Leveraging data structure to learn from few examples : applications to computer vision and neuroimaging

Myriam Bontonou

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THESE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPERIEURE MINES-TELECOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Signal, Image, Vision*

Par
Myriam BONTONOU

Leveraging Data Structure to Learn from Few Examples:
Applications to Computer Vision and Neuroimaging

Thèse présentée et soutenue à Brest, le 3 décembre 2021
Unité de recherche : Lab-STICC UMR CNRS 6285
Thèse N° : 2021IMTA0282

Rapporteurs avant soutenance :

Antonio MARQUES	Professeur, Université King Juan Carlos
Bertrand THIRION	Directeur de recherche, Inria-CEA Saclay

Composition du Jury :

Président :	Sophie ACHARD	Directrice de recherche, CNRS, Université Grenoble Alpes
Examinateurs :	Bertrand THIRION	Directeur de recherche, Inria-CEA Saclay
	Antonio MARQUES	Professeur, Université King Juan Carlos
	Vincent GRIPON	Chargé de recherche, IMT Atlantique
	Nicolas FARRUGIA	Maître de conférence, IMT Atlantique
Dir. de thèse :	Michel JEZEQUEL	Professeur, IMT Atlantique

Invité(s)
Pierre BELLEC Professeur, Université de Montréal

Résumé (French Summary)

La notion d'intelligence artificielle évoque généralement dans nos esprits la vision d'une machine capable de simuler l'intelligence humaine, capable par exemple de jouer à des jeux, d'interpréter des informations visuelles, auditives ou textuelles ou encore de s'adapter à des tâches complexes. Récemment, de telles machines ont commencé à apparaître grâce au développement d'algorithmes d'apprentissage suffisamment sophistiqués : les réseaux de neurones. Ces algorithmes sont maintenant utilisés dans des secteurs aussi variés que la médecine, la sécurité, l'art ou la politique.

Classifier des données avec des réseaux de neurones

L'émergence de réseaux de neurones efficaces a été rendue possible grâce à l'amélioration de l'efficacité des composants électroniques et à l'accumulation de grandes quantités de données. Dans cette thèse, nous cherchons à améliorer les performances de ces réseaux sur des tâches de classification. Nous nous focalisons sur deux applications : la reconnaissance d'objets sur des images et le décodage cérébral [Hax+01], autrement dit la détection de l'activation de fonctions cognitives particulières sur des échantillons d'activité cérébrale obtenus avec divers appareils de mesure.

Applications et défis

Pourquoi étudier ces deux applications simultanément ? La première application – la reconnaissance d'objets sur des images, est une application classique des réseaux de neurones. Les réseaux apprennent à extraire des caractéristiques pertinentes des images afin de différencier les objets. Ils apprennent à partir d'un ensemble d'images labellisées. Lorsque les réseaux de neurone sont entraînés sur un grand nombre d'exemples, ils sont très efficaces. Par contre, leur performance se détériore si la quantité de données est trop restreinte par rapport à la complexité du problème. La figure 1 illustre l'impact de la quantité de données d'entraînement sur la performance d'un réseau.

De nos jours, on cherche à entraîner les réseaux sur des jeux de données de plus en plus grands en utilisant des ordinateurs de plus en plus puissants. Malheureusement, de nombreuses applications sont limitées par la difficulté à récolter des données. Comment tirer parti des capacités d'apprentissage des réseaux de neurones dans ces cadres-là ? Comment évaluer la performance des réseaux ? Voici quelques questions étudiées dans le chapitre 2.

La deuxième application - le décodage cérébral, fait partie de ces applications pour lesquelles la récolte de données est très laborieuse. Malgré cela, quelques études ont été menées pour rassembler un nombre conséquent d'échantillons liés à des fonctions cog-

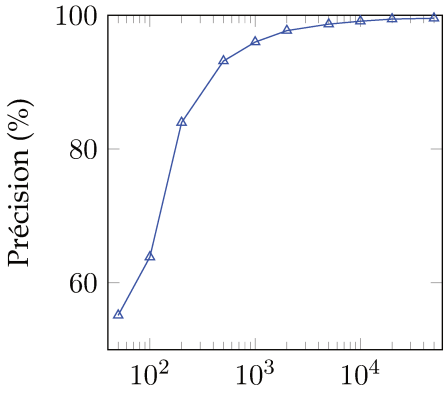


FIGURE 1 – Illustration de l'impact de la quantité de données d'entraînement sur la performance de classification. Ici, le jeu d'images utilisé est MNIST [LeC+98]. Un réseau de neurone (ResNet-18) est entraîné sur un nombre variable d'exemples d'entraînement (en gardant un nombre équilibré d'exemples par classe). La précision est évaluée sur un ensemble disjoint d'exemples du jeu de données MNIST.

nitives précises (e.g. Human Connectome Project [VE+12]). Des réseaux de neurones ont été entraînés à décoder ces fonctions cognitives sur les échantillons et leurs résultats se sont avérés prometteurs (e.g. [Zha+21b]). Dans cette thèse, nous cherchons à adapter les réseaux de neurones pour reconnaître d'autres fonctions cognitives à partir de données plus restreintes.

Historiquement, les méthodes utilisées pour décoder l'activité cérébrale ont évoluées en plusieurs étapes. Dans un premier temps, certains chercheurs ont tenté d'associer les régions cérébrales à des fonctions cognitives spécifiques (e.g. la reconnaissance des visages [KMC97]). Cependant, il s'est avéré qu'une région significativement activée dans un contexte l'était souvent également dans d'autres contextes. Pour affiner les capacités de décodage, des chercheurs ont ensuite cherché à reconnaître des fonctions cognitives à partir de l'activation relative de plusieurs régions cérébrales [Hax12].

Pour détecter des schémas d'activations permettant de différencier les fonctions étudiées, les réseaux de neurones sont des outils pertinents, à condition qu'il y ait suffisamment de données pour les entraîner. Comment exploiter les capacités des réseaux de neurones sur des tâches de décodage cérébral avec un nombre potentiellement très restreint de données ? Voici une question qui se pose dans le chapitre 3. Le chapitre 3 se base sur les réponses apportées au chapitre 2 (entraînement des réseaux dans un cadre avec peu d'exemples, évaluation des capacités apprises) pour construire de nouvelles méthodes. L'application de classification d'images nous permet de tester des méthodes d'apprentissage à partir de peu d'exemples dans un cadre relativement bien maîtrisé par la communauté. L'application liée au décodage cérébral nous permet de tester l'adaptabilité de ces méthodes à un autre type de données.

L'application de décodage cérébral a également un autre attrait : les échantillons d'activité cérébrale sont organisés autour d'une structure irrégulière complexe. Cela est un défi technique pour les réseaux de neurones. Plus précisément, les données que nous essayons de décoder dans cette thèse proviennent de l'imagerie par résonance magnétique fonctionnelle. Chaque échantillon enregistré s'apparente à une séquence vidéo contenant des images de cerveaux en 3 dimensions. Chacune des images met

en évidence les zones cérébrales actives à un instant donné. Pour réduire la taille des échantillons, chaque image est compressée en un ensemble de valeurs, chacune correspondant à l'activation cérébrale moyenne dans une région cérébrale prédéfinie. Pour réduire à nouveau la taille des échantillons, la composante temporelle est également compressée à l'aide d'un modèle linéaire généralisé [Mon11]. Finalement, les échantillons dans lesquels nous tentons de détecter des fonctions cognitives particulières se résument à un ensemble de valeurs, chacune étant associée à une région cérébrale, indiquant à quel point les régions ont été activées au cours de l'enregistrement.

Ce genre de données est dit structuré : les valeurs obtenues dans les différentes régions ne sont pas indépendantes entre elles. Au niveau anatomique, les régions sont reliées par des fibres blanches. Ce réseau de fibres est appelé connectome. La structure du connectome est dite irrégulière par opposition à d'autres structures dites régulières telles que celles des images, où chaque pixel est régulièrement entouré par 8 autres pixels, à gauche, en bas, à droite, en haut, et aux quatre coins.

La notion de structure peut être considérée de manière informelle comme l'échafaudage sous-jacent d'un échantillon. Cet échafaudage reflète diverses relations au sein d'un échantillon, telles que les similitudes, les corrélations ou les échanges dynamiques entre ses composants.

Les graphes sont des outils utiles pour représenter ce genre d'information. Les composants d'un échantillon sont modélisés par les nœuds du graphe. Les relations entre les composants sont représentées par des arrêtes entre les nœuds. Les arrêtes peuvent être pondérées par des poids. Par exemple, un graphe représentant une image contient autant de nœuds que l'image contient de pixels. Les nœuds associés à des pixels adjacents sont reliés par des arrêtes. Schématiquement, ce type de graphe a l'apparence d'une grille. Un graphe représentant les relations entre les régions cérébrales est quant à lui beaucoup moins régulier. Les nœuds sont associés à des régions cérébrales. Leur liaison symbolise un lien entre les régions sous-jacentes (densité de la connexion anatomique, co-activations fréquentes...).

En quoi la structure des données nous intéresse pour résoudre une tâche de classification ? Connaître la structure des données peut guider l'apprentissage. Par exemple, les réseaux de neurones utilisés sur les images ont été adaptés pour exploiter leur structure régulière. Cela a permis d'accroitre très significativement leurs performances. Le fait est qu'un objet apparaissant sur une image est souvent caractérisé par un groupe de pixels voisins. Si ce groupe de pixels est translaté, il représente toujours le même objet. Pour exploiter cette propriété, des filtres, appelés filtres convolutifs, sont appliqués pour détecter localement des motifs caractérisant les objets à reconnaître. Sans cette propriété d'invariance aux translations, les filtres devraient contenir beaucoup plus de paramètres pour apprendre à localiser une combinaison beaucoup plus grande de représentations d'objets.

L'exploitation de la structure des images facilite l'apprentissage. En théorie, ce devrait être également vrai pour l'exploitation de structures plus irrégulières. Mais dans ces cas-là, il est plus difficile de définir des stratégies pertinentes pour en tirer profit. Reprenons l'exemple du décodage cérébral. Nous pouvons, par exemple, considérer une tâche de classification consistant à distinguer deux fonctions cognitives. La structure des échantillons peut être représentée par un graphe dans lequel un nœud est associé à une région cérébrale et une arrête entre deux nœuds est associée à la densité des fibres blanches entre les deux régions associées. Contrairement à l'exemple des images, nous ne pouvons pas entraîner un détecteur local qui balaiera l'ensemble du graphe pour

associer des schémas d'activation similaires à des fonctions cognitives. Premièrement, comme un nœud n'est pas nécessairement connecté au même nombre de nœuds, le même filtre ne peut pas être traduit sur tous les nœuds. Ensuite, les différentes fonctions ne sont pas forcément représentées par des voisinages locaux de nœuds. Une fonction peut être représentée par un signal très diffus sur l'ensemble du graphe. De plus, un signal associé à une fonction cognitive dans une région cérébrale ne va très probablement pas se retrouver dans une autre région cérébrale sur les autres échantillons.

Pour tirer profit du graphe, il faut donc développer de nouvelles techniques adaptées aux informations a priori synthétisées sur le graphe. Tout un champ de recherche sur le traitement des signaux sur graphe [Shu+13] est en train de se développer, proposant par exemple des méthodes pour extraire des composantes pertinentes des signaux portés par les graphes.

Nous pourrions tenter de décoder l'activité cérébrale sans prendre en compte nos à priori sur les relations entre les régions cérébrales. Néanmoins, dans un cadre avec peu d'exemples, nous supposons que toute information supplémentaire pourrait permettre d'améliorer significativement les résultats.

Organisation des chapitres du manuscrit

Cette thèse s'articule autour de deux défis de l'apprentissage automatique – le peu de données et les données irrégulièrement structurées – apparaissant dans des problèmes de classification. Le manuscrit est organisé en 3 chapitres. Un chapitre préliminaire détaille les notions essentielles des domaines abordés. Un second chapitre adresse la problématique de l'apprentissage à partir de peu d'exemples. Un dernier chapitre est centré sur l'application du décodage cérébral, dans laquelle les deux défis se rencontrent. Voici un résumé de chacun de ces chapitres.

Chapitre 1 : les préliminaires

Ce chapitre introduit les domaines de l'apprentissage automatique, de la théorie des probabilités et du traitement des signaux sur graphe. Pour l'apprentissage automatique, des notions fondamentales telles que l'apprentissage supervisé et la généralisation sont expliquées. Comme nous nous intéressons principalement aux problèmes de classification, quelques-uns des classifieurs les plus utilisés sont présentés. La partie sur les probabilités présente quelques notions utiles pour comprendre la tentative de modélisation de l'activité cérébrale élaborée dans le dernier chapitre. Par exemple, les notions de variables et vecteurs aléatoires, de distribution normale, de distribution normale multivariée... Enfin, la dernière partie présente le formalisme lié au traitement des signaux sur graphe. Nous définissons les notions de graphe, de signaux sur graphe ainsi que les notions de fréquence sur graphe et de filtrage.

Chapitre 2 : apprendre et évaluer avec peu de données

Ce second chapitre réunit deux questions très importantes en apprentissage automatique : l'apprentissage à partir de quelques exemples et l'estimation de la capacité de généralisation d'un algorithme. Les expériences sont uniquement focalisées sur l'application de reconnaissance d'objets sur des images. Les images utilisées proviennent d'ensembles de données standards, collectés afin de pouvoir entraîner des réseaux neuronaux. Dans l'ensemble, les données sont faiblement bruitées : les classes sont recon-

naissables dans les images. Les techniques qui ne fonctionnent pas sur ces images ont peu de chances de fonctionner sur des données plus complexes.

En apprentissage profond, deux tendances complémentaires existent pour classifier des images à partir de peu d'exemples : apprendre à extraire des représentations pertinentes des données (plus faciles à classifier) ou augmenter artificiellement le nombre d'exemples. Ces méthodes sont souvent basées sur la notion de transfert de connaissances. Ce qui est appris dans un contexte donné est adapté à un nouveau contexte.

D'un côté, les méthodes basées sur la génération artificielle d'exemples sont souvent étudiées dans une perspective probabiliste [Cha+21]. Les méthodes de génération nécessitent de prêter attention à la nature des données. Il semble en effet relativement facile de générer de nouveaux exemples d'images naturelles en recombinant des objets entre eux, en zoomant les images, en les faisant tourner... Il semble beaucoup plus compliqué de faire la même chose avec des images provenant de l'imagerie par résonance magnétique. Du moins, il est plus facile de s'assurer que les images naturelles générées sont pertinentes, alors qu'il semble difficile de savoir si, par exemple, un échantillon d'activité cérébrale est associé de façon biologiquement plausible à telle ou telle fonction cognitive. Pour cette raison, cette famille de méthodes n'est pas étudiée dans ce manuscrit.

La deuxième tendance liée à l'extraction d'informations utiles tire souvent parti des capacités des réseaux de neurones. Dans le cadre de l'apprentissage à partir d'un petit nombre d'exemples, un réseau de neurones peut être pré-entraîné sur un grand jeu de données génériques avant d'être adapté au problème comportant peu de données. Ce réseau pré-entraîné est appelé « backbone ». Par exemple, dans le domaine de la classification d'images, un réseau peut être pré-entraîné à classifier des images appartenant aux grands jeux de données de référence tels que CIFAR-10, CIFAR-100 ou ImageNet, avant d'être adapté à un problème plus spécifique. Dans les cas les plus simples, l'adaptation consiste simplement à entraîner un classifieur avec peu de paramètres (régression logistique, plus proches voisins...) à partir des représentations des exemples récupérées dans le réseau pré-entraîné.

Les données utilisées pour le pré-entraînement peuvent provenir de problèmes similaires. Ce serait par exemple le cas pour créer une application visant à reconnaître des espèces végétales à partir d'une petite base de données. On pourrait pré-entraîner un réseau sur la grande base de données d'images ImageNet et l'ajuster ensuite. Il peut également exister de nombreux petits ensembles de données contenant des données similaires au problème à résoudre. L'entraînement d'un réseau pour apprendre à résoudre tous ces petits problèmes simultanément peut aider à apprendre à représenter de façon pertinente les données. Par exemple, dans le domaine des neurosciences, une base de données importante et diversifiée peut être constituée à partir de nombreuses études.

Pour apprendre de bonnes représentations, on peut aussi pré-entraîner les réseaux à résoudre des tâches auxiliaires sur les données. Dans la cadre de la classification d'images, la prédiction du degré de rotations d'une image préalablement pivotée est une tâche auxiliaire. Avec cette technique, il est également possible d'apprendre de bonnes représentations à partir de données non supervisées (non annotées). Par exemple, dans le contexte du traitement du langage naturel, un réseau pré-entraîné à prédire le mot qui suit le début d'une phrase s'est avéré très efficace pour prédire le sentiment des personnes à partir d'échantillons de texte sans aucun entraînement supplémentaire [RJS17]. La représentation est si pertinente qu'il n'est pas nécessaire de l'affiner.

Dans ce manuscrit, nous catégorisons les méthodes d'apprentissage à partir de peu d'exemples et nous mettons en avant leurs limites. Une de ces limites est justement liée à l'évaluation de la qualité de l'apprentissage. Rappelons qu'en apprentissage automatique, les algorithmes apprennent à résoudre un problème à partir d'exemples d'entraînement. Cependant, peu de travaux permettent d'expliquer rigoureusement comment ces algorithmes arrivent à reconnaître des classes d'objets à partir d'un ensemble fini d'exemples. Il est néanmoins crucial d'arriver à estimer de façon fiable la capacité de généralisation de ces algorithmes pour pouvoir les développer dans des technologies. Habituellement, cette capacité de généralisation est estimée grâce à un ensemble d'exemples disjoint de celui utilisé pour l'entraînement. Il est appelé ensemble « de validation ». Néanmoins, dans les cas où peu d'exemples sont disponibles, cet ensemble est également très limité, voire absent. Il est donc nécessaire de développer de nouvelles méthodes pour évaluer la capacité de généralisation.

De nombreuses mesures théoriques ou empiriques ont été proposées même s'il est difficile d'évaluer empiriquement l'existence d'une relation causale entre la capacité de généralisation et ces mesures. Comme aucune mesure n'est encore satisfaisante, des défis sont organisés à ce sujet dans de nombreuses conférences. Par exemple, à NeurIPS, une compétition a été organisée sur le thème Predicting Generalization in Deep Learning [Jia+20a]. Nous y avons proposé une mesure de complexité permettant de déterminer si les mêmes classes étaient attribuées aux exemples d'apprentissage associés à des représentations similaires au sein des réseaux – représentations apprises au cours de l'entraînement. Notre mesure, classée troisième à l'issue de la compétition, est détaillée dans cet article :

Carlos LASSANCE et al. "Ranking Deep Learning Generalization using Label Variation in Latent Geometry Graphs". In : *arXiv preprint arXiv :2011.12737* (2020)

Nous avons également publié un article mettant en avant la difficile évaluation des réseaux entraînés à partir de peu d'exemples et proposant des pistes de recherche :

Myriam BONTONOU, Louis BÉTHUNE et Vincent GRIPON. "Predicting the Generalization Ability of a Few-Shot Classifier". In : *Information* 12.1 (2021), p. 29

Dans cet article, différentes mesures de généralisation sont proposées et testées. De telle mesures ne peuvent être calculées qu'à partir des informations disponibles, c'est-à-dire qu'à partir de quelques exemples. Pour évaluer la pertinence des mesures, nous examinons leur corrélation avec la performance de généralisation sur plusieurs problèmes de classification d'images. Une des limites de l'article est justement lié à cette évaluation : l'information apportée par le coefficient de corrélation est limitée. Il ne nous renseigne pas sur la causalité entre deux variables. La mesure et la généralisation peuvent être fortement corrélées sur divers problèmes en raison de l'effet des variables cachées. Pour atténuer cet effet, nous estimons également la capacité des meilleures mesures à prédire la difficulté des problèmes. La conclusion de l'étude est mitigée. Une seule mesure prédit raisonnablement la performance de généralisation. Cette mesure estime la confiance d'une régression logistique en ses prédictions.

Ce chapitre contient des réflexions générales sur les méthodes d'apprentissage à partir de peu d'exemples. Le prochain chapitre explique comment adapter les méthodes d'apprentissage à partir de peu d'exemples à un cas pratique : le décodage cérébral.

