931	0.282
15. Congenital malformations and chromosomal abnormalities	0.929	0.885	0.885	0.898	0.282
16.1 Sudden infant death syndrome	0.940	0.929	0.940	0.929	0.049
16.2 Unknown and unspecified causes	0.998	0.964	0.997	0.976	2.823
16.3 Other symptoms, signs, ill-defined causes	0.997	0.994	0.989	0.948	4.370
17.1.1 Transport accidents	0.991	0.990	0.051	0.982	0.938
17.1.2 Accidental falls	0.973	0.960	0.355	0.951	1.097
17.1.3 Drowning and accidental submersion	0.966	0.959	0.810	0.959	0.174
17.1.4 Accidental poisoning	0.934	0.915	0.629	0.911	0.278
17.1.5 Other accidents	0.974	0.902	0.268	0.948	2.339
17.2 Suicide and intentional self-harm	0.992	0.991	0.805	0.990	1.859
17.3 Homicide, assault	0.912	0.900	0.394	0.887	0.094
17.4 Events of undetermined intent	0.901	0.895	0.361	0.890	0.112
17.5 Other external causes of injury and poisoning	0.820	0.448	0.155	0.478	0.213
Overall accuracy	0.980	0.961	0.927	0.964	nan

Fig. 7.22 Agreement at the Eurostat scale, per Eurostat chapter, for French certificates

Category	France	US	Italy	England	Prevalence (%)
1.1 Tuberculosis	0.952	0.935	0.952	0.968	0.044
1.2 AIDS (HIV-disease)	0.975	0.975	0.975	0.975	0.029
1.3 Viral hepatitis	0.971	0.943	0.943	0.943	0.025
1.4 Other infectious and parasitic diseases	0.877	0.941	0.931	0.952	0.769
2.1.1 Malignant neoplasm of lip, oral cavity, pharynx	1.000	0.971	0.981	1.000	0.526
2.1.2 Malignant neoplasm of oesophagus	0.999	0.996	0.997	1.000	1.727
2.1.3 Malignant neoplasm of stomach	0.999	0.995	0.995	1.000	1.041
2.1.4 Malignant neoplasm of colon, rectum and anus	0.998	0.995	0.995	0.998	3.639
2.1.5 Malignant neoplasm of liver and intrahepatic bile ducts	0.997	0.996	0.995	0.998	0.911
2.1.6 Malignant neoplasm of pancreas	1.000	0.996	0.997	1.000	1.926
2.1.7 Malignant neoplasm of larynx	1.000	0.995	0.995	1.000	0.156
2.1.8 Malignant neoplasm of trachea, bronchus, lung	0.998	0.997	0.997	0.999	7.493
2.1.9 Malignant melanoma of skin	0.993	0.989	0.990	0.996	0.518
2.1.10 Malignant neoplasm of breast	0.998	0.996	0.996	0.998	2.783
2.1.11 Malignant neoplasm of cervix uteri	0.993	0.993	0.993	0.996	0.197
2.1.12 Malignant neoplasm of other and unspecified parts of uterus	0.997	0.989	0.990	0.998	0.447
2.1.13 Malignant neoplasm of ovary	0.999	0.994	0.994	0.999	0.999
2.1.14 Malignant neoplasm of prostate	0.995	0.987	0.988	0.997	2.538
2.1.15 Malignant neoplasm of kidney	0.997	0.995	0.995	0.997	0.842
2.1.16 Malignant neoplasm of bladder	0.998	0.985	0.989	0.999	1.131
2.1.17 Malignant neoplasm of brain and central nervous system	0.999	0.998	0.998	0.999	0.885
2.1.18 Malignant neoplasm of thyroid	0.992	0.985	0.985	0.992	0.093
2.1.19 Hodgkin disease and lymphomas	0.996	0.987	0.994	0.993	0.964
2.1.20 Leukaemia	0.994	0.990	0.987	0.996	0.974
2.1.21 Other malignant neoplasm of lymphoid and haematopoietic tissue	0.996	0.991	0.991	0.996	0.653
2.1.22 Other malignant neoplasms	0.965	0.994	0.994	0.997	3.602
2.2 Non-malignant neoplasms (benign and uncertain)	0.972	0.974	0.976	0.986	0.656
3. Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism	0.860	0.892	0.871	0.957	0.199
4.1 Diabetes mellitus	0.958	0.961	0.931	0.987	1.155
4.2 Other endocrine, nutritional and metabolic diseases	0.958	0.939	0.946	0.967	0.306
5.1 Dementia	0.987	0.978	0.979	0.993	7.568

Category	France	US	Italy	England	Prevalence (%)
5.2 Alcohol abuse (including alcoholic psychosis)	0.909	0.727	0.773	0.909	0.016
5.3 Drug dependence, toxicomania	1.000	1.000	1.000	1.000	0.001
5.4 Other mental and behavioural disorders	0.955	0.932	0.943	0.955	0.063
6.1 Parkinson's disease	0.993	0.984	0.987	0.996	1.080
6.2 Alzheimer's disease	0.994	0.987	0.989	0.999	2.242
6.3 Other diseases of the nervous system and the sense organs	0.982	0.970	0.966	0.990	1.477
7.1.1 Acute myocardial infarction	0.991	0.989	0.991	0.995	4.217
7.1.2 Other ischaemic heart diseases	0.979	0.978	0.980	0.991	4.779
7.2 Other heart diseases	0.967	0.968	0.970	0.986	3.806
7.3 Cerebrovascular diseases	0.980	0.986	0.984	0.996	9.349
7.4 Other diseases of the circulatory system	0.967	0.926	0.931	0.985	2.583
8.1 Influenza	0.986	0.986	0.986	0.986	0.049
8.2 Pneumonia	0.966	0.973	0.961	0.992	6.115
8.3.1 Asthma	0.969	0.963	0.966	0.988	0.230
8.3.2 Other chronic lower respiratory diseases	0.986	0.981	0.983	0.996	5.816
8.4 Other diseases of the respiratory system	0.870	0.959	0.971	0.985	3.376
9.1 Ulcer of stomach, duodenum and jejunum	0.967	0.967	0.970	0.982	0.285
9.2 Cirrhosis, fibrosis and chronic hepatitis	0.986	0.980	0.978	0.988	0.619
9.3 Other diseases of the digestive system	0.974	0.974	0.976	0.987	2.649
10. Diseases of the skin and subcutaneous tissue	0.640	0.926	0.899	0.979	0.369
11.1 Rheumatoid arthritis and osteoarthrosis	0.967	0.933	0.940	0.960	0.214
11.2 Other diseases of the musculoskeletal system/connective tissue	0.913	0.948	0.935	0.982	0.436
12.1 Diseases of kidney and ureter	0.912	0.933	0.952	0.974	0.891
12.2 Other diseases of the genitourinary system	0.973	0.898	0.898	0.984	1.430
14. Certain conditions originating in the perinatal period	1.000	1.000	1.000	1.000	0.008
15. Congenital malformations and chromosomal abnormalities	0.919	0.891	0.875	0.931	0.177
16.3 Other symptoms, signs, ill-defined causes	1.000	1.000	1.000	0.999	2.592
17.1.2 Accidental falls	0.909	0.939	0.485	0.970	0.024
17.1.4 Accidental poisoning	0.000	0.000	0.000	0.000	0.001
17.1.5 Other accidents	0.684	0.868	0.025	0.992	0.282
17.5 Other external causes of injury and poisoning	0.333	0.410	0.154	0.744	0.028
Overall accuracy	0.977	0.979	0.977	0.993	nan

Fig. 7.23 Agreement at the Eurostat scale, per Eurostat chapter, for English certificates