Chapitre 3 : apprendre à décoder l'activité cérébrale

Ce dernier chapitre est consacré à une application pratique impliquant la classification de signaux complexes : le décodage de l'activité cérébrale d'un individu. Comme les réseaux de neurones ont prouvé leur efficacité sur de nombreuses applications, ils sont de plus en plus utilisés en neuroscience. Néanmoins, une limite demeure : les réseaux sont efficaces lorsqu'ils sont entraînés sur un grand nombre de données. Malheureusement, peu de données sont actuellement disponibles pour entraîner des réseaux à reconnaître diverses fonctions cognitives. Par exemple, dans ce manuscrit, nous tentons d'identifier certaines fonctions cognitives avec moins d'une centaine d'échantillons d'activité cérébrale. L'apprentissage est d'autant plus difficile que les données sont très bruitées et que l'activité cérébrale lié à une fonction cognitive varie énormément entre individus et au sein même d'un seul individu au cours du temps. La réalisation d'une même action peut passer par différents processus cérébraux et ces processus s'adaptent en permanence.

Les données utilisées dans nos expériences sont des cartes de contraste générées à partir d'images provenant de l'imagerie par résonnance magnétique fonctionnelle. L'activité cérébrale est enregistrée de façon non invasive par l'intermédiaire de la mesure du flux sanguin. La résolution spatiale des mesures est élevée (de l'ordre du millimètre) mais sa résolution temporelle est faible (de l'ordre de la seconde). L'activité mesurée lors d'une session au cours de laquelle un individu répète une même action est ensuite traitée pour générer une carte de contraste. Une carte de contraste peut être vue comme une image en 3 dimensions du cerveau composée de voxels (petits cubes). La valeur associée à un voxel représente le niveau d'activation d'une zone du cerveau durant la session. Grâce à une technique appelée imagerie de diffusion, il est également possible de modéliser le connectome à travers un graphe anatomique. Les nœuds du graphe représentent des régions cérébrales. Les poids des arêtes reliant ces nœuds représentent la densité de fibres blanches reliant les régions associées. Dans ce chapitre, nous utilisons dans nos expériences un graphe anatomique moyenné sur plusieurs individus [PVDV19].

Dans le cadre de l'apprentissage à partir de peu d'exemples, le mot d'ordre pour résoudre les problèmes avec peu de données est utiliser d'autres sources d'informations. Deux tendances se distinguent :

- la première est le transfert de connaissances à partir d'autres ensembles de données recueillis pour résoudre des problèmes similaires, comme le montre le chapitre précédent ;
- la seconde est la modélisation de la distribution des données en utilisant des connaissances a priori sur le problème à traiter. Le but est d'utiliser le modèle pour trouver une manière optimale de distinguer les classes.

Dans ce manuscrit, nous proposons tout d'abord de faire face au manque de données en suivant la première tendance. Nous adaptons les techniques développées sur les images classiques à ces signaux plus complexes. Pour cela, nous définissons un jeu de données générique regroupant des cartes de contraste issues d'une étude française intitulée Individual Brain Charting (IBC) [Pin+18 ; Pin+20]. Bien que chaque classe soit représentée par un petit nombre de cartes de contraste (entre 21 et 78), de nombreuses classes y sont étudiées en même temps. Nous supposons qu'un backbone préentraîné sur des classes couvrant des domaines cognitifs variés apprendra à extraire des informations variées des données, et donc potentiellement utiles pour distinguer de nouvelles classes. Notez qu'en pratique, nous compressons les cartes de contraste en

moyennant leurs valeurs sur 360 régions cérébrales (d'après l'atlas classique de Glasser [Gla+16]). Nous réduisons ainsi la taille de chaque exemple de plusieurs centaines de milliers de valeurs à 360 valeurs.

Dans une de nos expériences, nous pré-entrainons un petit réseau (un perceptron multicouche) en utilisant deux méthodes d'apprentissage différentes. La première méthode, basée sur le transfert d'information, pré-entraîne le petit réseau à distinguer de nombreuses classes à la fois. La deuxième méthode, inspirée du meta-apprentissage, apprend les paramètres initiaux du réseau de manière à faciliter la résolution d'un grand nombre de problèmes différents. Pour évaluer l'utilité des méthodes d'apprentissage, les classes de IBC sont séparées en deux ensembles. Le premier ensemble dit « de base » contient un grand nombre de classes pour pré-entraîner les réseaux. Le second ensemble de classes est utilisé pour générer des problèmes avec peu d'exemples d'entraînement. Dans notre étude, les performances des réseaux sont bien meilleures que celles d'une régression logistique entraînée directement sur données. La méthode de transfert obtient également de meilleurs résultats que la méthode inspirée du méta-apprentissage. Un article présentant cette contribution a été publié :

Myriam BONTONOU et al. "Few-shot Learning for Decoding Brain Signals". In : *arXiv preprint arXiv :2010.12500* (2020)

Dans un second temps, nous proposons d'exploiter la structure intrinsèque des cartes de contrastes en tirant profit des connaissances apportées par le graphe anatomique. Comme mentionné précédemment, les réseaux de neurones peuvent obtenir de très bonnes performances de classification sur les images grâce à l'utilisation de couches convolutives apprenant à détecter des combinaisons de motifs sur les images. Dans notre cas, l'activité cérébrale représentée sur les cartes de contraste n'a pas une structure aussi régulière que les objets sur les images. Les signaux liés à l'activité cérébrale peuvent être très diffus.

Ici, nous exploitons le fait que les informations se propagent dans un cerveau à travers les fibres blanches pour modéliser la structure des données. Nous supposons que les dépendances entre les signaux de différentes régions sont partiellement représentées par le graphe anatomique. Nous suivons deux protocoles différents pour d'exploiter ces connaissances préalables : 1/ en insérant un filtre exploitant ce graphe dans les réseaux précédents, 2/ en modélisant la distribution des données. La première approche est plus proche de l'apprentissage profond et la seconde est plus liée au traitement statistique du signal.

Pour notre première tentative, nous proposons d'utiliser un filtre consistant simplement à diffuser les valeurs associées aux régions sur le graphe anatomique, de sorte à moyenner le signal associé à une région avec celles des régions voisines. Néanmoins, la présence du filtre n'améliore pas les résultats. Cela est peut-être dû au fait que la division de l'image en régions cérébrales est très grossière (il n'y a que 360 régions).

Pour notre deuxième tentative, nous modélisons la distribution des données l'aide d'un modèle relativement simple. Le modèle représente chacune des classes à l'aide d'une carte de contraste de référence. Les cartes observées sont supposées être des versions bruitées de ces références. Le bruit est en partie défini à partir de nos connaissances préalables sur les relations entre les régions du cerveau. Nous supposons que chaque carte de contraste résulte de l'ajout de deux types de bruit à la carte de référence : un bruit isotrope et un bruit dépendant du graphe. Le bruit isotrope modélise les sources externes de bruit liées par exemple à l'acquisition des données. Le bruit dé-

pendant du graphe rend compte des relations intrinsèques entre les régions du cerveau. Dans ce modèle, chaque classe suit donc une distribution gaussienne multivariée. Il est notoire que ce type de problème de classification est résolu de manière optimale par une analyse discriminante linéaire. L'analyse discriminante linéaire est équivalente à un blanchiment des données suivi d'un classifieur centroïde le plus proche. Si le graphe était inconnu, le blanchiment des données nécessiterait d'estimer la matrice de covariance, ce qui est difficile avec peu d'exemples. Cependant, grâce au modèle, la matrice de covariance est estimée à partir du graphe avec un seul paramètre. Nous comparons les résultats obtenus à l'aide de notre modèle avec ceux d'un classifieur centroïde le plus proche, d'une régression logistique et d'une analyse discriminate linéaire pour laquelle la matrice de covariance est estimée à l'aide d'une technique conçue pour les cas où peu de données sont disponibles [Che+10]. Nous montrons que le prétraitement permet d'améliorer les performances du classifieur centroïde le plus proche et de la régression logistique sur les données d'entrée mais qu'il ne fonctionne pas mieux que l'analyse discriminante. Nous concluons que le modèle utilisé n'est peut-être pas assez riche pour représenter la complexité des données. Il est également possible que le graphe anatomique moyen que nous utilisons ne représente pas les corrélations entre les régions. Ce travail est en cours de publication et déjà présent sur arXiv :

Myriam BONTONOU, Nicolas FARRUGIA et Vincent GRIPON. "Graph-LDA : Graph Structure Priors to Improve the Accuracy in Few-Shot Classification". In : *arXiv preprint arXiv :2108.10427* (2021)

Conclusion

En résumé, cette thèse est organisée autour de la question suivante : peut-on apprendre à partir de quelques exemples en transférant des connaissances et/ou en exploitant la structure des données ? Nous nous concentrons principalement sur les problèmes de classification d'images naturelles et de neuro-imagerie.

Que faire si nous ne disposons que d'un petit nombre d'exemples d'apprentissage ? Lorsque de nombreux exemples d'apprentissage sont disponibles, les méthodes de classification à l'état de l'art contiennent souvent des réseaux de neurones. Néanmoins, entraîner des réseaux à partir d'un petit nombre d'exemples est très risqué : ils sont susceptibles de s'adapter de manière excessive aux quelques exemples sans réussir à généraliser la notion de classe à de nouveaux exemples. C'est pourquoi des approches dédiées aux applications avec peu d'exemples ont vu le jour. Dans certaines de ces approches, des réseaux sont pré-entraînés sur des jeux de données génériques avant d'être combinés avec des classifieurs avec quelques paramètres supplémentaires à apprendre et/ou si possible, avec des connaissances a priori sur les nouveaux problèmes. Dans cette thèse, nous montrons comment tirer parti des réseaux de neurones pour extraire des informations pertinentes des données. Nous mettons également en avant la problématique liée à l'évaluation des classifieurs lorsque les données disponibles sont très restreintes.

Que faire si les données présentent une structure irrégulière ? Par exemple, si nous disposons d'échantillons de données représentant l'activité dans diverses régions cérébrales d'un individu, nous savons que certaines régions sont plus connectées que d'autres. Comment pouvons-nous exploiter cette information ? Et devons-nous nécessairement l'exploiter pour obtenir de bonnes performances ? En théorie, les réseaux de

neurones peuvent apprendre à partir de données brutes sans exploiter d'informations supplémentaires. Cependant, les performances des réseaux sont bien boostées quand on peut exploiter des informations sur la structure des données (e.g. les convolutions pour les images). Pour des données ayant une structure plus irrégulière, de plus en plus de travaux tentent de combiner les opérations de filtrage définies dans le domaine du traitement des signaux sur graphe avec l'entraînement des réseaux de neurones.

Ouverture

L'apprentissage profond s'avère être un outil fantastique. Il a un énorme potentiel pour aider les êtres humains dans de nombreux aspects de leur vie. Les chercheurs travaillent actuellement sur des outils alimentés par l'intelligence artificielle, tels que des traducteurs automatiques ou des assistants à la décision capables d'analyser rapidement diverses sources de données. Cependant, pour développer de telles applications, le domaine de l'apprentissage profond doit relever plusieurs défis, dont l'adaptabilité à la quantité de données disponibles. La quantité exacte de données nécessaires pour résoudre un problème donné de façon optimale est souvent inconnue. C'est pourquoi la devise de l'apprentissage profond est la suivante : plus il y a de données, mieux c'est ; bien qu'au-delà d'un certain seuil, l'ajout de données supplémentaires entraîne souvent un gain de performance minime.

Néanmoins, la capacité à apprendre à partir de peu de données est cruciale pour de nombreux domaines. En médecine, rassembler des bases de données contenant les symptômes de toutes les maladies connues serait déjà une tâche énorme. Cependant, même si nous avions accès à de telles données, de nombreuses maladies sont encore inconnues et de nouvelles épidémies apparaissent régulièrement. Il pourrait être utile d'apprendre à les détecter à partir de quelques exemples. On peut imaginer un monde où de grands réseaux dirigeraient les diagnostics vers des algorithmes plus spécialisés. Les algorithmes spécialisées seraient peut-être eux-mêmes entraînées à partir de grands réseaux rassemblant une partie des connaissances humaines.

Table des matières

Acknowledgments

I thank all the people who helped me during my PhD.

In particular, I thank my supervisors Vincent Gripon and Nicolas Farrugia for having guided me, advised me and allowed me to discover the international world of research through an exchange in Montreal and numerous conferences. I thank my thesis director Michel Jézéquel for his support as well as Bastien Pasdeloup who generously provided me with the latex template for this manuscript.

I also thank the entire BRAIn team and the entire MEE department for the good atmosphere and the dynamism that prevails.

Finally, I deeply thank my partner Julien M'barek for having supported me all these years and also for having helped me to illustrate this thesis.

"Science sans conscience n'est que ruine de l'âme." Rabelais

"Un vrai maître est un éternel élève." Anonymous

Introduction

Contents

ARTIFICIAL Intelligence (AI) is generally seen as a machine able to simulate human intelligence, able for instance to play games, to interpret visual, auditory or text information or even to achieve tasks in unfamiliar and erratic places. Since the 2010s, discussions about AI have emerged in many sectors such as medicine, security, art or politics. The hype is explained by the fact that at that time, researchers managed to develop learning machines sophisticated enough to be used in real applications.

This introduction is designed in several parts. First, we present the historical steps of AI progress as they appear in many sources [1]. We describe current AI applications as well as society's expectations and fears. We then put forward the problematic studied in this thesis, explaining its concrete interest for the current world.

Brief History of AI

AI technologies are based on decades of previous research. Indeed, AI has been in the thoughts of human beings for a very long time. Homer in the Iliad describes automata forged by Hephaestus to assist him in his work. Some have designed real automata that mimic the functioning of humans and animals (such as the Digesting Duck of Jacques de Vaucanson able to drink water and flap its wings).

Note that these automata were not strictly speaking AIs: they were not able to learn. Indeed, learning is not simply the capacity to memorize an action and to replay it. Learning means being able to generalize what has been experienced so far and to adapt it to a new situation. For instance, let us consider an example in which a machine observes many images with different labels. In this case, memorizing means that the machine is able to reproduce the labels of the images it has observed. On the other hand, generalizing means that the machine is able to predict the label of a new image.

In the 1950s, with the emergence of computers, the creation of an AI capable of matching the cognitive abilities of human beings seemed increasingly imminent. However, AI research has gone through periods of optimism interspersed with periods of disappointment. In 1958, one of the first learning algorithm was invented by the psychologist Frank Rosenblatt. This algorithm, called perceptron, imitates the functioning of a single neuron. It teaches itself to solve a problem by trial and error on a training dataset. For instance, it can learn to solve a classification problem in which two classes must be separated. The perceptron is described in Fig. 2. Its input is a vector (made of real values). Each component of the vector is multiplied by a certain parameter (another real value). If the sum of the weighted components exceeds a threshold value, the perceptron fires. In other words, it emits a bit. In that case, the input is associated with one of the class. In the other case, when nothing is returned, the input is associated with the other class. What was remarkable at that time was that the perceptron learns to recognize the labels on a training dataset by adapting its parameters. If its output is wrong, the parameters are updated so that the output becomes closer to the true answer. The same mechanism is repeated on all the training examples until they are all properly classified.

The unprecedented learning capacities of the perceptron are however limited. The perceptron can only divide the input vector space with a hyperplane. In other words, in a one-dimensional space, inputs upon a certain value belongs to one class and vectors

1. For instance, in `https://en.wikipedia.org/wiki/History_of_artificial_intelligence` or in Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016

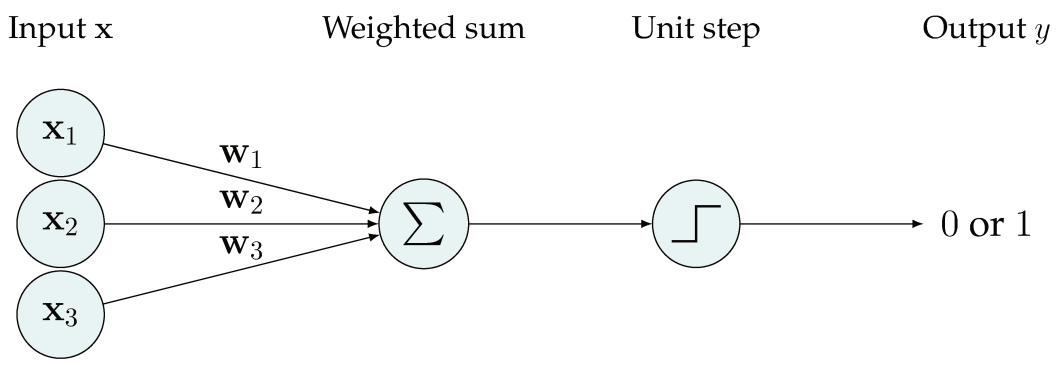


Figure 2 – Description of the perceptron. In this example, the input $\mathbf{x}$ has 3 components.

below the same value belong to the other class. In a two-dimensional space, classes are separated by a line. In a three-dimensional space, by a plan and so on. When classes cannot be separated by a hyperplane, the problem cannot be solved. Partly because of that, the research on learning algorithms has stagnated for many years.

Meanwhile, the notion of AI moved in another direction. Researchers started to conceive machines for which the reasoning followed a set of symbolic rules. Their expert systems mimicked the ability of a human expert in particular fields. The limit of these systems was the acquisition of the set of rules. Despite that, they are commonly used as decision support tools for instance to diagnose infectious disease or to monitor industrial plants from sensor data.

The machine learning field re-emerged in the 1980s by proposing models inspired from probability theory to tackle specific problems. For instance, the support vector machines were designed to separate two classes with a maximal margin. In parallel, the efficiency of a new generation of learning machines inspired from the perceptron has been demonstrated. The natural extension of the perceptron is the Multilayer Perceptron (MLP). It consists in a succession of linear functions interleaved with non-linear functions. An example of MLP is shown in Fig. 3. From the input vector, several linear functions with separated parameters are computed. The output of each linear function goes through a non-linear function. Then, the obtained outputs are used as input for new linear functions. This goes on until the last layer. The parameters of the different layers are updated with a mechanism based on the backpropagation of the gradient of the output error. With this architecture, the MLP can learn non-linear decision boundaries. The algorithm has then been improved, notably to efficiently learn convolutional filters and recurrent relationships.

These new learning algorithms are known under the general name of neural networks or Deep Neural Networks (DNNs) when they contain more than one hidden layer of linear functions between the input and the output. An entire subfield of machine learning called deep learning is dedicated to the study of Deep Neural Networks (DNNs). Since then, DNNs have been adapted in many applications.

AI Applications

One of the first successful applications of neural networks was the automatic recognition of handwritten numbers [LeC+98], successfully applied in reading zip code numbers and bank checks. Then, thanks to better hardware equipment, DNNs succeeded to

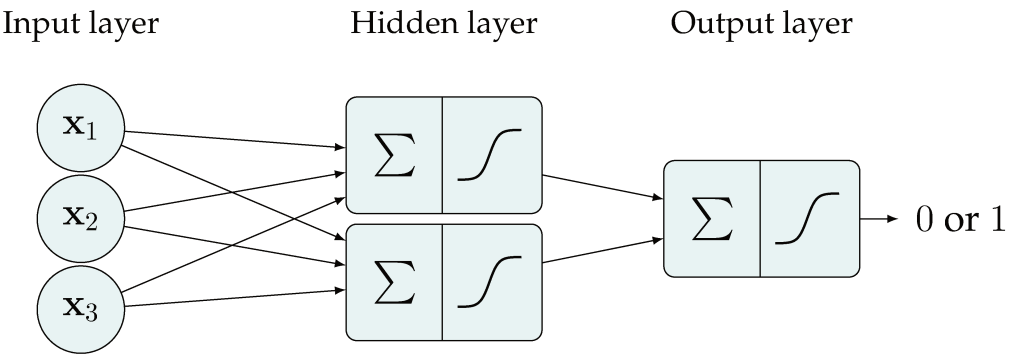


Figure 3 – Extension of the perceptron into a Multilayer Perceptron (MLP). In this example, the MLP contains 1 hidden layer with 2 features. The unit step function is replaced by sigmoids to make backpropagation possible.

reach human performance on a large image classification dataset in 2015 [He+15]. Since then, they have been adapted to many applications. Note that most researchers are not trying to make an overpowerful AI outperforming humans in every field. Current AIs are specialized for particular applications.

Recent advances

For natural language processing, AI helps human beings to better communicate with each other. New products range from automatic spell checkers, to efficient automated translators such as DeepL, to voice assistants able to create videos of a synthetic character speaking from written text (see for instance the Hour One company), or to chatbot helping users to be directed towards a relevant information source. Figure 4 illustrates the ability of DNNs to generate coherent text.

For image processing, AI also has a huge success: image classification, segmentation, object detection have no more secrets for the machines. They routinely outperform humans. AI can be used as a medical assistant, for instance helping radiologists to detect anomalies on medical images. Another example is the AI-powered dermatology assist tool that Google is developing [2] to help people identifying dermatological issues with their phone's camera. Their tool is based on a DNN released in [Liu+20] whose performance is on par with U.S. dermatologists.

AI is also applicable to the art domain. The style of a painter can be transferred to an ordinary photograph as shown in Fig. 5, and even to movies [RDB18]. Written texts can be analyzed for authorship by identifying distinctive patterns in the way a person writes. Such a process has been used on the works of William Shakespeare and John Fletcher [Ple19], for which researchers attempted to answer the famous debate of who between Shakespeare and Fletcher wrote passages of theater pieces attributed to Shakespeare. Other researchers have also tried to model music composition [Col21], by learning for instance the style of Beethoven and trying to complete Beethoven's last symphony from partial sketches of musical ideas Beethoven left over his life. For now, the results are often mitigated but this is an interesting direction of research.

In the game field, AI also had a few major successes [Sch+20]: algorithms managed to beat world champions in chess and go games.

2. Link to a blog post where the tool is described: `https://blog.google/technology/health/ai-dermatology-preview-io-2021/`

Artificial intelligence is not only changing our lives, it's shaping our museums.

An artificial intelligence machine recently played an important role at the Detroit Institute of Arts. A researcher on the Detroit Institute of Arts team who helped the museum develop the program hopes it will lead to exciting developments in the future.

"Right now we're using AI to make computer programming cheaper, more accessible, and we're using it to explore emerging technology in art," said PhD student Chuxin Pan. "There's a huge opportunity for AI to start transforming our entire museum experience."

Pan was a part of the research team who developed a 3D machine learning computer program called PigBot. PigBot was trained to play an ancient Chinese board game called Go. The program can play at a level five under current computer programs, but only under the guidance of a human instructor.

Figure 4 – Example of text generated by a DNN trained on millions of web pages. We wrote the first words in bold. The highlighted words were generated by the DNN [3]. Even if the text content does not always make sense, the structure is coherent.

(a) Style image [4]. **(b)** Content image. **(c)** Generated image.

Figure 5 – Example of photograph adapted by a DNN. The content of an image is reproduced with the style of another image [5]. Here, the color palette has been transferred but the face is too dark.

Societal impact

AI promises many more advances to society. For instance, in many countries, there is a shortage of doctors; the actual doctors are overworked and the time it takes to get an appointment is longer and longer. AI-based technologies could be adapted to assist doctors in their work by for instance handling a wide range of data to diagnose, predict the evolution of diseases, and evaluate the best available treatments for patients. The research for new medicines would also probably be accelerated by extrapolating from the chemical properties of the molecules [Jum+21].

Many other motivating applications are currently being researched. Robots could

3. This text was generated on a website called InferKit.
4. Part of a painting entitled self portrait at the age of 34 by Rembrandt.
5. Image generated with PyTorch [Pas+19] according to the method developed in [GEB15].

replace humans to perform dangerous tasks for instance on sites contaminated by radiation. Some work on automatic translators allowing in real time to translate a French speaking voice to an English speaking voice with a particular British accent. Others try to use AI to better manage the resources of the planet. Some labs propose to automatically generate 3D maps of places from drones capturing images on their way. From an entertainment perspective, AIs could also extract artistic styles from novelists and apply them to invent new stories.

However, advances in AI also induce fear in many people's minds. For instance, widespread surveillance of citizens becomes easier with technologies such as automatic facial recognition. In an optimistic perspective, facial recognition can help to make the society safer and more respectful by identifying criminals or detecting deviant behaviors (throwing away cigarette butts on the ground, not following the road safety rules...). In a pessimistic perspective, it could be used to control the actions of all citizens and restrict their freedom. This is not science-fiction. China is deploying a social credit system already in a test phase. The idea is to give citizens a certain number of points. If they do good deeds, their score goes up. If they break the rules (non-compliance with traffic regulations, immorality, incivility, late payment of taxes...), their score goes down. This system foresees penalizing individuals with, for example, travel bans.

AI also plays a dual role in the climate crisis [Dha20]. On one hand, AI helps to predict climate change and to better manage resources. On the other hand, AI systems require a lot of resources: metals need to be extracted for hardware, a lot of energy is needed to handle data and train algorithms... Although efforts are made to reduce this impact, this is not in line with the need to reduce the impact of humans on earth.

Finally, there is also the fact that a few private companies such as Google, Facebook, Amazon, Microsoft or Baidu own the data of thousands of individuals all over the world. They have the power to manipulate these people on their platforms (e.g. by influencing the information reaching them). Nothing prevents them from using this data for malicious purposes. This can influence political decisions and threaten democracies. A precedent that has unleashed the press is related to the company Cambridge Analytica. It had access to the data of millions of Facebook users' profiles. It is suspected of having used this data to create psychological profiles of voters and of having used these profiles to influence the outcome of the 2016 American election campaign and the Brexit referendum.

To counteract the drifts linked to the use of AI, charters for responsible AI are starting to emerge. One of them is the Montreal Declaration pushed by Yoshua Bengio (2019 Turing award winner for his pioneering work on deep learning) resulting from discussions with scientists and the public. An example of a recommendation would be to prevent research on armed drones that automatically aim at their targets, although it is difficult in practice to impose such a decision on all countries. Europe also passed a data protection law in 2016: the general data protection regulation. It regulates the use of data related to people in the European Union.

In this thesis, we bring methodological contributions to deep learning to improve the current capabilities of the machines. Nevertheless, we are committed to work in an ethical framework respecting individuals and the planet.

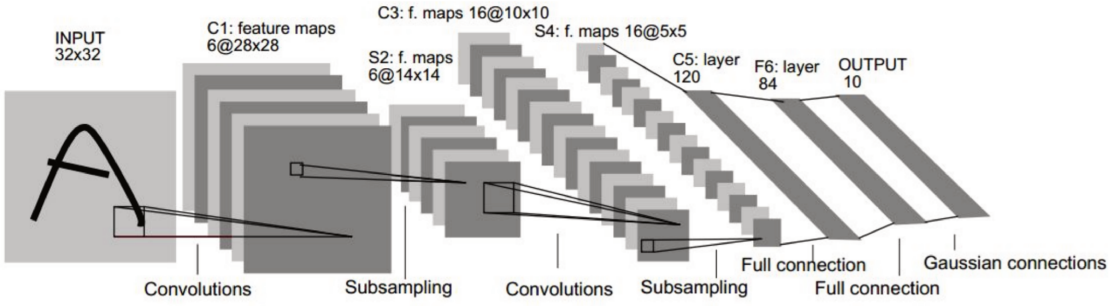


Figure 6 – Architecture of LeNet-5 [LeC+98] ©1998 IEEE.

Focus on Deep Learning

DNNs are the algorithms that made the above applications possible. As this thesis is mainly focused on the ability of DNNs and other machine learning algorithms to learn from supervised data, it is appropriate to dwell on that before going further. We mentioned in the section dedicated to a brief history of AI that DNNs took off around the 2010s. Their emergence was linked to three factors: the increase in computing power (e.g. graphics processing units), the use of efficient algorithms (e.g. convolutional layers, backpropagation) and the access to large amounts of data (mainly thanks to the Internet).

The architecture of the neural networks has evolved since the first MLPs. Numerous types of layers and various architectures have been designed to learn more efficiently. For instance, one of the first DNNs learning convolutional filters is displayed in Fig. 6. It has been designed to classify images. It contains 3 convolutional layers learning convolutional filters and 1 fully connected layer.

In supervised learning, notably in classification, datasets are gathered to train the algorithms. These datasets contain data samples associated with labels. Algorithms learn to recognize labels from data by inferring general patterns characterizing the labels. In the case of DNNs, these patterns are learned with trial and error on a training dataset. The power of DNNs over other learning algorithms is that they are able to learn rich representations of the data, enabling to separate the classes with linear boundaries. For instance in image classification, the state-of-the-art DNNs contain a succession of convolutional layers. In each convolutional layer, convolutional filters learn to detect relevant patterns. The first layer learns to recognize basic patterns from the image (such as straight lines of a few pixels), then it passes on the extracted data to the next layer and so on. The last layer learns complex combinations of patterns, which allows class recognition.

Here are two examples of datasets which have been created to train machines. For the handwritten digits recognition application previously mentioned, a dataset had already been gathered in 1998. It is the ancestor of the famous MNIST dataset [LeC+98] still used today in machine learning tutorials. MNIST contains 70000 black-and-white 28 x 28 images on which a handwritten digit is centered. This dataset has been created from digits written by high-school students and United States Census Bureau employees. From 2006, to expand and improve the data available to train AI algorithms, a very large dataset called ImageNet [Rus+15] has been gathered for image classification and object detection. The images are photographs collected from Flickr and other search engines. From ImageNet was born the ImageNet Large Scale Visual Recognition Chal-

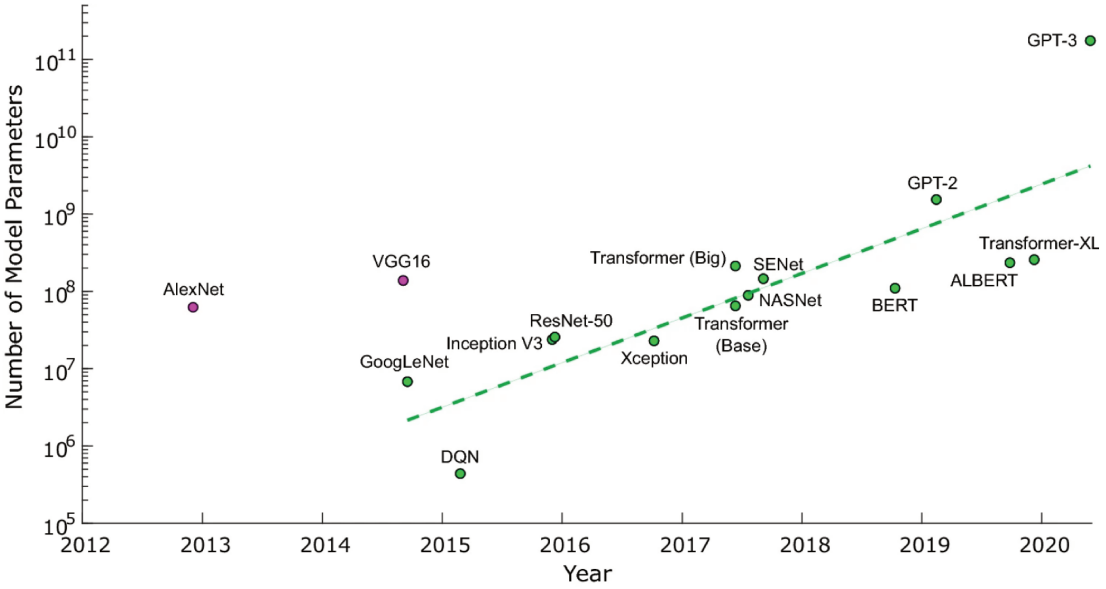


Figure 7 – Number of parameters in recent DNNs. Reproduced from ©2021 Bernstein et al., CC-BY-4.0 [Ber+21].

lenges (ILSVRC), where many labs competed to improve the algorithms performance. In the challenges, subsets of ImageNet were used. This competition contributed to make the DNNs popular in 2012 when a DNN succeeded to drastically reduced the classification error rate [KSH12]. Nowadays, ImageNet is still widely used to compare algorithms performance in image classification and object detection. One of the most widely used subset spans 1000 object classes and contains about a thousand images per class. Since then, many other free high-quality datasets have been gathered for various applications to train AI algorithms.

State-of-the-art DNNs are trained on increasingly larger datasets and powerful computers. Figure 7 indicates that they tend to contain more and more parameters. For instance, recent natural language generation models contain billions of parameters (e.g. 175 billion for GPT-3 [Bro+20]). In image classification, the number of parameters of state-of-the-art DNNs such as ResNet-50 is also very large.

The fields of machine learning and especially deep learning are very promising for a lot of great applications but there are still improvements to make. That is why they are receiving significant funding from public and private partners to apply the algorithms to particular problems, to collect more data, and, most importantly, to try to overcome the many obstacles the field faces. For instance, in 2019, France has announced the creation of four specialized AI research centers in Grenoble, Nice, Paris and Toulouse, for which it has released 75 million euros of public funds.

Challenges

Now that we have completed our overview of AI, here are a few examples of challenges that need to be solved to go further.

Data acquisition

One of the main challenges is linked to data acquisition. Indeed, the effectiveness of a DNN depends on the quality of the data (reliable, balanced between classes) and its quantity. Unfortunately in many scenarios, the datasets are not representative of the real-world data distribution. They are small, noisy or contain few examples of particular cases. For instance, in the medical field, some data are really difficult to obtain either for privacy reasons or because it is too expensive to annotate a supervised dataset. Another example regards entities which gather private datasets to solve very specific problems such as detecting anomalies on their networks. Such datasets may contain many examples of situations labeled as normal for only a few examples of anomalies. Unbalanced datasets can also raise ethical issues. For instance, if a network is trained on thousands of private datasets to decide whether the bank can extend credit to a person without taking too much risk, the algorithm will learn to make good decisions for major subsets of the population and will not be properly evaluated on minority subsets.

Interpretation

Another challenge is linked to the design of the DNNs themselves. DNNs are known as black boxes. Indeed, once they are trained, they produce a solution but they do not explain their decision. Given an image classified as a cat for instance, one would like to know why the DNN chose a cat among many other animals. Is it because of the ears, the nose, the mustache or is the decision based on a completely irrelevant factor (color, background representing a living room while the other animals are outside...)? In some applications, the ability to explain the prediction is crucial. For instance, if AIs were used as decision-support tools in medicine, a doctor would like to know why the machine thinks that this image is a tumor or that this patient has a certain disease. Especially if the doctor disagrees with the machine, it would be important for him to know the reasons behind the machine's choice. It would also be interesting to know which features are more important for a machine to make such a decision more than another one.

Robustness

There are also a lot of issues with the robustness of the decision of trained DNNs. A DNN is trained on a given dataset, with a given distribution. When an unexpected input comes, there is no guarantee that the system will work correctly. That is, for instance, a problem for advances in autonomous driving. What if a red balloon appeared in front of a self-driving car? Will the car interpret the balloon as an unknown hazard, as a stop sign or as something really different? It has also been shown that adversarial attacks could severely harm the DNNs' predictions. By modifying the values of a few pixels, invisible for a human eye, the DNNs' prediction can be completely modified [GSS15]. The face recognition tools can also be fooled by natural makeup [Gue+21].

Multi-modal data

Another challenge is to successfully exploit the different information contained in the multi-modal data simultaneously. In medicine once again, data can take a vast range of shapes, from blood measures, to medical images, genetic data or even behavioral data. This is also the case in applications related to global warming forecasting or crop management where satellite images, radar images, ground temperature sensors and numerous other sources of data are used.

Irregularly structured data

DNNs are currently really efficient on regularly structured data thanks to the use of convolutions (images) or sequence-to-sequence networks (natural language processing). However, it is still a challenge to take into account an irregular noisier structure.

The notion of structure can be informally considered as the underlying scaffolding of a signal. This scaffolding can reflect various relationships within a data sample, such as similarities between components, correlations or message exchange. Graphs are useful tools for representing the structure of a data sample. The components of the data sample are represented by the nodes of a graph. The relationships are represented by (weighted) edges between these nodes.

For a temporal sequence, reporting for instance the evolution of the measurement of a sensor, a natural structure is the timeline. The measurement at a point is preceded by the measurement of the previous time point and followed by the measurement of the next time point. For an image, a natural structure can be a grid in which each pixel is a node and the edges of the grid connect neighboring pixels (above, below, right, left, top left, top right, bottom left, bottom right). For social networks, each node can represent a user and the edges can represent connections between users (friendship, colleagues...). For brain activity, cerebral regions can be grouped into bigger regions sharing similar activity. A graph can be used to represent the interactions between regions.

Structure and signals are complementary to understand the phenomena. The learning of convolutional filters on images is one of the best examples highlighting the interest brought by the data structure. An object appearing on an image is often characterized by a group of neighboring pixels. If the group of neighboring pixels is translated, it still represents the same object. A convolutional filter is used to learn to locally recognize an object on an image. It is applied as a local detector that scans the different areas of the image. Without translation invariance, the detector filter would have to learn to locate a much larger combination of object representations. Note that to overcome the problems associated with the scale of objects in images, the structure is once again exploited: successive layers of filters and pooling layers (to zoom out) are applied to the image.

When the data structure is irregular data structure, the problems are often more complex. For instance, let us consider a brain decoding application where we try to distinguish brain activities linked to two different cognitive tasks (e.g. a memorization task and a motor task) and where we have access to a graph representing the anatomical connections between brain regions. It appears that we cannot directly learn a local detector that will scan the whole graph to detect similar patterns of activation. First, as a node is not necessarily connected to the same number of regions, the same filter cannot be translated on all nodes. Second, it is not necessarily desirable to define a local filter. Indeed, a task can be represented by a very diffuse signal on the whole graph. It is not necessarily local. To take advantage of the knowledge of different kinds of structure, an entire research field called Graph Signal Processing (GSP) [Shu+13] works on various solutions, proposing for instance to extract relevant components from the graph signals.

All of these challenges and many others are actively studied by many research groups and industries.

Emphasis on Few-Shot Learning

In this thesis, we are really focusing on one of these issues: the lack of labeled data. Problems with few labeled examples are known as few-shot problems. For the AI community, the issue related to the lack of labeled data is not a new problem brought by deep learning [Thr98]. Algorithms need to be trained on data, which must be annotated in the case of supervised learning.

Need for labeled data

To address the need of labeled data, numerous datasets have been gathered and made public over the years. The creation of such datasets has been greatly helped by the democratization of the Internet. On the Internet, millions of people share photographs, write blogs, comment on movies... The majority of this data is not directly usable for supervised learning but it can be preprocessed.

For instance, a large dataset has been built from Amazon reviews [HM16]. It has been used to learn to assess a person's satisfaction from text. We also previously mentioned ImageNet [Rus+15], one of the biggest labeled dataset for image classification. It has been built from photos shared on search engines such as Flickr. As thousands of images needed to be annotated with a label and a bounding box indicating the position of the object, a crowdsourcing website called Amazon Mechanical Turk has been used to hire people remotely to do the job.

Many datasets are also shared through competitions hosted on platforms such as Codalab. Through these challenges, an entire community cooperates (or rather competes) to solve a given problem. The organizer is responsible for formatting the data, detailing the problem and fairly evaluating the proposed solutions. The participants "only" have to tackle the problem. Such competitions are really valuable: they showcase the most efficient methods on particular problems and they help to make data accessible to research labs. Big data companies also have a significant advantage over others: they have huge servers storing a large amount of Internet data.

Sometimes, as was the case for ImageNet, remotely working people are hired to annotate the data (associate a class to an image, transcribe speech into text...). Companies such as Amazon propose online platforms to bring companies and workers together. Other research laboratories encourage internet users, researchers and students to annotate data for free [Rus+08]. Annotating a large dataset is a time-consuming task for a single person but it becomes realistic for a large community of people.

However, for certain applications, such annotation mechanisms cannot be set up. Data annotation requires the participation of a human expert, which is expensive. For instance, people without training cannot identify a tumor and segment it on a medical image. A similar case appears in the study of wildlife. DNNs could be trained to recognize animals from sound recordings. To gather a dataset, a first step would consist in placing microphones in the wild. A second step would require the help of human experts to identify animal sounds.

Finally, in certain areas, collecting data in itself is expensive, time-consuming or simply not possible. For instance, in neuroscience, some researchers try to decode human thoughts from brain activity. One of the techniques enabling the recording of brain activity with a good spatial resolution is functional Magnetic Resonance Imaging (fMRI). To record data, a person must enter a scanner and perform a cognitive task. The record-

ing time varies from a few minutes to about 1 hour. So it takes time to build up a dataset. The cost of the scanner and its maintenance is not to be neglected either. In the case of facial recognition, it would be useful if an algorithm could learn to recognize a person from a few photos (for instance, to locate kidnapped children or to identify wanted people).