Category	France	US	Italy	England	Prevalence (%)
1.1 Tuberculosis	0.963	0.963	1.000	1.000	0.045
1.2 AIDS (HIV-disease)	0.695	0.966	0.932	0.644	0.098
1.3 Viral hepatitis	0.977	0.946	0.973	0.965	0.428
1.4 Other infectious and parasitic diseases	0.883	0.978	0.978	0.944	1.578
2.1.1 Malignant neoplasm of lip, oral cavity, pharynx	0.993	0.951	0.985	0.989	0.447
2.1.2 Malignant neoplasm of oesophagus	0.994	0.981	0.994	0.994	0.263
2.1.3 Malignant neoplasm of stomach	0.998	0.980	0.995	0.992	1.537
2.1.4 Malignant neoplasm of colon, rectum and anus	0.996	0.981	0.996	0.994	3.118
2.1.5 Malignant neoplasm of liver and intrahepatic bile ducts	0.985	0.988	0.996	0.991	1.580
2.1.6 Malignant neoplasm of pancreas	0.998	0.994	0.998	0.998	1.805
2.1.7 Malignant neoplasm of larynx	0.967	0.948	0.987	0.967	0.255
2.1.8 Malignant neoplasm of trachea, bronchus, lung	0.998	0.996	0.998	0.995	5.580
2.1.9 Malignant melanoma of skin	0.995	1.000	1.000	1.000	0.327
2.1.10 Malignant neoplasm of breast	0.998	0.988	0.995	0.989	2.008
2.1.11 Malignant neoplasm of cervix uteri	1.000	1.000	1.000	1.000	0.065
2.1.12 Malignant neoplasm of other and unspecified parts of uterus	0.996	0.974	1.000	1.000	0.392
2.1.13 Malignant neoplasm of ovary	0.997	0.986	0.993	0.997	0.488
2.1.14 Malignant neoplasm of prostate	0.996	0.973	0.993	0.988	1.228
2.1.15 Malignant neoplasm of kidney	0.988	0.980	0.994	0.988	0.573
2.1.16 Malignant neoplasm of bladder	0.995	0.969	0.995	0.995	0.912
2.1.17 Malignant neoplasm of brain and central nervous system	0.995	0.992	0.995	0.992	0.647
2.1.18 Malignant neoplasm of thyroid	1.000	0.962	0.981	0.981	0.087
2.1.19 Hodgkin disease and lymphomas	0.996	0.978	0.996	0.988	0.840
2.1.20 Leukaemia	0.994	0.977	0.995	0.995	1.037
2.1.21 Other malignant neoplasm of lymphoid and haematopoietic tissue	0.992	0.989	0.997	0.989	0.595
2.1.22 Other malignant neoplasms	0.881	0.986	0.986	0.881	3.658
2.2 Non-malignant neoplasms (benign and uncertain)	0.959	0.947	0.969	0.955	1.397
3. Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism	0.824	0.912	0.912	0.915	0.492
4.1 Diabetes mellitus	0.968	0.979	0.989	0.976	3.433
4.2 Other endocrine, nutritional and metabolic diseases	0.884	0.924	0.965	0.900	1.052
5.1 Dementia	0.980	0.972	0.989	0.985	2.940