Current methods for learning with few examples

What is problematic is that DNNs can rarely generalize from a few examples. Indeed, the efficiency of DNNs is related to their ability to learn relevant data representations, which requires tuning millions of parameters. A representation is the image of an input data learned within a DNN. For instance, in the case of Convolutional Neural Networks (CNNs) trained on image classification problems, different representations of an input image are obtained after different layers. The first layers extract general information from the image (basic patterns, lines...) while the last layers recognize the presence or absence of specific shapes related to the underlying classification problem. To learn relevant representations, state-of-the-art DNNs are trained on very large datasets. If their numerous parameters were trained from scratch on a few labeled examples, the DNNs would probably not perform well on new data.

To tackle learning problems based on few labeled examples, two complementary trends exist for natural image classification: learning good representations in a less traditional way or artificially increasing the number of examples. Such methods are often based on the notion of knowledge transfer. What is learned in a given context is adapted to a new context.

Learning relevant data representations generally consists in pre-training a DNN to perform certain tasks on diverse data. To solve the few-shot problem, a relatively simple classifier (i.e. with few parameters to learn, such as a logistic regression or a nearest class mean) is trained on top of the extracted representations of the new inputs processed by the pre-trained DNN, with or without fine-tuning. Depending on the application, one may be interested in the representations obtained more or less deeply in the pre-trained network. For instance, SoundNet [AVT16] is a DNN which has been trained to learn rich natural sound representations. The authors showed that representations extracted from the first layers are associated with low-level audio characteristics whereas representations obtained from the last layers tend to be associated with semantic categories.

DNNs can be pre-trained on labeled data from similar problems. For a particular problem, there may exist a large dataset in a similar domain. This would be for example the case of an application consisting in learning to recognize plant species from a small image database. One could pre-train a network on the large ImageNet image database and fine-tune it afterwards. There may also be plenty of small datasets containing data similar to the problem at hand. Training a network to learn to solve all these small problems simultaneously may help to learn a more robust representation of the data. For instance in neuroscience, a large and diverse database could be gathered from numerous studies.

Another technique for learning more robust representations is to define auxiliary tasks on the data. To classify images, an auxiliary task might be to rotate the images by a certain angle and try to predict their degree of rotation. This technique is often used as a complement to the first technique during the pre-training phase. With this technique, it is also possible to learn good representations from unsupervised (unannotated) data.

For example, in the context of natural language processing, a network pre-trained to predict the word following the beginning of a sentence has proven to be very efficient in predicting the sentiment of people from the text they wrote without being trained on this task. This phenomenon is called zero-shot learning. The representation is so well adapted that there is no need to fine-tune the network.

Methods based on the artificial generation of examples are conceptually distant from the previous techniques: they are often studied from a probabilistic perspective [Cha+21]. Generation methods require paying attention to the nature of the data. It seems indeed relatively easy to generate new examples of natural images by recombining objects between them, by zooming the images, by rotating them... It seems much more complicated to do the same thing with fMRI images. At least, it is easier to make sure that the natural images generated are relevant, while it seems difficult to know if, for instance, a given pattern of brain activity is biologically plausible. For this reason, we do not investigate this family of methods in our work.

A promising few-shot application: brain activity decoding

Among the promising areas for few-shot learning, we can think of neuroscience and in particular brain decoding. We first provide a brief introduction to neuroscience before explaining why few-shot learning is promising for brain decoding. Note that through brain decoding, we are also confronted with the issue of irregular data structure.

Short primer on neuroscience

The field of neuroscience can be seen through several facets. On one hand, researchers try to understand how the brain works, in order to discover the underlying mechanisms of cognition. On the other hand, other researchers draw inspiration from these mechanisms to build machines.

The brain works through a series of chemical reactions. Sensors located on our body receive stimuli. These stimuli arrive in certain areas of the brain from where they are distributed to other areas through complex chains of reactions. Our capacities to think, react and adapt come from there. Much progress has been made in understanding how such reactions can give rise to varied abilities. For example, in the context of visual perception, we know that some of the cells in our eyes are sensitive to certain frequencies of electromagnetic radiation. These cells send information to a brain region at the back of the head called the visual cortex. In the visual cortex, networks of neurons are organized to be more activated by contrasts. Other neurons are connected to the outputs of these first networks and allow the recognition of more complex shapes. Through a succession of neurons, the preprocessed visual signals are sent to other areas of the brain more or less involved in phenomena related to memories, emotions...

Advanced technologies are used to better understand the brain: imaging methods have been invented from fundamental discoveries in chemistry, physics, mathematics and electronics.

In parallel, a whole community is working on a related issue: taking inspiration from the brain to improve technologies. As the most powerful machines currently performing a whole range of cognitive tasks are human beings, our functioning has inspired many technologies. The perceptron, the ancestor of the DNNs, had been designed by a psychologist inspired by the abilities of the neurons to accumulate inputs from various other neurons and be activated once their sum exceeds a certain threshold. The

sequence of convolutional filters in the CNNs is similar to the visual mechanisms we described above. Sometimes, algorithms, inspired by the natural functioning of organisms, are adapted to human technologies. For example, the learning mechanisms in the brain are thought to be localized. One of the prevailing rules of learning is that neurons that fire together connect together. On the contrary, DNNs implement a learning mechanism called backpropagation that is not biologically plausible. Using backpropagation, the parameters of the entire network are adjusted according to the error on the final output.

A growing application in the field of neuroscience is brain decoding. Brain decoding consists in trying to recognize cognitive function from a more or less reliable measure of brain activity. An efficient decoder could tell us more about the underlying mechanisms of cognition but this is not necessary the primary goal of brain decoding. For many applications, the decoding accuracy matters in itself. In the long term, brain decoding may allow paralyzed people to communicate with their brains and to interact with objects (select words with their mind, say yes, no, move prostheses and robotic arms...). On this type of problem, it seems natural to try to use machine learning and deep learning techniques.

Introduction to brain decoding

Brain decoding [Hax+01] is similar to a classical classification problem. We have data, we want to classify them. The most efficient classifiers on images are currently CNNs that learn to extract interesting information from the data without a human needing to implement specific filters. However, they might not be directly applicable to brain data and there are two reasons behind this statement: the lack of labeled data and the complex data structure.

The datasets with the best spatial resolutions come from fMRI studies. As fMRI studies are expensive, they often contain a relatively small amount of data. fMRI images can be assimilated to videos of 3D images highlighting the activated areas in the brain of a person over time during a certain activity. Each image is divided into a set of cubes called voxels. However, contrary to images in computer vision which are regularly structured (localized classes, invariance to translation...), the information on fMRI images is spatially structured by the connectome (i.e. the set of neuronal connections between brain regions).

Historically, researchers first investigated whether certain regions could be associated with particular tasks (for example, face recognition) [KMC97]. It turned out that regions significantly activated in one context are also often activated in other contexts. Then, researchers tried to associate a task with a recurrent activation pattern over a group of regions [Hax12]. As each voxel gathers thousands of neurons, a weakly activated voxel can be informative if it is always activated within a certain context. A set of regions that is more or less robustly activated during a task can allow for a more reliable identification of the task. In this context, machine learning and deep learning methods are valuable tools for learning to recognize these patterns.

In another perspective, other researchers have focused on the intrinsic nature of brain activity: a network of neurons. Some proposed to study the brain through the prism of network theory. To that end, brains are divided into small regions. From this division, functional graphs are inferred from the data. The nodes of a functional graph correspond to brain regions. Nodes whose activity over time is highly correlated are

connected. For example, in this study [CD16], several functional graphs were identified for different tasks. It was found that the networks identified in motor tasks tend to be more segregated (i.e. they are rather independent of each other) whereas the networks associated with memory tasks are more integrated. In other works [Zha+21c], a single graph is used to model the overall structure of the brain. This can be a functional graph defined as before from data from people at rest. It can also be anatomical graphs obtained from other brain imaging techniques. In this framework, brain activity appears as a signal on a graph. The rapidly growing field of Graph Signal Processing (GSP) [Shu+13], which defines how to extract information from graph signals, can therefore be useful for brain decoding. In this paper [Zha+21c], the filters defined in the GSP framework are learned through DNNs to learn how to better classify brain activity.

How to Better Learn from Few Examples?

The general theme of this thesis is learning from few examples by transferring knowledge and exploiting data structure. Throughout this thesis, we emphasize on classification techniques for natural images and for fMRI images.

Knowledge transfer techniques are very promising. It has been proven that using more data allows for significant performance gains and more robust learning. Many potential deep learning applications which are currently restricted by the lack of labeled data, could learn to leverage existing resources by broadening the context of their study.

By considering two types of data, images and fMRI images, we cover two different paradigms. On one hand, the natural images come from standard datasets, collected in order to serve as a training set for neural networks. Overall, the data is low-noise; the classes are recognizable in the images. Each class is represented by the same number of examples. Techniques that do not work on these images are unlikely to work on more complex data. On the other hand, fMRI images are used from a brain decoding perspective. Brain decoding represents a practical application for few-shot learning, which is further complicated by the fact that the data is extremely noisy. With this example, we show how state-of-the-art techniques on images can be adapted to a specific problem. Raw fMRI images can be seen as 3D videos of a person's brain activity. The measured activity is noisy for many different reasons such as the spatial and temporal resolutions of fMRI, the unavoidable interfering signals during a particular task (e.g moving his fingers, being distressed by the enclosed space of the scanner...). Brain activity also varies between individuals and even for a single individual over time: several brain processes can achieve the same task and these processes are continually adapting.

Our interest in exploiting the data structure is two-fold. We can consider the inherent data structure of a data sample and also the structure generated by the relationships between the data samples. In scenarios with little data, and therefore little information, it seems all the more essential to take into account related sources of information. On one hand, the relative distribution of the data samples provides additional information about the difficulty of the problems to be addressed. On the other hand, taking into account the inherent data structure of the data samples can really help to ease the learning.

For instance, as stated before, natural images have a grid structure. Objects can be translated through the grid while still being the same object. CNNs have been conceived to take advantage of this structure: they learn small convolutional filters (a few pixels) which are applied locally on an image to detect particular shapes. However, the

structure of the brain activity shown on the 3D images obtained with fMRI is more complex. Although the values of the voxels are related to the activation of the underlying brain regions, the dynamics of brain activity is not well represented by the regular grid of the image. As brain regions are connected by bundles of neurons, regions involved in a same task can be physically very distant. Interestingly, these underlying neural connections between brain regions can be inferred from other imaging techniques. They are a valuable source of information on the intrinsic data structure of brain activity.

Training better algorithms on problems with little data is crucial for many fields. In medicine, we mentioned earlier the impact related to medical image analysis and to diagnoses. Gathering databases containing symptoms of all known diseases would already be a tremendous task. However, even if we had access to such data, many diseases are yet unknown and new epidemics appear regularly. Learning from a few examples to detect them could be useful and identifying groups of people with similar symptoms (zero-shot learning) could help advance the research. One can imagine a world where large networks will direct diagnoses to many small and very specialized AIs. The small AIs would perhaps themselves be trained from other large networks gathering a part of the human knowledge. Similar trends may also appear to detect rare anomalies, forecast new events...

Outline of the Manuscript

In this thesis, we found it interesting to study current few-shot learning methods, to highlight their weaknesses and to try to adapt them to the brain decoding application.

Preliminaries

In the first chapter entitled Preliminaries, we explain the fundamental notions related to 4 domains: machine learning, deep learning, probability theory and graph signal processing. In the context of machine learning, we introduce the notions of supervised and unsupervised learning. We introduce the notion of generalization. As we are mainly interested in classification problems, we present and compare some of the most famous classifiers in machine learning. In the context of deep learning, we detail what a neural network is. We show how its parameters are learned from a training set and we present several commonly used network architectures. The part on probability theory is mainly related to an attempt we will make in the last chapter to model brain activity. We define what are random variables and vectors as well as the notions of normal distribution and multivariate normal distribution. Finally, in the last section, we introduce the formalism used to process signals on graphs. We describe what are graphs and the signals defined on these graphs. We detail the notion of frequency on a graph defined by analogy with the Fourier transform. We also present some filters on graphs as well as some methods that propose to learn directly these filters using neural networks.

One of our contributions characterizes graph neural networks with a unified formalism. It has been published at a French conference ([Bon+19c] for the English version):

Myriam Bontonou et al. "Un modèle unifié pour la classification de signaux sur graphe avec de l'apprentissage profond". In: *GRETSI*. 2020

Learning with few examples

The second chapter gathers the work we have done on the problem of few-shot learning in the context of image classification. In the first section, we present the methods that are currently used. They are divided into three parts: methods inspired by transfer learning, methods inspired by meta-learning and methods aiming at artificially increasing the number of examples. In this manuscript, we only consider transfer learning and meta-learning. Transfer learning aims at training large networks on rich and diverse databases and adapting the knowledge acquired to the new problem with few examples. Meta-learning aims at learning to learn. The goal is to learn a network whose initial parameters allow it to quickly learn to solve several few-shot problems. In the second section, we highlight one of the big problems of all these methods for practical applications: the evaluation of their performance. In general, in machine learning, datasets are divided into three sets: a training set, a validation set and a test set. The parameters of the algorithms are learned on the training set. The hyperparameters (i.e. parameters that cannot be learned directly such as the number of layers in the network) are chosen on the validation set. Finally, the generalization performance (i.e. the ability of the algorithm to classify examples that it did not use during training) is evaluated on the test set.

In one of our contributions, we seek alternative methods to evaluate the performance of trained neural networks on small datasets:

Myriam Bontonou, Louis Béthune, and Vincent Gripon. "Predicting the Generalization Ability of a Few-Shot Classifier". In: *Information* 12.1 (2021), p. 29

It is also interesting to note that the problem of assessing the performance of DNNs without using a test set is also of interest to researchers who have access to a lot of data. In particular, this was the theme of one of the competitions at NeurIPS 2020. Paradoxically, discovering new ways to assess the generalization ability of networks could improve their capabilities: networks could be trained to optimize the generalization measures. We participated in the NeurIPS competition by proposing a solution based on the relative distances between the representations of the examples obtained in the last layers of the DNNs:

Carlos Lassance et al. "Ranking Deep Learning Generalization using Label Variation in Latent Geometry Graphs". In: *arXiv preprint arXiv:2011.12737* (2020)

Few-shot learning for decoding brain activity

Finally, the last chapter is dedicated to a concrete application of neuroscience: how to decode the thoughts of people from measurements of their brain activity? We first explain the phenomena that induce brain activity and enable to record it. As we use data from Magnetic Resonance Imaging (MRI), we also describe the different images that are obtained with MRI-based techniques such as diffusion Magnetic Resonance Imaging (MRI) and functional Magnetic Resonance Imaging (fMRI). In brief, diffusion MRI provides images of the fiber networks through which information is transmitted.

fMRI records data related to the brain activity of a person. This data looks like a 3D video showing how much activity there is in certain regions of the brain over time. In experimental studies, the spatial resolution is often limited to a few millimeters and the temporal resolution is of the order of a few seconds. Brain activity related to a particular task is often measured in the following way: a person performs a cognitive, sensory

or motor task within a scanner. For example, he or she may be asked to remember a letter of the alphabet. Then they are shown new letters one by one and asked to identify whether the letter they have retained has appeared. Using a general linear model, the components that are not related to the task of interest can be removed (parasite movement, rest period between repetitions of a certain task...).

The raw data are transformed into 3D statistical maps highlighting the regions activated during the task. There is no more temporal component. However, such maps do not only account for the studied cognitive mechanism. In the previous example, the activity highlighted in the map is probably as much related to the reading of letters on the scanner screen as to the memorization effort. To dissociate both mechanisms, a second experiment can be performed, in which the person passively watches letters scroll by. The statistical maps resulting from the two experiments can be subtracted to highlight the regions more specifically activated by the memorization process. The result of the subtraction is called a contrast map. We consider learning from these preprocessed data instead of learning from the raw temporal signals. This choice is motivated by a computational constraint. The contrast maps are smaller (a few gigas compared to several teras for the raw data). They are therefore easier to store and manipulate. It is also easier to train networks on an image than on a time series of images. As our objective is to analyze the relevance of transfer methods, we preferred to start by analyzing "simplified" data.

We compare several few-shot learning methods, which are state-of-the-art on natural images, on decoding problems. To generate the decoding problems, we rely on a set of contrast maps from a French study named the Individual Brain Charting (IBC) project [Pin+18; Pin+20]. This work has been synthesized in this article:

Myriam Bontonou et al. "Few-shot Learning for Decoding Brain Signals". In: *arXiv preprint arXiv:2010.12500* (2020)

Finally, we try to take advantage of the data structure by exploiting an anatomical graph obtained from diffusion MRIs averaged over several individuals [PVDV19]. The nodes of the graph represent 360 predefined brain regions. The weight of the connections between the nodes is proportional to the number of fibers averaged between the different regions. On one hand, we try a machine learning oriented approach in which we insert the anatomical graph into a neural network. This approach is detailed in the previous article. On the other hand, we try to classify the data by modeling their distribution. In our model, we imagine that each of the classes is well represented by a reference contrast map. The contrast maps that have been measured are assumed to be noisy versions of this representative. The noise is partly defined from our prior knowledge on the relationships between brain regions. The second approach is currently being published:

Myriam Bontonou, Nicolas Farrugia, and Vincent Gripon. "Graph-LDA: Graph Structure Priors to Improve the Accuracy in Few-Shot Classification". In: *arXiv preprint arXiv:2108.10427* (2021)

Unrelated publications

It is also worth mentioning that we worked on other questions whose contributions are unrelated to the main problematic of this manuscript. These questions explore other limitations of deep learning.

Notably, we proposed to introduce graph to improve DNNs training and to compress DNNs. In this article, we define a new loss based on the smoothness of the class signals on a graph whose nodes represent training examples and edges encode distances between the representations of the examples within a DNN:

Myriam Bontonou et al. "Introducing graph smoothness loss for training deep learning architectures". In: *2019 IEEE Data Science Workshop (DSW)*. IEEE. 2019, pp. 160–164

In this article, we improve the concept of knowledge distillation between a teacher network and a student network. We foster the student network to project the training examples in a space where their representations are as distant as in the space learned by the teacher network:

Carlos Lassance et al. "Deep geometric knowledge distillation with graphs". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 8484–8488

We also wrote a book chapter on a set of problems where graphs contribute to improve deep learning techniques.

Finally, in another line of work, we compared the effectiveness of representations learned by DNNs and the ones extracted by structure-based filters from GSP to predict the evolution of graph-supported time series:

Myriam Bontonou et al. "Comparing linear structure-based and data-driven latent spatial representations for sequence prediction". In: *Wavelets and Sparsity XVIII*. vol. 11138. International Society for Optics and Photonics. 2019, 111380Z

Chapter 1

Preliminaries

Contents

Callout: Why Reading this Chapter?

THIS chapter is conceived as a toolbox containing information required to understand the next chapters. Before going into detail, here is a short paragraph explaining why these tools are important.

Throughout this thesis, we want to learn to solve tasks from examples by addressing two questions. Most of the time, we face classification problems. We have access to training examples (e.g. observations of biological phenomena, images...) associated with labels. Our goal is to be able to recognize the same labels on new observations. When many training examples are available, the state-of-the-art method is often a deep learning method.

→ *Machine learning and deep learning are presented in sections 1.1 and 1.2.*

What if we only have a small number of training examples? A Deep Neural Network (DNN) with many parameters trained from scratch is very likely to overfit on the few training examples without being able to generalize well to new examples. Within deep learning, a community focused on few-shot learning has emerged. In combination with knowledge extracted from DNN, they often advise to use a machine learning classifier with few parameters and, if possible, a model of the distribution of the data.

→ *Elements of probability theory (for the models) are presented in section 1.3.*

What if the data yields an irregular structure? For instance, a data sample can represent brain activity. Each feature measures the activity within one cerebral region. We know that some regions are more connected than others. How can we exploit this information? Do we need to exploit it to reach a good performance? In theory, DNNs could learn without using the data structure. However, what really boosted the performance of deep learning on images (regularly structured data, with the notion of translation), is the use of convolutions. With convolutions, filters containing a small number of parameters are applied locally on the images and translated. As such, the DNN learns to recognize patterns from the images using a small number of parameters. Learning the same patterns without using the structure would require to use many more parameters and would be harder to train. This is why more and more works try to merge deep learning and filtering operations from the graph signal processing field.

→ *Graph signal processing is presented in section 1.4.*

Notations Throughout this manuscript, the coefficients of a matrix $\mathbf{X}$ and a (column) vector $\mathbf{x}$ are denoted by $\mathbf{X}_{i,j}$ and $\mathbf{x}_i$. A set is represented by a calligraphic capital letter $\mathcal{S}$. The i^{th} data sample of a dataset is always represented by a bold lowercase letter $\mathbf{x}^i$, even if it is a tensor with several dimensions. The label associated with the input i is y^i when it directly represents the name of a class or $\mathbf{y}^i$ when the class is encoded in a one-hot vector. Similarly, the label estimated by a model is $\hat{y}^i$ or $\hat{\mathbf{y}}^i$. A random variable is represented by X and a random vector by $\boldsymbol{X}$.

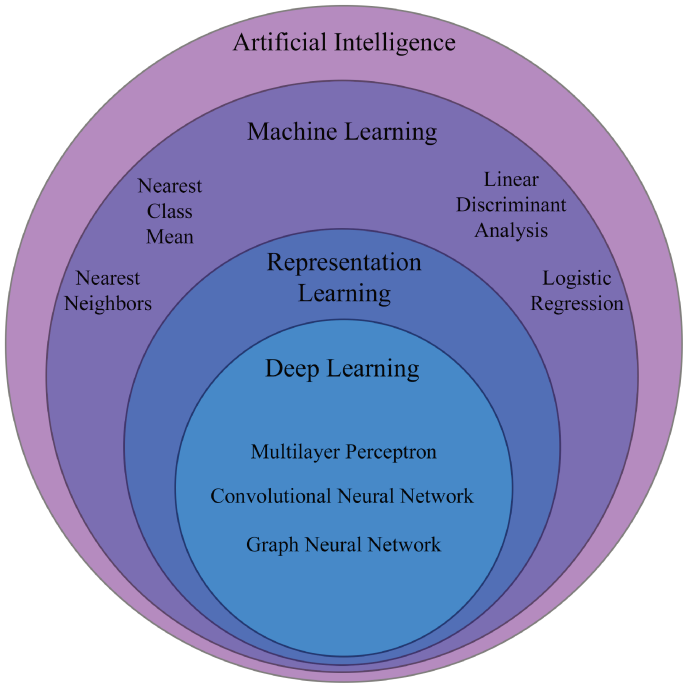


Figure 8 – Illustration highlighting a few algorithms used in this manuscript.

1.1 Machine Learning

The goal in machine learning is to define algorithms solving problems by themselves. Humans only have to specify how the machine will learn to solve the problems. Machine learning belongs to the broader field of AI and contains a sub-domain called deep learning. Figure 8 illustrates this nesting while mentioning the names of the main algorithms used throughout the manuscript. Machine learning algorithms and deep learning algorithms have particular characteristics that distinguish them from other algorithms. These characteristics are presented concisely in Fig. 9. Throughout this section, we present some of the major concepts of machine learning. The next whole section is dedicated to deep learning as it is a central subject of this manuscript.

1.1.1 Learning problems

Many learning problems are defined in different contexts: it depends on the input data samples and on the desired output (e.g. a class domain, a compact representation...). The aim is to find a function enabling to infer the desired output from an input. Here, we distinguish supervised learning, semi-supervised learning and unsupervised learning. In this manuscript, we are mostly interested in supervised learning and in particular in classification problems. As for semi-supervised and unsupervised learning, they appear a few times.

Definition 1

In ***supervised learning***, examples of input/output couples $(\mathbf{x}, f(\mathbf{x}))$ are provided. The aim is to infer the function f. Once it has been inferred, it is used to predict the outputs associated with previously unseen inputs.

Classification is a supervised learning problem. The labels $f(\mathbf{x})$ represent a finite number of classes. For instance, the data samples and their labels can be images repre-

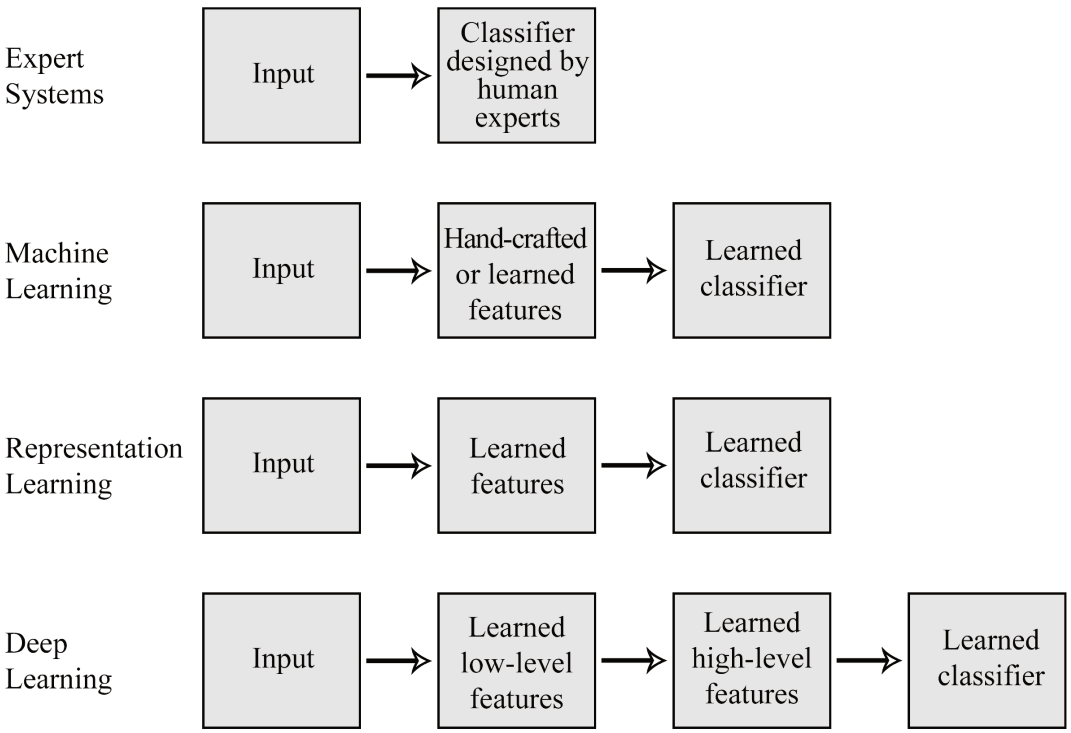


Figure 9 – Scheme illustrating the differences between several sub-domains of AI on a classification task. Deep learning is included within representation learning, itself included within machine learning.

senting particular objects.

Definition 2

In ***semi-supervised learning***, both examples of the form $(\mathbf{x}, f(\mathbf{x}))$ and unlabeled inputs $\mathbf{x}$ are available during training. The aim is to predict the outputs of the unlabeled inputs.

Semi-supervised classification is a semi-supervised learning problem. As in (supervised) classification, the labels represent a finite number of classes. The main difference is the fact that the inputs $\mathbf{x}$ on which the predictions will be made are also accessible without their labels during training.

Definition 3

In ***unsupervised learning***, we only have access to input data samples. The aim is to find disentangling representations of the data or to partition the inputs into coherent subsets.

Clustering is the name of the unsupervised learning problem that consists in finding coherent subsets within the data.

1.1.2 Performance metrics

A common performance metric in classification is the accuracy. In the ideal case, 100% accuracy is achieved. All examples are correctly classified. In the worst case, the accuracy is at the chance-level. It correctly classifies as many examples as a random classifier. For instance, if a dataset contains 10 balanced classes (i.e. with as many examples per class), the chance-level accuracy is 10%.

Definition 4

The ***accuracy*** consists in dividing the number of examples correctly classified by the total number of examples.

If the classes are not balanced, the accuracy can be misleading. For instance, if a dataset contains 950 examples of one class and 50 examples of another class, the chance-level accuracy is at 95%. In that case, other measures such as the sensitivity and the specificity can be reported for all classes.

Definition 5

Given a class, the ***sensitivity*** is the proportion of examples classified as belonging to this class among the examples of this class.

Definition 6

Given a class, the ***specificity*** is the proportion of examples classified as belonging to the other classes among the examples of the other classes.

1.1.3 Datasets

In machine learning, the function to infer is learned by an algorithm from data samples. Thus, each learning problem has its type of dataset. For supervised learning, the dataset contains labeled data samples. It is split into 3 disjointed parts: a training set, a validation set and a test set, as shown in Fig. 10. For instance, we may want to solve a

Figure 10 – Structure of a dataset in supervised learning.

classification problem containing a certain number of classes. In that case, each part of the dataset contains labeled data samples from all classes.

Training, validation and test sets

Definition 7

The ***training set*** is used to learn the parameters of an algorithm. The algorithm is trained to map the training examples with predefined labels.

A high training accuracy means that the algorithm has successfully learned to associate the correct labels with the training examples. When the training accuracy is lower than expected (compared for instance to the accuracy of human beings performing the same task), the algorithm is said to be ***biased***.

Definition 8

The ***validation set*** is used to evaluate if the trained algorithm is able to generalize well to new examples. It is also used to tune the ***hyperparameters*** of the algorithms (the ones that cannot be learned). In practice, algorithms with different sets of hyperparameters are trained on the training dataset. The algorithm with the best performance on the validation dataset is selected.

The ***variance*** of an algorithm is measured by the difference between the training and the validation accuracy. Situations where the variance is large are called ***overfitting***. Such situations can occur for several reasons. The training set may not be representative of the diversity of the data. Therefore, the algorithm may be too focused on irrelevant details of the training examples. For instance, let us consider a classification problem in which an algorithm is trained to recognize two classes of images, cats and dogs. The training set only contains black dogs and white cats whereas the validation set contains various dogs and cats. By focusing on the color of the pets, the training accuracy could reach 100% whereas the validation one would be close to 50% (chance level).

Definition 9

The ***test set*** is often used in academic benchmarks to compare the performance of various solutions. In this case, the test set enables to evaluate the potential overfitting that can occur on the validation dataset. It must not be used during training. Sometimes, for practical applications, no test set is set aside in order to use all available data for training. In that case, the probability of overfitting on the validation set is neglected.

Examples of supervised datasets

Here are a few examples of datasets used in this manuscript. MNIST and CIFAR-10 are used for standard classification problems. Mini-ImageNet and Tiered-ImageNet are used for few-shot classification problems. For such problems, the datasets have a different structure. Their classes are split into three sets: a base set, a validation set and a novel set. The logic behind this division is explained in the next chapter (section 2.1.3).

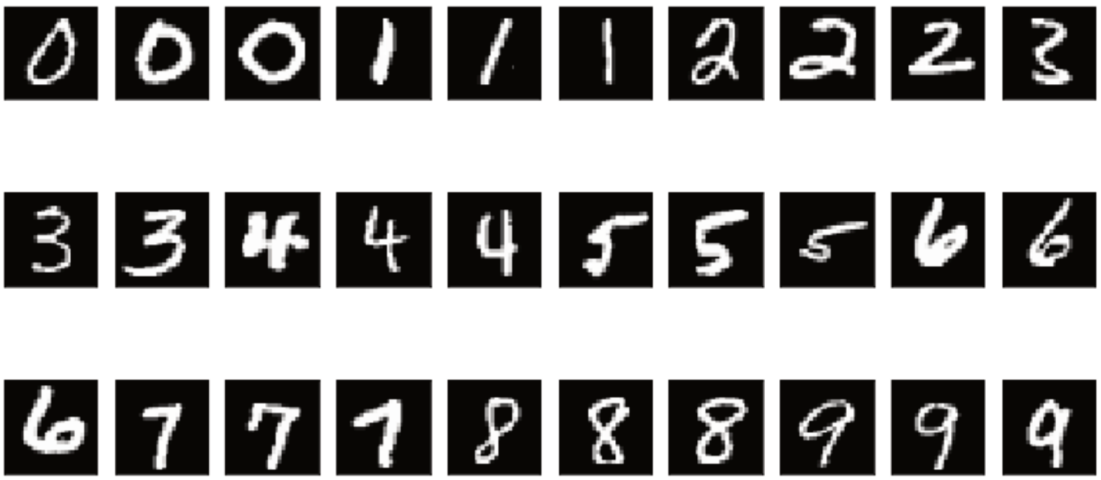


Figure 11 – Examples of images in MNIST.

Figure 12 – Examples of images in CIFAR-10.

Figure 13 – Examples of images in mini-ImageNet.

MNIST MNIST [LeC+98] contains 70000 black-and-white images of size 28×28. There are 60000 training images and 10000 test images. The images are divided into 10 classes, with 7000 images per class. Each class represents a handwritten digit. MNIST is one of the first large datasets designed for AI. It has been created from digits written by high-school students and United States Census Bureau employees. It is still used today in many tutorials. Fig. 11 shows a few examples.

CIFAR-10 CIFAR-10 [KH+09] contains 60000 color images of size 32×32. There are 50000 training images and 10000 test images. The images are divided into 10 classes, with 6000 images per class. Fig. 12 shows a few examples.

Mini-ImageNet Mini-ImageNet [Vin+16] contains 100 classes from ImageNet [Rus+15]. The base set contains 64 base classes, the validation set contains 16 validation classes and the novel set contains 20 novel classes. Each class contains 600 images of size 84×84. Fig. 13 shows a few examples.

Tiered-ImageNet Tiered-ImageNet [Ren+18] contains 608 classes from ImageNet [Rus+15]. Its classes are split into three sets: a base set with 351 base classes, a validation set with 97 validation classes and a novel set with 160 novel classes. Each class contains about 1000 images of size 84×84. Tiered-ImageNet is supposed to be less biased than mini-ImageNet because its 3 sets contain semantically different classes (semantics evaluated with WordNet [Mil95]).

1.1.4 Classifiers

In this section, we present some of the most common classifiers used in machine learning. The number of classes in a classification problem is denoted by C. As stated at the end of the callout, the i^{th} data sample of a dataset is denoted by $\mathbf{x}^i$, its label by y^i and the label estimated by a classifier by $\hat{y}^i$. Sometimes, the labels are represented by one-hot vectors $\mathbf{y} \in \mathbb{R}^C$ where $\mathbf{y}_c = 1$ if c is the class associated with $\mathbf{y}$ and $\mathbf{y} = 0$ otherwise. In this case, the labels and their estimations are represented by bold letters.

Nearest neighbors

A k Nearest Neighbors (k-NN) classifier [Bis06] (section 2.5) memorizes all training examples. When a new test example comes, the classifier looks at the k training examples closest to the new one. The metric used to evaluate the proximity of the examples is often the Euclidean distance. The class attributed to the new example is the most present class among these k nearest neighbors.

Nearest class mean

The Nearest Class Mean (NCM) classifier, also called nearest centroid, memorizes the centroids of all classes, computed by averaging the training examples belonging to each of these classes. To classify a new example, the classifier computes the distance between these centroids and the new example. The class attributed to the example is the one with the closest centroid. Once again, the most often used distance is the Euclidean distance.

Linear discriminant analysis

Linear Discriminant Analysis (LDA) is a classifier that is well suited when the classes follow multivariate Gaussian distributions with the same covariance matrix [DH+06], [FHT+01]. When the covariance matrix is known, the examples are linearly transformed so that their new covariance matrix becomes the identity matrix. Such transformation is called ***sphering*** or ***whitening***. After that, a NCM is applied on the examples. Note that in the case where the priors on the probability of the classes are not the same, the computed distances are corrected by these priors.

Several techniques exist to estimate the covariance matrix when it is unknown. If a lot of training examples are available, the empirical estimator of the covariance is a good estimator. If there is a small number of examples, shrinkage techniques, such as the Ledoit-Wolf shrinkage estimator [LW04], are used. Moreover, if the distribution of the classes is assumed to be Gaussian, a better shrinkage estimator is the Oracle Approximation Shrinkage (OAS) estimator [Che+10]. As it is supposed to be the case, it is advised to use LDA with the OAS estimator.

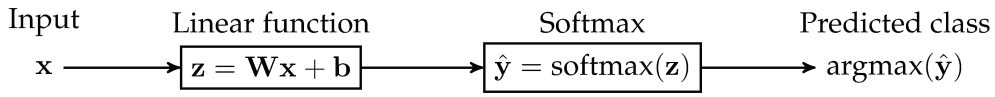


Figure 14 – Steps in a logistic regression. The parameters are $\mathbf{W} \in \mathbb{R}^{C \times d}$ and $\mathbf{b} \in \mathbb{R}^{C}$.

Logistic regression

Another very classic classifier is the Logistic Regression (LR) [Bis06]. In Fig. 14, we present the steps required to perform a LR. A linear function is applied on the input vector $\mathbf{x}$ before a softmax function. The softmax function applied to a vector scales the coefficients of this vector so that they lie in the range $[0, 1]$ and sum to 1. The predicted class is the index of the maximum coefficient of the output of the softmax. Let the matrix $\mathbf{W} \in \mathbb{R}^{C \times d}$ and the vector $\mathbf{b} \in \mathbb{R}^{C}$ containing the trainable parameters. Formally, the class predicted by the LR is given by:

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b}) \quad \text{with} \quad \text{softmax}(\mathbf{x}_i) = \frac{\exp(\mathbf{x}_i)}{\sum_j \exp(\mathbf{x}_j)} \,. \tag{1}$$

The parameters are learned with a gradient descent or one of its variants to minimize a loss function. Most of the time, the loss function is the cross-entropy loss. Given $\ln$ the natural logarithm, the cross-entropy loss is defined as:

$$\mathcal{L}_{\text{CE}}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_j \mathbf{y}_j \ln(\hat{\mathbf{y}}_j) \,. \tag{2}$$

Gradient descent consists in computing the partial derivatives of the loss function with respect to the parameters and then to update the parameters. Given a learning rate $\alpha > 0$, the parameters are updated according to the following equations:

$$\mathbf{W}_{i,j} \coloneqq \mathbf{W}_{i,j} - \alpha \frac{\partial \mathcal{L}_{\text{CE}}}{\partial \mathbf{W}_{i,j}} \tag{3}$$

$$\mathbf{b}_i \coloneqq \mathbf{b}_i - \alpha \frac{\partial \mathcal{L}_{\text{CE}}}{\partial \mathbf{b}_i} \,. \tag{4}$$

The partial derivatives of $\mathcal{L}_{\text{CE}}$ with respect to the parameters are easy to compute because $\mathcal{L}_{\text{CE}}$ is a composition of simple differentiable functions so that its derivative can be computed with the chain rule:

$$\frac{\partial \mathcal{L}_{\text{CE}}}{\partial \mathbf{W}_{i,j}} = \frac{\partial \mathcal{L}_{\text{CE}}}{\partial \mathbf{z}_i} \frac{\partial \mathbf{z}_i}{\partial \mathbf{W}_{i,j}} \,, \frac{\partial \mathbf{z}_i}{\partial \mathbf{W}_{i,j}} = \mathbf{x}_j \text{ and } \frac{\partial \mathcal{L}_{\text{CE}}}{\partial \mathbf{z}_i} = \hat{\mathbf{y}}_i - \mathbf{y}_i \,. \tag{5}$$

The parameters are updated until the loss converges towards a local minimum.

Comparison of the classifiers

In Fig. 15, the decision boundaries of the three previously mentioned classifiers are plotted on two simple classification problems. The code is based on scikit-learn [Ped+11]. This figure highlights the fact that the decision boundaries of LR, LDA and NCM are linear.