Category	France	US	Italy	England	Prevalence (%)
5.2 Alcohol abuse (including alcoholic psychosis)	0.913	0.826	0.913	0.913	0.038
5.3 Drug dependence, toxicomania	0.900	0.900	0.900	0.900	0.017
5.4 Other mental and behavioural disorders	0.887	0.930	0.948	0.913	0.192
6.1 Parkinson's disease	0.957	0.924	0.983	0.932	1.008
6.2 Alzheimer's disease	0.983	0.973	0.994	0.979	1.828
6.3 Other diseases of the nervous system and the sense organs	0.915	0.925	0.942	0.906	1.560
7.1.1 Acute myocardial infarction	0.973	0.987	0.993	0.977	3.917
7.1.2 Other ischaemic heart diseases	0.964	0.984	0.989	0.972	7.308
7.2 Other heart diseases	0.928	0.973	0.984	0.936	8.463
7.3 Cerebrovascular diseases	0.975	0.975	0.992	0.970	9.638
7.4 Other diseases of the circulatory system	0.947	0.953	0.981	0.955	7.300
8.1 Influenza	0.977	0.977	0.977	0.977	0.072
8.2 Pneumonia	0.968	0.974	0.987	0.984	1.733
8.3.1 Asthma	0.957	0.957	1.000	0.978	0.077
8.3.2 Other chronic lower respiratory diseases	0.971	0.931	0.988	0.966	3.543
8.4 Other diseases of the respiratory system	0.868	0.958	0.975	0.902	2.005
9.1 Ulcer of stomach, duodenum and jejunum	0.983	0.966	0.983	0.983	0.097
9.2 Cirrhosis, fibrosis and chronic hepatitis	0.949	0.980	0.989	0.956	0.918
9.3 Other diseases of the digestive system	0.962	0.966	0.970	0.961	2.577
10. Diseases of the skin and subcutaneous tissue	0.873	0.951	0.971	0.961	0.170
11.1 Rheumatoid arthritis and osteoarthrosis	0.929	0.929	0.964	0.929	0.187
11.2 Other diseases of the musculoskeletal system/connective tissue	0.836	0.882	0.916	0.887	0.397
12.1 Diseases of kidney and ureter	0.811	0.864	0.972	0.838	1.617
12.2 Other diseases of the genitourinary system	0.934	0.955	0.960	0.955	0.330
13. Complications of pregnancy, childbirth and puerperium	0.500	0.500	0.500	0.500	0.003
14. Certain conditions originating in the perinatal period	0.667	1.000	1.000	1.000	0.005
15. Congenital malformations and chromosomal abnormalities	0.943	0.920	0.943	0.943	0.147
16.2 Unknown and unspecified causes	0.984	0.953	0.990	0.948	0.320
16.3 Other symptoms, signs, ill-defined causes	0.989	0.995	0.999	0.742	1.703
NA	0.014	0.025	0.929	0.026	3.925
Overall accuracy	0.920	0.933	0.984	0.921	nan

Fig. 7.24 Agreement at the Eurostat scale, per Eurostat chapter, for Italian certificates

Category	France	US	Italy	England	Prevalence (%)
1.1 Tuberculosis	0.977	0.985	0.985	0.977	0.029
1.2 AIDS (HIV-disease)	0.951	0.979	0.962	0.955	0.424
1.3 Viral hepatitis	0.990	0.989	0.981	0.983	0.281
1.4 Other infectious and parasitic diseases	0.915	0.972	0.954	0.946	1.948
2.1.1 Malignant neoplasm of lip, oral cavity, pharynx	0.997	0.996	0.995	0.996	0.336
2.1.2 Malignant neoplasm of oesophagus	0.995	0.998	0.997	0.995	0.573
2.1.3 Malignant neoplasm of stomach	1.000	1.000	0.999	1.000	0.469
2.1.4 Malignant neoplasm of colon, rectum and anus	0.997	0.998	0.997	0.996	2.145
2.1.5 Malignant neoplasm of liver and intrahepatic bile ducts	0.997	0.998	0.997	0.997	0.767
2.1.6 Malignant neoplasm of pancreas	0.999	0.999	0.998	0.998	1.403
2.1.7 Malignant neoplasm of larynx	0.992	0.997	0.995	0.989	0.145
2.1.8 Malignant neoplasm of trachea, bronchus, lung	0.997	0.999	0.999	0.997	6.269
2.1.9 Malignant melanoma of skin	0.996	0.994	0.995	0.996	0.322
2.1.10 Malignant neoplasm of breast	0.996	0.998	0.996	0.995	1.670
2.1.11 Malignant neoplasm of cervix uteri	1.000	0.999	0.999	0.999	0.156
2.1.12 Malignant neoplasm of other and unspecified parts of uterus	0.995	0.998	0.998	0.996	0.320
2.1.13 Malignant neoplasm of ovary	0.998	0.998	0.997	0.997	0.577
2.1.14 Malignant neoplasm of prostate	0.994	0.997	0.996	0.993	1.160
2.1.15 Malignant neoplasm of kidney	0.997	0.997	0.997	0.996	0.495
2.1.16 Malignant neoplasm of bladder	0.998	0.998	0.998	0.998	0.556
2.1.17 Malignant neoplasm of brain and central nervous system	0.999	0.998	0.999	0.998	0.562
2.1.18 Malignant neoplasm of thyroid	1.000	1.000	1.000	1.000	0.065
2.1.19 Hodgkin disease and lymphomas	0.994	0.992	0.991	0.988	0.872
2.1.20 Leukaemia	0.995	0.995	0.993	0.990	0.894
2.1.21 Other malignant neoplasm of lymphoid and haematopoietic tissue	0.994	0.998	0.995	0.991	0.483
2.1.22 Other malignant neoplasms	0.944	0.994	0.986	0.973	2.544
2.2 Non-malignant neoplasms (benign and uncertain)	0.967	0.985	0.980	0.981	0.603
3. Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism	0.881	0.941	0.913	0.906	0.400
4.1 Diabetes mellitus	0.978	0.994	0.968	0.972	2.905
4.2 Other endocrine, nutritional and metabolic diseases	0.950	0.973	0.963	0.959	1.193
5.1 Dementia	0.964	0.991	0.986	0.932	3.502
5.2 Alcohol abuse (including alcoholic psychosis)	0.921	0.953	0.937	0.942	0.291
5.3 Drug dependence, toxicomania	0.890	0.915	0.932	0.902	0.091
5.4 Other mental and behavioural disorders	0.889	0.949	0.908	0.906	0.186
6.1 Parkinson's disease	0.988	0.993	0.993	0.977	0.846
6.2 Alzheimer's disease	0.993	0.996	0.997	0.987	3.142