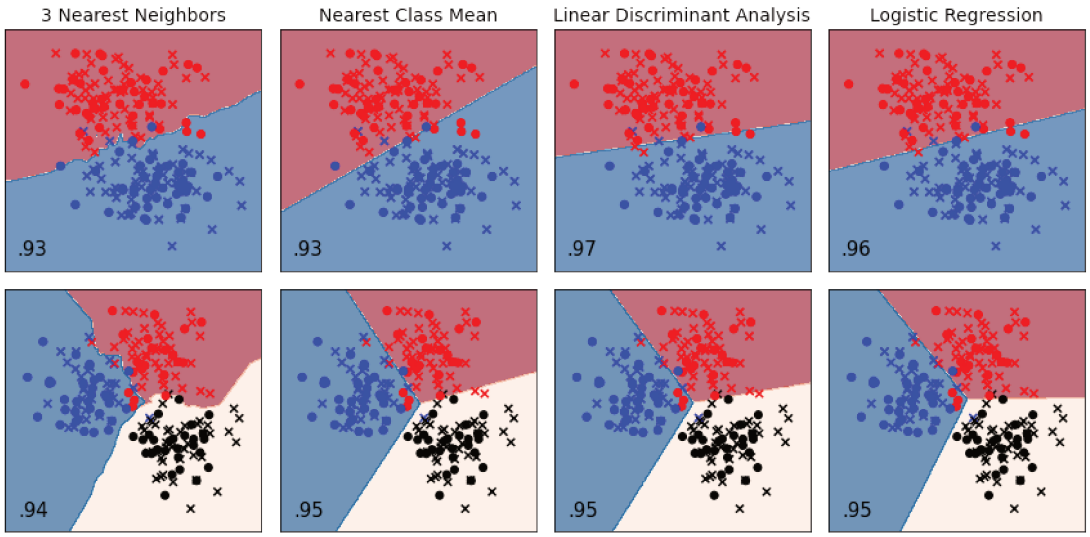


Figure 15 – Illustration of the decision boundaries obtained with several classifiers. Top and bottom rows correspond to cases with respectively 2 and 3 classes. The crosses are the training examples. The circles are the test examples on which the accuracy scores (bottom left corner) are computed. Each color corresponds to a distinct class.

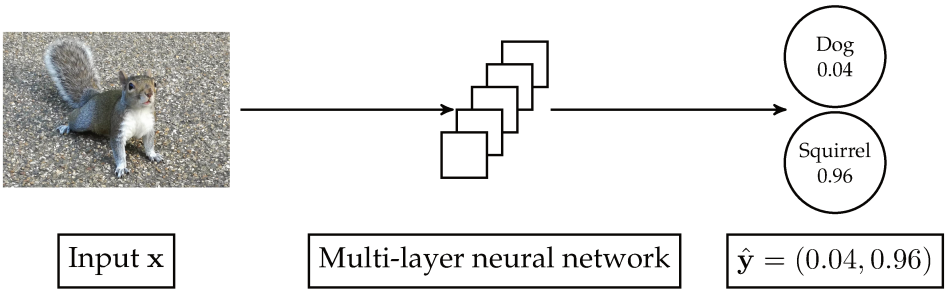


Figure 16 – DNN used to classify images of squirrels and dogs.

1.2 Deep Learning

Deep learning is a subsection of machine learning [GBC16]. In deep learning, algorithms called Deep Neural Networks (DNNs) learn hierarchical representations of the data in order to perform a given task. For instance, to solve a classification problem, these representations are often learned to enable a LR to better classify the data. The representations are said hierarchical because they are obtained through a succession of intermediate representations. In subsection 1.2.1, DNNs are formally described. In subsection 1.2.2, several types of DNNs layers and architectures are described.

1.2.1 Deep neural networks

Definition 10

A ***Deep Neural Network (DNN)*** is an assembly of linear and non-linear functions organized into multiple layers. It learns to map an input (i.e. a data sample) with an output. Specifically, a DNN learns successive ***intermediate representations*** of the initial data sample, which helps it associate the desired output with the input.

Formally, the simplest DNN architecture f can be defined as:

$$f(\mathbf{x}) = \sigma(\mathbf{b}^{[n]} + \mathbf{W}^{[n]}\sigma(\dots\sigma(\mathbf{b}^{[2]} + \mathbf{W}^{[2]}\sigma(\mathbf{b}^{[1]} + \mathbf{W}^{[1]}\mathbf{x})))) \,, \tag{6}$$

where $\mathbf{x} \in \mathbb{R}^d$ is a data sample containing d features and σ is a non linear function (such as the ReLU function). In this architecture, each layer contains a linear function followed by a non-linear one. The function $\mathbf{a}^{[l]} = \sigma(\mathbf{W}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]})$, inside the DNN is the function computed within a layer l, where $\mathbf{W}^{[l]} \in \mathbb{R}^{d^{[l]} \times d^{[l-1]}}$ and $\mathbf{b}^{[l]} \in \mathbb{R}^{d^{[l]}}$ contain the parameters learned in the l^{th} layer. For instance, in the first layer, $\mathbf{a}^{[0]} = \mathbf{x}$ and $d^{[0]} = d$.

For instance, Fig. 16 represents a DNN designed to perform a binary classification task. Its input is an image. It outputs a vector with two coefficients between 0 and 1, which sum to 1. The first class is associated with the label $\mathbf{y} = (1, 0)$ and the second one with the label $\mathbf{y} = (0, 1)$. The coefficient $\hat{\mathbf{y}}_0$ can be interpreted as the probability that the input belongs to the first class. Similarly, $\hat{\mathbf{y}}_1$ would be the probability that the input belongs to the second class.

Training

As for LR introduced in Chapter 1.1.4, the parameters of a DNN are learned through a mechanism called gradient descent. The algorithm behind gradient descent is Algorithm 1. In classification, the purpose of gradient descent is to update the parameters of

Algorithm 1: Mini-batch gradient descent

Require: Learning rate α.
Require: Batch size batch_size.
Require: Number of epochs epochs.
Require: Loss function $\mathcal{L}$.
Require: DNN parameters $\mathbf{W}_{i,j}^{[l]}$,$\mathbf{b}_i^{[l]}$.
Randomly initialize $\mathbf{W}_{i,j}^{[l]}$ and $\mathbf{b}_i^{[l]}$.
for epoch $= 1 \rightarrow$ epochs **do**
Generate mini-batches of size batch_size from the shuffled training examples.
for each mini-batch **do**
Forward pass
For each input $\mathbf{x}$ of the mini-batch, compute the loss function $\mathcal{L}(\mathbf{x})$.
Backward pass
For each input $\mathbf{x}$, compute the gradients with the chain rule $\frac{\partial \mathcal{L}(\mathbf{x})}{\partial \mathbf{W}_{i,j}^{[l]}}$ and $\frac{\partial \mathcal{L}(\mathbf{x})}{\partial \mathbf{b}_i^{[l]}}$.
Average the gradients: $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{i,j}^{[l]}} = \frac{1}{\text{batch\_size}} \sum_{\mathbf{x}} \frac{\partial \mathcal{L}(\mathbf{x})}{\partial \mathbf{W}_{i,j}^{[l]}}$, $\frac{\partial \mathcal{L}}{\partial \mathbf{b}_i^{[l]}} = \frac{1}{\text{batch\_size}} \sum_{\mathbf{x}} \frac{\partial \mathcal{L}(\mathbf{x})}{\partial \mathbf{b}_i^{[l]}}$.
Update the parameters: $\mathbf{W}_{i,j}^{[l]} := \mathbf{W}_{i,j}^{[l]} - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{i,j}^{[l]}}$, $\mathbf{b}_i^{[l]} := \mathbf{b}_i^{[l]} - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{b}_i^{[l]}}$.
end for
end for

the DNN in order to minimize a loss function, so that $\hat{\mathbf{y}}$ converges to the label $\mathbf{y}$. It uses several specific hyperparameters which are defined below. Recall that a ***hyperparameter*** is a parameter of the DNN that cannot be learned during training. For instance, some hyperparameters influence the DNN architecture (e.g. number of layers, number of parameters within each layer), other influence the training process (e.g. learning rate, batch size, number of epochs)...

Definition 11

The most common loss function is the ***cross-entropy loss***. Given $\ln$ the natural logarithm, it is defined as:

$$\mathcal{L}_{\text{CE}}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_j \mathbf{y}_j \ln(\hat{\mathbf{y}}_j)\,. \tag{7}$$

Definition 12

The ***learning rate*** controls the speed of the gradient descent. When it is too small, learning is really slow. When it is too large, the loss may oscillate and not be able to converge to a minimum. Learning rate scheduling is a very common technique that aims at decreasing the learning rate during the training process.

Definition 13

The ***batch size*** is the number of examples on which the gradients are averaged. Averaging the gradients on several examples enables to reduce the computation time.

Definition 14

One ***epoch*** is achieved when the DNN has been trained on all training examples. For instance, training a DNN for 10 epochs means that its parameters will be updated 10 times on all training examples.

Training process

To solve a supervised learning problem, a DNN is trained on a dataset split into 3 parts: the training set, the validation set and the test set. The purpose of these sets is described in the previous section on machine learning (section 1.1.3). To summarize, the parameters of the DNN are learned in order to map the training examples with their labels. The validation set is used to evaluate the generalization ability of the DNN on new examples. It is also used to select the hyperparameters (e.g. architecture of the DNN, learning rate...) leading to the best performance. The test set is supposed to represent the real-world data distribution. As such, it must not be used during training. In research, performance are reported on the test set. For practical applications, this set is not necessarily defined. Performance may only be evaluated on the validation set in order to keep as much data as possible for training.

The bias of a DNN, which is measured by the difference between the training accuracy and the expected training accuracy, can be reduced by training the algorithm for a longer time or increasing its number of parameters. Indeed, neural networks can theoretically compute any function. Even a MLP with a single hidden layer can compute any function as long as its hidden layer contains enough features [1]. On the other hand, the variance of a DNN, which is measured by the difference between the training accuracy and the validation accuracy, can be reduced by training on more diverse data or by introducing regularization techniques (described in the next subsection).

Optimization and regularization techniques

Many techniques have been designed to train the DNNs faster and to help them to generalize better by regularizing their learning process.

Input normalization Input normalization consists in centering and scaling the features of the inputs, so that each feature has an equal importance. For instance, the mean of each feature can be computed on the training set and subtracted from the same features on all data samples.

Batch normalization Batch norm consists in centering and scaling the features of the representations obtained within a DNN over a mini-batch of inputs. The values of the new mean and the new variance are learned like the other parameters during training.

Variants of the gradient descent Many variants of gradient descent are used to train current DNNs. Among them, the most common are the following. Gradient descent with momentum [Qia99] consists in smoothing the partial derivatives of the parameters can be smoothed over time. Adam [KB15] is similar to gradient descent with momentum. It also adapts the learning rate to each parameter so that it is inversely proportional to the magnitude of its partial derivative.

Weight decay Weight decay prevents the parameters to become too large by adding a regularization term to the loss function. The regularization term is the sum of the squared coefficients of all parameters.

1. The universality of DNNs is illustrated in the web page `http://neuralnetworksanddeeplearning.com/chap4.html` coming from a book on neural networks and deep learning [Nie15].

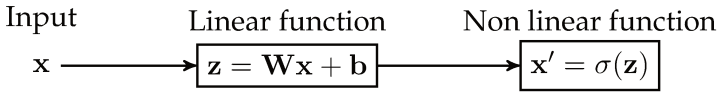


Figure 17 – Steps in a fully connected layer. The input is $\mathbf{x}$ and the output is $\mathbf{x}'$. The parameters are $\mathbf{W} \in \mathbb{R}^{d' \times d}$ and $\mathbf{b} \in \mathbb{R}^{d'}$.

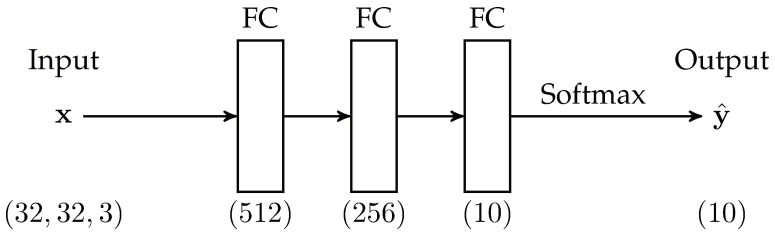


Figure 18 – MLP obtaining 55.26% accuracy on CIFAR-10. FC stands for fully connected layer. The last FC only contains a linear function. The numbers inside brackets are the number of features of the input and of the intermediate representations obtained after the layers. In total, this MLP contains about 2 million parameters.

Dropout Using dropout [Sri+14], several parameters are randomly set to 0 at some points during training. Thus, the DNN learns to rely on a larger set of features to make a decision.

1.2.2 Layers and architectures

Over the years, many DNN layers and architectures have been designed to meet specific needs. Here is a description of the most common layers and architectures.

Fully connected layer

The ***fully connected layer*** learns features without exploiting knowledge on the structure of the input (sequentiality, invariance by translation, localization...). Thus, this layer can be applied on various inputs such as temporal signals, text sequences or images.

Given an input vector $\mathbf{x} \in \mathbb{R}^d$, a non linear function σ, a parameter matrix $\mathbf{W} \in \mathbb{R}^{d' \times d}$ and a parameter vector $\mathbf{b}^{d'}$. The output of the layer $\mathbf{x}' \in \mathbb{R}^{d'}$ is computed by $\mathbf{x}' = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$. See Fig. 17.

Here are some limitations of the fully connected layer.

- ✗ If the input is structured, the information is not exploited.
- ✗ The number of parameters is linear in the size of the input: $\Theta(d'd)$.

A DNN made of a succession of fully connected layers is called a ***Multilayer Perceptron (MLP)***. In Fig. 18, we show an example of MLP trained on CIFAR-10.

2D convolutional layer

The 2D ***convolutional layer*** learns to extract relevant patterns from two-dimensional Euclidean space representing, for instance, images. Greyscale images have two dimensions (height h and width w). Color images have the same dimensions but also 3 channels (red, green, blue). Each channel can be represented by a 2D matrix.

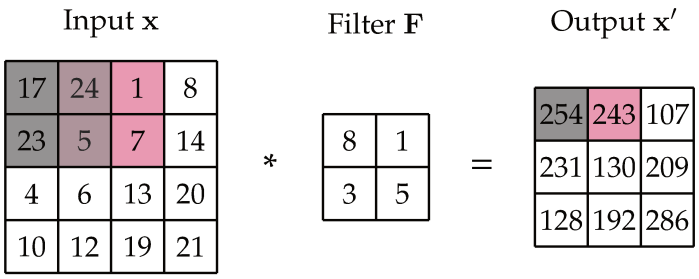


Figure 19 – Illustration of the 2D convolution. The input $\mathbf{x}$ contains 1 channel. The output $\mathbf{x}'$ contains 1 feature map.

Given a two-dimensional input with c channels $\mathbf{x} \in \mathbb{R}^{h \times w \times c}$. A $h_f \times w_f$ filter $\mathbf{F}$ applied on this input is a volume of size $h_f \times w_f \times c$. It produces an output called feature map of size $h' \times w' \times 1$. If k different filters are applied on the same input, the feature map has a size of $h' \times w' \times k$. Figure 19 illustrates the convolution mechanism. The filter $\mathbf{F}$ is first superposed to the upper left corner of the input $\mathbf{x}$ (gray case). The upper left coefficient of the output $\mathbf{x}'$ is computed by multiplying $\mathbf{F}$ with $\mathbf{x}$, superposed element per superposed element, and summing the results. Mathematically, the gray case of the feature map $\mathbf{x}'_{0,0}$ is equal to $\mathbf{x}_{0,0}\mathbf{F}_{0,0} + \mathbf{x}_{0,1}\mathbf{F}_{0,1} + \mathbf{x}_{0,2}\mathbf{F}_{0,2} + \mathbf{x}_{1,0}\mathbf{F}_{1,0} + \mathbf{x}_{1,1}\mathbf{F}_{1,1} + \mathbf{x}_{1,2}\mathbf{F}_{1,2} + \mathbf{x}_{2,0}\mathbf{F}_{2,0} + \mathbf{x}_{2,1}\mathbf{F}_{2,1} + \mathbf{x}_{2,2}\mathbf{F}_{2,2}$. Then, to obtain the value of the pink case of the feature map, the filter is slid to the right, and the operation is repeated. Convolutional layers also have a stride and a padding hyperparameters. ***Stride*** is the number of coefficients the filter moves over before computing a new value. ***Padding*** is a certain number of rows and columns of 0 added around the image. It enables, among others, to keep the same image size before and after the convolution. In the above example, stride $= 1$ and padding $= 0$.

Here are some advantages of the convolutional layer.

- ✓ As the same filter is locally applied on all portions of the image, the same patterns are detected independently of the position of the relevant information on the data samples.
- ✓ The number of parameters is independent of the size of the input: $\Theta(h_f w_f c k)$.

A DNN made of a succession of convolutional layers is called a ***Convolutional Neural Network (CNN)***. See Fig. 20 for an example. A CNN often contains ***pooling layers***. The purpose of this layer is to decrease the size of the image. Like in a convolutional layer, a filter is applied on the input. However, the filter does not contain any parameter. In max pooling, the filter outputs the maximal value of the small square of the image. In average pooling, it outputs the average of the values. The stride is often equal to 2 so that the height and the width of the image are divided by 2.

Layer exploiting a graph structure

The coefficients within a data sample are sometimes irregularly linked to each other (i.e. far from the grid structure of an image or from the line structure of time). In some cases, their relationships can be modeled with a graph (cf Section 1.4) in which each node is associated with a coefficient of the data sample. For instance, if the data sample was an image, each node would represent a pixel. As convolutional layers cannot handle irregular structures, new types of layers have been proposed in the literature.

Most of the time, layers handling a graph structure can be approximated by formula

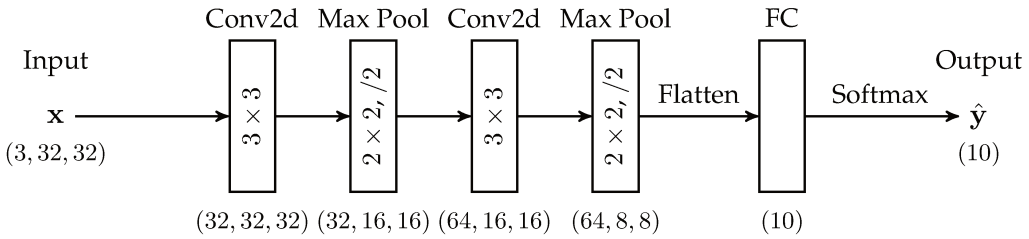


Figure 20 – CNN obtaining 67.19% accuracy on CIFAR-10. 3×3 means that a 3×3 filter is applied. $/2$ means that a stride equals to 2 is used. FC only contains a linear function. The number inside brackets are the shape of the tensors (number of channels, height, width) for the input, (number of feature maps, height, width) for the convolutional layers, (number of features) for the linear layer and the output. In total, this CNN contains about 60 thousand parameters.

similar to $\sigma(\mathbf{HxW})$, where $\mathbf{x} \in \mathbb{R}^{N \times d}$ is a data sample, N is the number of nodes in the graph and d the number of features associated with each node. σ is a non linear function. $\mathbf{H} \in \mathbb{R}^{N \times N}$ is a filter applying operations on the nodes of the graph. $\mathbf{W} \in \mathbb{R}^{d \times d'}$ is a parameter matrix allowing to learn d' new features. More details and examples are given in Chapter 1.4.4 dedicated to graph signal.

The advantages and limits of layers exploiting a graph structure are the following.

- ✓ The data structure is exploited through the filter $\mathbf{H}$.
- ✓ When the filter $\mathbf{H}$ is fixed, the number of parameters is independent of the size of the input:$\Theta(dd')$.
- ✗ Such layers are rarely robust to modifications of the graph structure. When the structure changes, the filter $\mathbf{H}$ may no longer be relevant [TGB18a].

A DNN made of a succession of layers exploiting a graph structure is called a ***Graph Neural Network (GNN)***.

ResNet architecture

ResNet [He+16] is an architecture created to train DNNs with a very big number of layers. Indeed, when the number of layers of DNNs is really large, the DNNs are not able to correctly perform on the training set because of vanishing/exploding gradients.

This is because the partial derivatives of the loss function with respect to the parameters are computed with the chain rule. As such, the partial derivatives are computed through products of other partial derivatives. Higher is the number of layers, higher is the number of terms in the product. When all terms are between 0 and 1, the partial derivatives of the loss end up really close to 0. The parameters are no longer updated. However, when all terms are higher than 1, the partial derivatives become really large and the values of the parameters become too large for a computer.