Category	France	US	Italy	England	Prevalence (%)
6.3 Other diseases of the nervous system and the sense organs	0.959	0.973	0.970	0.964	1.441
7.1.1 Acute myocardial infarction	0.990	0.995	0.995	0.993	5.498
7.1.2 Other ischaemic heart diseases	0.989	0.994	0.992	0.991	11.040
7.2 Other heart diseases	0.959	0.984	0.972	0.965	6.707
7.3 Cerebrovascular diseases	0.969	0.991	0.979	0.979	5.664
7.4 Other diseases of the circulatory system	0.971	0.985	0.969	0.975	4.546
8.1 Influenza	0.979	0.981	0.981	0.976	0.084
8.2 Pneumonia	0.929	0.979	0.945	0.975	2.210
8.3.1 Asthma	0.957	0.972	0.958	0.969	0.150
8.3.2 Other chronic lower respiratory diseases	0.971	0.990	0.985	0.975	5.313
8.4 Other diseases of the respiratory system	0.889	0.979	0.961	0.937	1.990
9.1 Ulcer of stomach, duodenum and jejunum	0.974	0.978	0.980	0.983	0.130
9.2 Cirrhosis, fibrosis and chronic hepatitis	0.971	0.988	0.982	0.973	1.266
9.3 Other diseases of the digestive system	0.966	0.976	0.972	0.973	2.284
10. Diseases of the skin and subcutaneous tissue	0.589	0.957	0.889	0.918	0.160
11.1 Rheumatoid arthritis and osteoarthrosis	0.910	0.958	0.930	0.907	0.136
11.2 Other diseases of the musculoskeletal system/connective tissue	0.880	0.947	0.920	0.931	0.442
12.1 Diseases of kidney and ureter	0.943	0.972	0.962	0.951	1.913
12.2 Other diseases of the genitourinary system	0.961	0.980	0.967	0.979	0.585
13. Complications of pregnancy, childbirth and puerperium	0.895	0.874	0.874	0.881	0.032
14. Certain conditions originating in the perinatal period	0.973	0.976	0.975	0.976	0.520
15. Congenital malformations and chromosomal abnormalities	0.945	0.951	0.934	0.949	0.402
16.1 Sudden infant death syndrome	0.992	1.000	1.000	1.000	0.085
16.2 Unknown and unspecified causes	0.996	0.996	0.996	0.996	0.563
16.3 Other symptoms, signs, ill-defined causes	0.958	0.991	0.958	0.821	0.733
17.1.1 Transport accidents	0.981	0.994	0.171	0.978	1.740
17.1.2 Accidental falls	0.979	0.994	0.026	0.961	0.974
17.1.3 Drowning and accidental submersion	1.000	0.998	0.000	0.994	0.148
17.1.4 Accidental poisoning	0.992	0.995	0.152	0.989	1.299
17.1.5 Other accidents	0.966	0.972	0.080	0.961	0.831
17.2 Suicide and intentional self-harm	1.000	1.000	0.047	0.999	1.484
17.3 Homicide, assault	0.996	0.997	0.211	0.994	0.681
17.4 Events of undetermined intent	0.992	0.996	0.237	0.976	0.185
17.5 Other external causes of injury and poisoning	0.745	0.682	0.138	0.618	0.135
NA	1.000	1.000	1.000	1.000	0.007
Overall accuracy	0.973	0.989	0.917	0.976	nan