The main advance in ResNet is the creation of the residual block. Described in Fig. 21, the residual block has been designed to output by default the identity function. The non linear function σ used in this architecture is the ReLU function. Given a vector $\mathbf{a}^{[l]}$ obtained after the ReLU function of a layer l and a vector $\mathbf{a}^{[l+2]}$ obtained after the ReLU of the layer $l+2$. The residual block outputs $\mathbf{a}^{[l+2]} = \sigma(\mathbf{W}^{[l+2]}\mathbf{a}^{[l+1]} + \mathbf{b}^{[l+2]} + \mathbf{a}^{[l]})$. By regularizing $\mathbf{W}^{[l+2]}$ and $\mathbf{b}^{[l+2]}$ to avoid them taking large values, we easily have $\mathbf{W}^{[l+2]}\mathbf{a}^{[l+1]} + \mathbf{b}^{[l+2]} \approx 0$ and thus, $\mathbf{a}^{[l+2]} \approx \sigma(\mathbf{a}^{[l]}) = \mathbf{a}^{[l]}$. Figure 22 shows the ResNet-18 architecture, containing 18 layers with parameters.

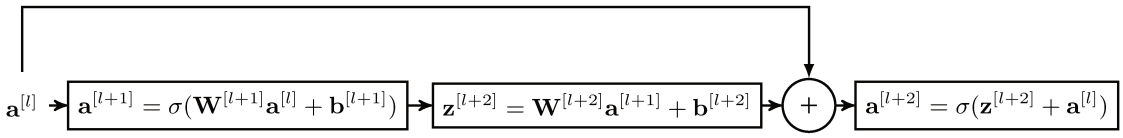


Figure 21 – Residual block used in ResNet.

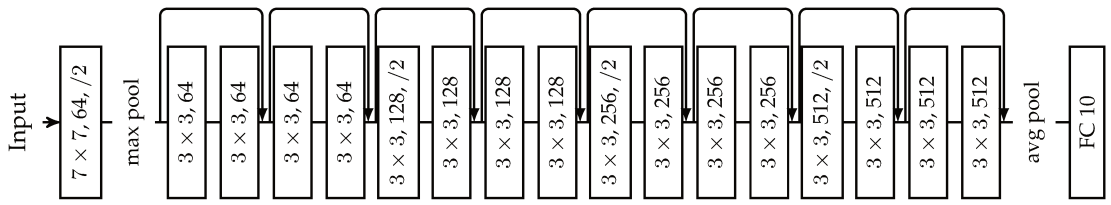


Figure 22 – ResNet-18 obtaining 78.85% accuracy on CIFAR-10. It contains about 12 million parameters.

1.3 Elements of Probability Theory

In this thesis, we sometimes need some basic notions of probability theory [Bis06] to take advantage of prior knowledge on the data distribution. In this section, we introduce the notations we use to describe random variables, random vectors and their distributions. We also introduce a few important properties and functions used in the next chapters.

1.3.1 Random variables

Definition 15

A ***random variable*** X is a variable whose values depend on the result of a random phenomenon.

In this manuscript, we study continuous real-valued random variables. For instance, in chapter 3, we use such a variable to model the stochastic activity of a brain region.

Formally, the probability distribution of X is characterized by its ***cumulative density function*** F_X. For a given $a \in \mathbb{R}$,

$$F_X(a) = P(X \leq a)\,. \tag{8}$$

It can also be characterized by a ***probability density function*** f_X such that,

$$\int_{-\infty}^{a} f_X(x)dx = P(X \leq a)\,. \tag{9}$$

The ***joint distribution*** of two random variables X and Y is characterized by $F_{X,Y}$ defined as:

$$F_{X,Y}(a,b) = \int_{-\infty}^{a}\int_{-\infty}^{b} f_{X,Y}(x,y)dxdy = P(X \leq x, Y \leq y)\,. \tag{10}$$

The function f_X is called the ***marginal probability density function*** of X. The function $f_{X|Y=y}$ is the ***conditional probability density function*** of X given the occurrence of the value y of Y.

Several properties of a random variable can be described by the following functions.

Definition 16

The ***expected value*** of X, denoted $\mathrm{E}(X)$, measures the average of the values taken by the random variable weighted by their probabilities: $\mathrm{E}(X) = \int_{-\infty}^{+\infty} x f_X(x)dx$.

A random variable is said to be centered when its expected value is zero.

Definition 17

The ***variance*** $\mathrm{V}(X)$ measures the dispersion of the outcomes of the random variable with respect to the expected value: $\mathrm{V}(X) = \mathrm{E}((x - \mathrm{E}(X))^2)$. The ***standard deviation*** $\sigma = \sqrt{\mathrm{V}(X)}$ is more often reported than the variance as it is homogeneous with the outcomes.

A random variable is said to be reduced when its variance is one.

Definition 18

The ***covariance*** $\mathrm{Cov}(X,Y)$ measures to what extent two random variables jointly evolve together: $\mathrm{Cov}(X,Y) = \mathrm{E}((X - \mathrm{E}(X))(Y - \mathrm{E}(Y)) = \mathrm{E}(XY) - \mathrm{E}(X)\mathrm{E}(Y)$.

When we do not know the distribution of a random variable X, but we have a finite number of points $x_1, \ldots, x_N$ drawn from the distribution, we can approximate its expected value and variance with unbiased estimators, respectively the empirical mean and the empirical variance:

$$\overline{X} = \frac{1}{N}\sum_{n=1}^{N} x_n \text{ and } \overline{\mathrm{V}}(X) = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \overline{X})^2 \,. \tag{11}$$

These estimators are unbiased because their expected values are equal to the values they try to estimate. Similarly, the empirical covariance between two variables X and Y can be estimated from a finite number of points $(x_1, y_1) \ldots, (x_N, y_N)$:

$$\overline{\mathrm{Cov}}(X, Y) = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \overline{X})(y_n - \overline{Y}) \,. \tag{12}$$

An important notion in probability is the notion of ***independence*** between random variables. Two random variables are independent when the realization of one variable does not affect the probability distribution of the other one. Formally, two real random variables X and Y are independent if,

$$\forall x, y, F_{X,Y}(x, y) = F_X(x)F_Y(y) \tag{13}$$

or equivalently if f_X, f_Y and $f_{X,Y}$ exist, if

$$\forall x, y, f_{X,Y}(x, y) = f_X(x)f_Y(y) \,. \tag{14}$$

Considering a finite set of n random variables, we can define two notions of independence. The weak one is the notion of pairwise independence describing a set where any pair of random variables from this set are independent from each other. The stronger definition is the one called mutual independence. Each variable is independent of any subset of the others. Formally, random variables $X_1, \ldots X_n$ are said to be mutually independent if,

$$\forall x_1, \ldots, x_n, F_{X_1,\ldots,X_n}(x_1, \ldots, x_n) = F_{X_1}(x_1) \ldots F_{X_n}(x_n) \tag{15}$$

or equivalently when defined, if

$$\forall x_1, \ldots, x_n, f_{X_1,\ldots,X_n}(x_1, \ldots, x_n) = f_{X_1}(x_1) \ldots f_{X_n}(x_n) \,. \tag{16}$$

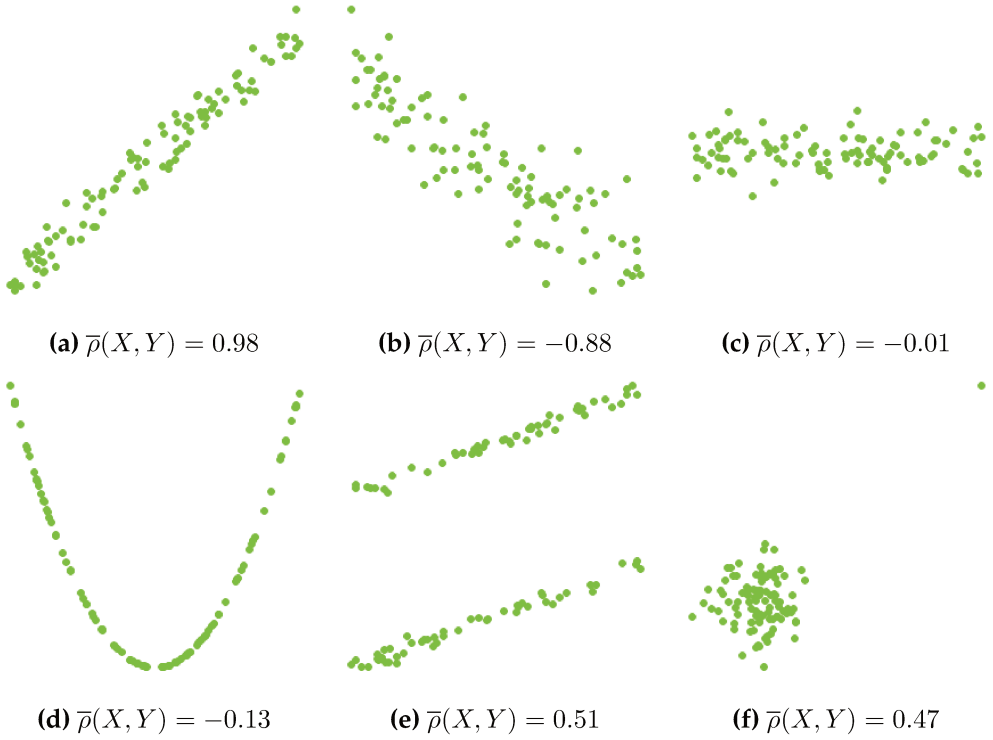
Another indicator which is used a lot in the literature to evaluate the relationship between two variables is the Pearson correlation coefficient.

Definition 19

The ***linear correlation coefficient*** $\rho(X, Y)$ between two random variables X and Y, also called ***Pearson correlation coefficient***, is defined as: $\rho(X, Y) = \frac{\mathrm{Cov}(X,Y)}{\sigma(X)\sigma(Y)}$.

Here a few properties of the Pearson correlation coefficient.

- When X and Y are independent, $\rho(X, Y) = 0$. The converse is not always true.
- When X and Y have similar behaviors, $\rho(X, Y) = 1$.



(a) $\overline{\rho}(X,Y) = 0.98$ **(b)** $\overline{\rho}(X,Y) = -0.88$ **(c)** $\overline{\rho}(X,Y) = -0.01$

(d) $\overline{\rho}(X,Y) = -0.13$ **(e)** $\overline{\rho}(X,Y) = 0.51$ **(f)** $\overline{\rho}(X,Y) = 0.47$

Figure 23 – Pearson's correlation coefficient $\overline{\rho}$ computed between two random variables X and Y. Revealed by $\overline{\rho}$: linear relationships (a-b), absence of relationship (c). Not revealed by $\overline{\rho}$: non-linear relationships (d-e). Misleading case: absence of relationship but the presence of an outlier increases the $\overline{\rho}$ value (f).

- When X and Y have opposite behaviors, $\rho(X,Y) = -1$.

What do similar or opposite behaviors mean in term of random variables? Indeed, ρ is only sensible to linear relationships between variables. See Fig. 23 for an illustration.

The Pearson correlation coefficient can be estimated from a set of points with:

$$\overline{\rho}(X,Y) = \frac{\overline{\text{Cov}}(X,Y)}{\overline{\sigma}(X)\overline{\sigma}(Y)} \tag{17}$$

Last but not least, we would like to mention one of the most useful theorem for machine learning: the Bayes'theorem. Given two random variables X and Y, it states that:

$$f_{Y|X} = \frac{f_{X|Y} f_Y}{f_X}\,. \tag{18}$$

For instance, in classification, the variable Y represents the parameters of the model $\mathbf{W}$ and the variable X the data $\mathbf{x}$. Informally, we can rewrite $P(\mathbf{W}|\mathbf{x}) = \frac{P(\mathbf{x}|\mathbf{W})P(\mathbf{W})}{P(\mathbf{x})}$. In some classification techniques, the trained parameters are the ones maximizing $P(\mathbf{x}|\mathbf{W})$, probability that is known as the ***likelihood*** of the data given the parameters $P(\mathbf{x}|\mathbf{W})$ (e.g. LR in subsection 1.1.4 and DNNs in section 1.2.1 as maximizing the likelihood amounts to minimizing the cross-entropy loss). Sometimes, a ***prior probability*** $P(\mathbf{W})$ linked to specific knowledge on the parameters is available. In these cases, other classification techniques aim at maximizing the ***posterior probability*** of the parameters $P(\mathbf{W}|\mathbf{x})$ (e.g. LDA in subsection 1.1.4).

1.3.2 Gaussian distribution

In this manuscript, we consider random variables following Gaussian distributions. Here is a small subsection explaining the interesting properties of the Gaussian distribution.

Definition 20

A random variable X following a ***Gaussian distribution*** is denoted $X \sim \mathcal{N}(\mu, \sigma^2)$. Its probability density function is fully characterized by its expected value μ and its variance σ^2:

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp \frac{-(x-\mu)^2}{2\sigma^2}. \tag{19}$$

The Gaussian distribution enables to characterize many natural phenomena. Given (X_n) a sequence of independent and identically distributed (i.i.d.) random variables of expected value μ and of variance σ^2. The central limit theorem establishes that the distribution of the average $S_n = \frac{X_1+\cdots+X_n}{n}$ tends toward $\mathcal{N}(\mu, \frac{\sigma^2}{n})$. This theorem states that whatever their distributions are, the average of i.i.d random variables having the same mean and variance converges in distribution to a Gaussian distribution. This theorem confirms that it is natural to model noise in natural phenomena with Gaussian variables. For instance, consider the brain activity signal. A part of the noise contained in the measure of the activity in a brain region can be seen as the sum of the outputs of many independent neurons.

The Gaussian distribution has several useful stability properties. Given $a, b \in \mathbb{R}$, $X \sim \mathcal{N}(\mu, \sigma^2)$, $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$.

- By linearity, $aX + b \sim \mathcal{N}(a\mu + b, a^2\sigma^2)$.
- By additivity, when X_1 and X_2 are independent, $X_1 + X_2 \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$.

This generalizes to n mutually independent variables. In particular, when $X_1 \sim \mathcal{N}(0,1), X_2 \sim \mathcal{N}(0,1) \ldots X_n \sim \mathcal{N}(0,1)$, $S_n \sim \mathcal{N}(0, \frac{1}{n})$.

1.3.3 Random vectors

Definition 21

A ***random vector*** is a vector whose coefficients are random variables. We denote it as a column vector $\boldsymbol{X} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)$.

A random vector is characterized by the joint distribution of its coefficients. The cumulative density function of $\boldsymbol{X}$ is:

$$\begin{aligned} F_{\boldsymbol{X}} : \mathbb{R}^n &\to [0,1] \\ \mathbf{x} &\mapsto P(\boldsymbol{X}_1 \leq \mathbf{x}_1, \ldots, \boldsymbol{X}_n \leq \mathbf{x}_n). \end{aligned}$$

Previously, we have taken the example of a random variable modeling the activity in a brain region. Continuing this example, a random vector would model the activities of several brain regions at the same time, each of its coefficients being a random variable associated with a brain region. As the joint model takes into account the covariance between brain regions, it is more powerful.

The expected value of the random vector $\boldsymbol{X}$ is defined by computing independently the expected value of each coefficient of $\boldsymbol{X}$. It is represented by the column vector $\boldsymbol{\mu} = (\mathrm{E}(\boldsymbol{X}_1), \ldots, \mathrm{E}(\boldsymbol{X}_n))^\intercal$. The variances and the covariances between the coefficients

of $\boldsymbol{X}$ are stored in a ***covariance matrix*** $\boldsymbol{\Sigma}$, where $\boldsymbol{\Sigma}_{i,j}$ is the covariance between $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$. Formally, $\boldsymbol{\Sigma} = E((\boldsymbol{X} - \mathrm{E}(\boldsymbol{X}))(\boldsymbol{X} - \mathrm{E}(\boldsymbol{X}))^\intercal) = \mathrm{E}(\boldsymbol{X}\boldsymbol{X}^\intercal) - \mathrm{E}(\boldsymbol{X})\mathrm{E}(\boldsymbol{X})^\intercal$.

1.3.4 Multivariate Gaussian distribution

The multivariate Gaussian distribution is the extension of the Gaussian distribution to random vectors.

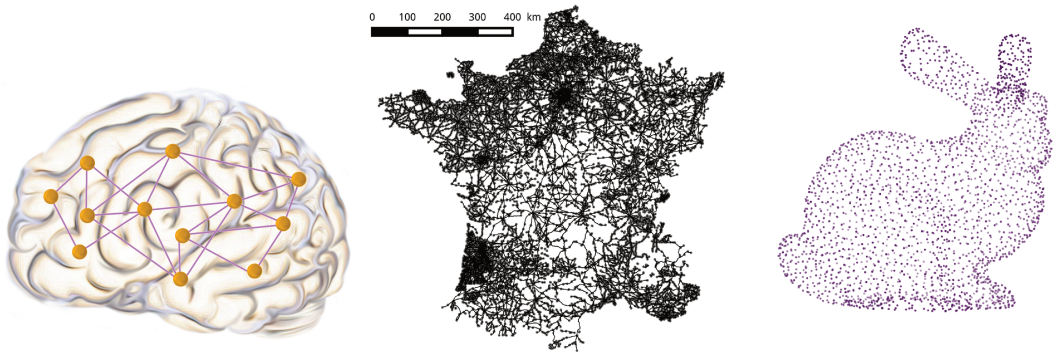
Definition 22

A random vector $\boldsymbol{X}$ follows a ***multivariate Gaussian distribution*** when any linear combination of its coefficients is a Gaussian random variable. In the case where the covariance matrix of $\boldsymbol{X}$ is invertible, its probability density function can be computed. Given $|\boldsymbol{\Sigma}|$ the determinant of $\boldsymbol{\Sigma}$, it is defined by:

$$f_{\boldsymbol{X}}(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\intercal \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right). \tag{20}$$

Here are some properties of the multivariate Gaussian distribution. Given $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{X}}, \boldsymbol{\Sigma}_{\boldsymbol{X}})$, $\boldsymbol{Y} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Y}}, \boldsymbol{\Sigma}_{\boldsymbol{Y}})$ and $\mathbf{A} \in \mathbb{R}^{n \times n}$.

- The marginal distributions of $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ are Gaussian.
- The conditional distributions (e.g. $\boldsymbol{X}_1 | \boldsymbol{X}_2 = \mathrm{x}_2$) are also Gaussian.
- When $\boldsymbol{\Sigma}_{i,j} = 0$, $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$ are independent.
- When $\boldsymbol{\Sigma}$ is diagonal, $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ are mutually independent.
- When $\boldsymbol{Z} = \mathbf{A}\boldsymbol{X}$, $\boldsymbol{Z} \sim \mathcal{N}(\mathbf{A}\boldsymbol{\mu}_{\boldsymbol{X}}, \mathbf{A}\boldsymbol{\Sigma}_{\boldsymbol{X}}\mathbf{A}^\intercal)$.
- When $\boldsymbol{X}$ and $\boldsymbol{Y}$ are independent, $\boldsymbol{X} + \boldsymbol{Y} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{X}} + \boldsymbol{\mu}_{\boldsymbol{Y}}, \boldsymbol{\Sigma}_{\boldsymbol{X}} + \boldsymbol{\Sigma}_{\boldsymbol{Y}})$.



(a) Brain network. **(b)** Road network. **(c)** Point cloud representing a rabbit.

Figure 24 – Examples of graph signals. The point cloud is not a graph in itself but it can be used to generate one by, for instance, connecting the closest points. (b) comes from [PGB15]. (c) has been generated with the Python package PyGSP [Def+17] using the previous works of [TL94].

1.4 Graph Signal Processing

When we observe signals composed of several coefficients, it is common that these coefficients are correlated. The complex interactions within a signal can be modeled by a graph. For instance, brain signals reflecting the activation of certain brain regions can be modeled as graph signals (Fig. 24a). The nodes of the graph correspond to brain regions. The edges between nodes reflect the natural neuronal connections between these regions. A road network can also be seen as a graph signal where nodes model the cities and the edges modeled the roads between cities (Fig. 24b). A signal on this graph could be the number of people infected by a virus in each city. A last example is linked to point clouds (Fig. 24c). The nodes of the graph are the points and the edges depend on the distances between the points. The signal could be the temperature on each point. In these examples, the graph and the signals are complementary in order to understand the described phenomena. Exploiting this complementarity is crucial for many applications [Zha+21c; LGO21]. Among them, in this manuscript, we address questions related to brain activity decoding in chapter 3.

Processing graph signals is not trivial as classical signal processing techniques are not optimal on irregular spaces. They have not been thought to account for this type of structure. The purpose of Graph Signal Processing (GSP) is to define a framework to handle these graph signals by decomposing them into components of different frequencies and to design filters that can extract or modify parts of a graph signal according to these frequencies.