Fig. 7.25 Agreement at the Eurostat scale, per Eurostat chapter, for American certificates

Ecole Doctorale de Santé Publique

Titre : Epidémiologie profonde : méthodes d'apprentissage profond et leurs applications sur des bases de données médico-administratives

Mots clés : Epidémiologie, sante publique, apprentissage profond, biostatistiques

Résumé : L'exploitation des bases de données médico administratives est récemment devenue un sujet d'importance en épidémiologie et santé publique. Parallèlement, la dernière décade a été témoin de la démocratisation des méthodes dites d'apprentissage machine, et tout particulièrement d'apprentissage profond, qui propose toute une famille de puissants modèles prédictifs tout particulièrement adaptés à l'analyse de complexes interactions non linéaires dans des jeux de données autant massif que non structurés, et dont certaines ont déjà été appliquées avec succès sur des bases médico administratives par le passé. L'objectif de cette thèse est double. En premier lieu, cette thèse se veut être une introduction pour les épidémiologistes et biostatisticiens aux méthodes modernes d'apprentissage profond, à travers un travail de reformulation de ces méthodes dans une optique purement statistique, par opposition au langage cognitiviste ou computationnel auquel elles sont souvent associées. On exposera notamment que l'intégralité des méthodes modernes d'apprentissage profond, ce jusqu'aux modèles de séquence typiquement utilisés en traitement du langage naturel, peuvent s'exprimer en termes de modèles prédictifs, et notamment comme des extensions du concept de modèle linéaire généralisé. Un exemple innovant d'application de ces méthodes au problème de modélisation de variable ordinale sera également introduit pour montrer l'adaptabilité de ces méthodes à tout un spectre de problèmes régulièrement rencontrés dans des contextes de science des données médicales. Dans un second temps, ces concepts d'apprentissage profond seront directement appliqués à une base de données médico-administrative, la base du Centre d'Epidémiologie sur les Causes médicales de Décès (CépiDc) pour illustrer leur potentiel, autant d'un point de vue purement méthodique que pratique, avec des résultats variant d'une accélération de la production des données de mortalité par cause ayant notamment permis la production en temps réel de statistiques de comorbidités relatives aux décès liés à la covid-19 pendant le premier confinement, à l'élaboration d'une étude sur la question de la comparabilité des statistiques de mortalité à l'échelle internationale.

Title : Deep learning methods in epidemiology and their applications to electronic health databases

Keywords : Epidemiology, public health, deep learning, biostatistics

Abstract : The exploitation of electronic health databases has gained a significant amount of interest in fields such as epidemiology or public health. On another side, the raise of machine learning, and most specifically deep learning in the past decade has led to the development of entire families of powerful models fit to analyse complex, non-linear interactions on massive and unstructured datasets, some of whom have already been successfully applied to electronic health database analysis. The purpose of this thesis is twofold. First, this thesis is meant as an introduction to modern deep artificial neural network based models to epidemiologists and biostatisticians, by showing how these methods sometimes denoted as "algorithms" actually share a profound relationship with more traditional statistical predictive models commonly used in health sciences, such a linear or logistic regression. Finally, this thesis will showcase some practical applications of these methods on an example of real electronic health database, the Centre for Epidemiology on medical causes of death (CépiDc) database, with results ranging from the acceleration of cause of death statistics production leading to the real time production of comorbidity statistics on covid-19 related deaths during the first lockdown, to the design of a study investigating the comparability of mortality statistics at the international scale.

Maison du doctorat de l'Université Paris-Saclay
2ème étage aile ouest, Ecole normale supérieure Paris-Saclay
4 avenue des Sciences,
91190 Gif sur Yvette, France