Definition 23

A ***graph signal*** is represented by a N-dimensional vector $\mathbf{x}$ in which the coefficient $\mathbf{x}_i$ is associated with the i^{th} node of a graph $\mathcal{G}$.

In this manuscript, the ***graph*** $\mathcal{G}$ is always a weighted undirected graph composed of a set of N ***vertices*** $\mathcal{V} = \{1, \dots, N\}$ and a set of ***edges*** $\mathcal{E}$ storing pairs of vertices (i, j). The weights of the interactions between signal components are stored in an ***adjacency matrix*** $\mathbf{A}$, $\mathbf{A}_{i,j} \geq 0$ being the weight of the edge linking the nodes i and j. The vertices may also be called ***nodes***.

In the following, we introduce the main principles behind GSP, necessary to understand the works in this thesis.

1.4.1 Graph Laplacian

In GSP, we often use an operator called combinatorial graph Laplacian.

Definition 24

Let $\mathbf{D}$ be the diagonal ***degree matrix*** of the graph defined as $\mathbf{D}_{i,i} = \sum_{j=1}^{N} A_{i,j}$. The ***combinatorial graph Laplacian*** is defined as: $\mathbf{L} = \mathbf{D} - \mathbf{A}$.

The name of this operator comes from the physic field. In this field, the Laplace operator defines the divergence of the gradient of a function. In other words, it characterizes the spreading out of a function around each coordinate. Similarly, on a graph, multiplying the Laplacian $\mathbf{L}$ by a graph signal $\mathbf{x}$ amounts to replace the value of the signal on each node by the weighted sum of the differences between the value on this node and the values on its neighboring nodes. Mathematically, $(\mathbf{L}\mathbf{x})_i = \sum_{j=1}^{N} \mathbf{A}_{i,j}(\mathbf{x}_i - \mathbf{x}_j)$.

The combinatorial Laplacian is diagonalizable in an orthonormal basis. Indeed, according to the ***spectral theorem***, a real symmetric matrix is diagonalizable in a real orthogonal basis and its eigenvalues are real. This diagonalization is often called ***spectral decomposition***.

This Laplacian is studied because it has several interesting properties. First, its eigenvalues are always non-negative, and in particular the smaller is always null. Second, the multiplicity of eigenvalue 0 is equal to the number of connected components in the graph. The quadratic form of the Laplacian also has a particular meaning. In the literature, it is called smoothness. The smaller the value of the quadratic form on a graph signal is, the more neighboring nodes have similar values on them. Examples of more or less smooth graph signals are given in Fig. 25.

Definition 25

Let $\mathbf{L}$ be the combinatorial graph Laplacian of the graph $\mathcal{G}$ and $\mathbf{A}$ its adjacency matrix. The ***smoothness*** of a graph signal $\mathbf{x}$ on $\mathcal{G}$ is defined as:

$$Q_{\mathbf{L}}(\mathbf{x}) = \mathbf{x}^{\intercal}\mathbf{L}\mathbf{x} = \sum_{(i,j)\in\mathcal{E}} \mathbf{A}_{i,j}(\mathbf{x}_i - \mathbf{x}_j)^2 \ .$$

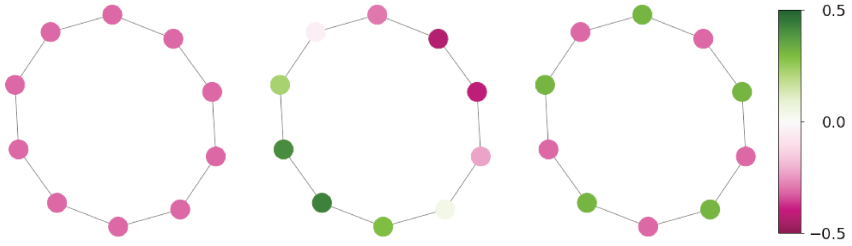
Sometimes, in this manuscript, we also use a ***normalized version of the graph Laplacian*** $\mathbf{L}_n = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$. The advantage is that its eigenvalues are all between 0 and 2. The quadratic form obtained with $\mathbf{L}_n$ becomes:

$$Q_{\mathbf{L}_n}(\mathbf{x}) = \mathbf{x}^{\intercal}\mathbf{L}_n\mathbf{x} = \sum_{(i,j)\in\mathcal{E}} \mathbf{A}_{i,j}\left(\frac{\mathbf{x}_i}{\sqrt{\mathbf{D}_{i,i}}} - \frac{\mathbf{x}_j}{\sqrt{\mathbf{D}_{j,j}}}\right)^2 \ .$$

Another interesting property of the Laplacian is that it can be used to decompose a signal onto components of different frequencies by analogy with the classical Fourier transform, as shown in the next subsection.

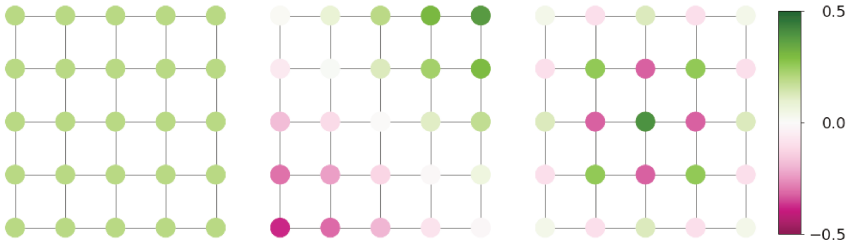
1.4.2 Graph Fourier transform

Generally speaking, the Graph Fourier Transform (GFT) is defined from a graph-shift operator representing the graph $\mathcal{G}$, provided that it is diagonalizable in a real orthogonal basis [HBL15; Shu+13; TGB18b]. A ***Graph-Shift Operator (GSO)*** is a matrix $\mathbf{S} \in$



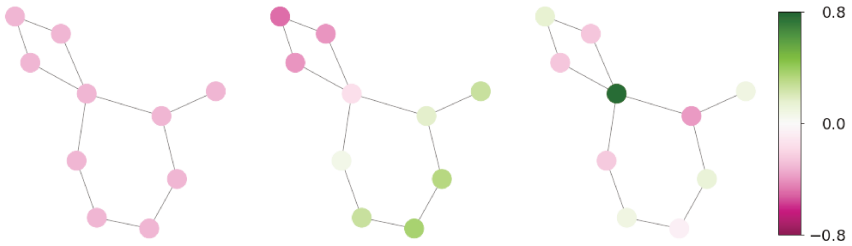
$Q_{\mathbf{L}}(\mathbf{u}_0) = 0.000$ $Q_{\mathbf{L}}(\mathbf{u}_1) = 0.382$ $Q_{\mathbf{L}}(\mathbf{u}_{24}) = 4.000$

(a) Ring graph.



$Q_{\mathbf{L}}(\mathbf{u}_0) = 0.000$ $Q_{\mathbf{L}}(\mathbf{u}_1) = 0.382$ $Q_{\mathbf{L}}(\mathbf{u}_9) = 7.236$

(b) 2-dimensional grid graph.



$Q_{\mathbf{L}}(\mathbf{u}_0) = 0.000$ $Q_{\mathbf{L}}(\mathbf{u}_1) = 0.376$ $Q_{\mathbf{L}}(\mathbf{u}_9) = 5.509$

(c) Erdős-Rényi graph randomly connecting edges with a probability equal to 0.25.

Figure 25 – Illustration of the notions of smoothness and frequency on graphs. For each graph, the vectors $\mathbf{u}_i$ are the eigenvectors associated with the frequencies λ_i (eigenvalues) of their combinatorial Laplacian. The frequencies λ_i are equal to the smoothness of their associated eigenvector $Q_{\mathbf{L}}(\mathbf{u}_i)$. Figures have been generated with PyGSP [Def+17].

$\mathbb{R}^{N\times N}$ representing the global structure of the graph with the only following constraint: $\mathbf{S}_{i,j}$ can be non-zero only if $i = j$ or if $(i,j) \in \mathcal{E}$. For instance, $\mathbf{S}$ could be the combinatorial Laplacian or the adjacency matrix. In this manuscript, we only consider real symmetric GSOs, so they always admit a spectral decomposition.

Let us formally define the GFT. As $\mathbf{S}$ is real symmetric, it exists a matrix $\mathbf{U} \in \mathrm{O}(N)$ (orthogonal group) and a diagonal matrix $\mathbf{\Lambda} \in \mathbb{R}^{N\times N}$ such that $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$. The columns of the matrix $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_N]$ contain the eigenvectors of $\mathbf{S}$ and $\mathbf{\Lambda}$ contains the associated eigenvalues listed in increasing order ($\lambda_1 \leq \dots \leq \lambda_N$).

Definition 26

The ***Graph Fourier Transform (GFT)*** of a graph signal $\mathbf{x} \in \mathbb{R}^N$ is $\hat{\mathbf{x}} = \mathbf{U}^\top\mathbf{x}$. It enables to go from the node basis to the eigenvector basis. The ***inverse GFT*** is simply defined by $\mathbf{x} = \mathbf{U}\hat{\mathbf{x}}$.

Note that the GFT is ill-defined when the multiplicity of some eigenvalues is higher than 1. In that case, the GFT can be computed on an infinite number of sets of eigenvectors. In practice, this issue is often neglected.

Definition 27

The ***graph frequency*** associated with an eigenvector $\mathbf{u}_i$ is its eigenvalue λ_i.

A common choice for $\mathbf{S}$ is the combinatorial graph Laplacian. In Fig. 25, several eigenvectors computed from different graphs are shown. One main reason for this choice is the link between the notion of graph frequency and the classical notion of frequency representing how fast a signal oscillates. Indeed, the smoothness of an eigenvector $\mathbf{u}_k$ shows that λ_k is linked to the sum of the local variations of $\mathbf{u}_k$ on $\mathcal{G}$, weighted by the edges of the graph:

$$Q_{\mathbf{L}}(\mathbf{u}_k) = \sum_{(i,j)\in\mathcal{E}} A_{i,j}(\mathbf{u}_{k,i} - \mathbf{u}_{k,j})^2 = \lambda_k \left\|\mathbf{u}_k\right\|_2^2 = \lambda_k \ .$$

1.4.3 Graph filters

Graph filters are functions that can extract or modify parts of graph signals [TGB18b]. Several strategies exist to define them.

One strategy consists in exploiting the GFT to act on the spectral components independently. Given a graph signal $\mathbf{x}$ and a GFT associated with a GSO $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, $\mathbf{U}$ and $\mathbf{\Lambda}$ being as above the eigenvectors matrix and the eigenvalues diagonal matrix. The graph filter is a function of the eigenvalues:

$$\begin{aligned} h\colon \mathbb{R} &\to \mathbb{R} \\ \lambda &\mapsto h(\lambda). \end{aligned}$$

To apply the filter, the graph signal is simply projected into the spectral space: $\hat{\mathbf{x}} = \mathbf{U}^\top\mathbf{x}$. Then, each component of the signal is filtered independently of the others, before being projected back to the vertices space. Thus, the resulting filtered signal is $\tilde{\mathbf{x}} = \mathbf{U}h(\mathbf{\Lambda})\hat{\mathbf{x}}$, where $h(\mathbf{\Lambda})$ is the diagonal matrix such that $h(\mathbf{\Lambda})_{i,i} = h(\lambda_i)$. Denoting by $\mathbf{H} = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^\top$ the graph filter matrix, we have $\tilde{\mathbf{x}} = \mathbf{H}\mathbf{x}$.

The advantage of such filter is that it is directly related to the notion of frequency on a graph. The drawback is that it requires computing the GFT of $\mathbf{S}$. The complexity of

the eigendecomposition of $\mathbf{S}$ is bounded by $\mathcal{O}(N^3)$ [PCZ+98] and the complexity of the matrix-vector multiplication required to project the graph signal into the GFT space is $\Theta(N^2)$.

A second strategy consists in multiplying the graph signal by a polynomial function of the graph-shift operator [DBV16]. Given the parameters $K \in \mathbb{N}$, $\alpha_0, \ldots, \alpha_K \in \mathbb{R}$, a K-order polynomial graph filter is represented by $\mathbf{H} = \sum_{k=0}^{K} \alpha_k \mathbf{S}^k$. In fact, this strategy can be seen as a simplification of the first one, where the function h is approximated by a polynomial function. Given $\lambda \in \mathbb{R}$, $h(\lambda) = \sum_{k=0}^{K} \alpha_k \lambda^k$. So, we get $h(\mathbf{\Lambda}) = \sum_{k=0}^{K} \alpha_k \mathbf{\Lambda}^k$. The filtered signal becomes $\tilde{\mathbf{x}} = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^\intercal\mathbf{x} = \sum_{k=0}^{K} \alpha_k \mathbf{U}\mathbf{\Lambda}^k\mathbf{U}^\intercal\mathbf{x} = \sum_{k=0}^{K} \alpha_k \mathbf{S}^k\mathbf{x}$.

An advantage of this filter is that it is K-hop localized. In other words, the filtered value at one vertex will only depend on the values of its neighbors at at most, K connections away from it.

Proof. This is due to the design of the GSO $\mathbf{S}$. When $\mathbf{S}$ is multiplied by $\mathbf{x}$, the new value on a vertex only depends on the values of its neighbors. When $\mathbf{S}$ is multiplied once again by $\mathbf{Sx}$, the new value on a vertex depends on the values of the neighbors of the previous signal which are the neighbors of the neighbors of $\mathbf{x}$. □

A second advantage of this filter is its complexity. It does not required to compute a GFT. As it contains $K + 1$ vector-matrix multiplications to recursively compute the powers of $\mathbf{S}$ ($\mathbf{S}^{k+1}\mathbf{x} = \mathbf{S}(\mathbf{S}^k\mathbf{x})$), it requires $\Theta(KN^2)$ operations. The sparsity of the graph can be taken into account to reduce the complexity to $\Theta(K|\mathcal{E}|)$, $|\mathcal{E}|$ being the number of edges of the graph.

The parameters of the graph filters are either chosen using prior knowledge or learned. In the next session, these filters are combined with DNNs to extract information from irregularly structured data.

1.4.4 Graph neural networks

When dealing with classification problems on graph signals, graph filters can be used within the layers of DNNs giving birth to Graph Neural Networks (GNNs) [Bro+17]. In the following, a few examples are described.

Let us consider a classification problem with C classes. We denote the graph signals $\mathbf{x} \in \mathbb{R}^{N \times d}$, where N is the number of nodes in the graph and d is the number of features associated to each node. The classes associated with these graph signals are integers between 0 and $C - 1$. They are stored within binary vectors $\mathbf{y} \in \mathbb{R}^C$ where $\mathbf{y}_i = 1$ if the class of the signal is i and $\mathbf{y} = 0$ otherwise.

As stated before, graph filters can be used to create DNNs layers taking into account the graph structure. These layers often contain a graph filter added to a MLP and followed by a non-linear function. Formally, given a non-linear function σ, such layers can often be defined as $\sigma(\mathbf{HxW})$ [Bon+19c]. $\mathbf{W} \in \mathbb{R}^{d \times d'}$ is a parameter matrix allowing to learn d' new features. $\mathbf{H} \in \mathbb{R}^{N \times N}$ is a filter applying operations on the nodes of the graph. Here are a few examples of GNNs using different kind of filters.

ChebNet

One of the first GNN published in the literature is called ChebNet [DBV16]. The idea is to learn a polynomial filter approximated by a truncated Chebyshev expansion at each

layer: $\mathbf{H} = \sum_{k=0}^{K} \alpha_k T_k(\mathbf{S})$ with $\mathbf{S} = \frac{2}{\lambda_{\max}} \mathbf{L}_n - \mathbf{I}$.

Graph convolutional network

Graph Convolutional Network (GCN) [KW17] uses only first order polynomial filters ($K = 1$) assuming that more complexity can be captured by increasing the number of layers of the DNN. In that case, $\mathbf{H} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}$ with $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}}$ degree matrix of $\tilde{\mathbf{A}}$.

Simplifying graph convolutional network

In Simplifying Graph Convolutional Networks (SGC) [Wu+19], the graph filter is used as a preprocessing step before a LR. It is a simplification of GCN where the non-linear functions between layers have been removed. In this method, the filter $\mathbf{H} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}$ is applied before a LR. Given a parameter $m \in \mathbb{N}$, a weight matrix $\mathbf{W} \in \mathbb{R}^{d \times C}$ and a softmax function, the learned classes $\hat{\mathbf{y}}$ are obtained by: $\hat{\mathbf{y}} = \text{softmax}(\mathbf{H}^m \mathbf{x} \mathbf{W})$. The parameter m represents the number of hop-connections considered to average the signal on. In other words, if $m = 1$, the value of the signal on one vertex will be a linear combination of the values of its neighbors. If $m = 2$, it will be a linear combination of the values of the neighbors of its neighbors and so on.

Now, the question is which method to choose when one wants to solve a particular classification problem. It depends on the expressiveness required for the problem. Networks often favor low-frequencies components whereas in some problems high-frequencies would have been more relevant. The GNN has to be designed carefully while keeping in mind this fact. Reviews on these issues are numerous [VOF20; Xu+19; Bo+21; DBY19]. We also published an article proposing a unified deep learning formalism for processing graph signals at a French conference (see [Bon+19c] for the English version):

Myriam Bontonou et al. “Un modèle unifié pour la classification de signaux sur graphe avec de l’apprentissage profond”. In: *GRETSI*. 2020.

Other elements such as pooling layers decreasing the number of nodes of the graph have been developed to improve the performance of GNN, to ease their training and to reduce their complexity. An overview has been published in [Che+20].

Now that we have presented the main notions behind machine learning, deep learning, probability theory and graph signal processing used through this manuscript, we focus on the problems raised by learning from few examples.

Chapter 2

Learning with Few Examples

Contents

In many areas, collecting data is expensive or time-consuming. For instance, some companies are interested in specific image recognition tasks based on their small private dataset; some researchers try to decode brain activity from scarce fMRI data; some scientists collect data to assess the impact of humans on earth [Sch+19] but the amount of data is small compared to the complexity of the question... In other areas, a large amount of data is available but it is challenging to label it. For instance, in the following examples, thousands of images are registered but few are annotated: the food and agriculture organization wants to evaluate the state of crops from Earth observations to enhance global food security [Tse+21] but it is not straightforward to annotate the crops by hand; some governments want to detect private swimming pools from drone images to verify tax information [DM19]; some companies tries to recognize road signs, pedestrians or obstacles from images of cameras on cars [Li+20]... In other cases, a lot of annotated data is available but some classes contain very few examples. For instance, in the context of rare event detection, an organization may store the state of its system regularly and ask an employee to associate a label with it. Over a period of time, only a few rare events will occur. The organization will therefore collect a lot of data related to normal situations but very few examples of the events it wants to detect.

All problems with *few labeled data* are gathered under the name ***few-shot problems***, *shot* meaning *training example*. The problems with *few labeled examples but with additional unlabeled data* are called ***few-label problems***.

Since the emergence of the field of machine learning, researchers have faced few-shot problems [Thr98]. In deep learning, the lack of labeled data prevents to efficiently train a DNN end-to-end as it leads to overfitting (i.e. the DNN does not generalize well to new examples). However, the largely demonstrated potential of deep learning for efficient representation learning and feature extraction can be used to make few-shot learning easier. Sometimes, it is possible to transfer knowledge learned on a big generic dataset containing data somehow similar to that of the specific problem. It should be noted that the general principles behind transfer learning [Yan+20] are not only applicable to few-shot problems. In practice, a DNN can be pre-trained on a generic dataset to learn to extract good representations of the data. In the context of classification, a good representation is a representation easily classifiable. It is then used on the few-shot problem data in the hope that the learned representations will untangle the classes. At best, the new classes will be projected in a space in which they can be separated with linear boundaries as in Fig. 15 on page 48.

In this chapter, we focus on classification problems. We show how to solve few-shot problems with DNNs. Section 2.1 contains an overview of existing few-shot learning techniques. Then, section 2.2 is dedicated to the following question: when few examples are available, how to evaluate the generalizability of a classifier? At the end, a summary highlights our contributions to this area.

2.1 Few-Shot Learning for Classification

Few-shot problems are formalized and described with a specific vocabulary. In this section, we first detail this vocabulary before describing some deep learning methods designed to solve few-shot problems.

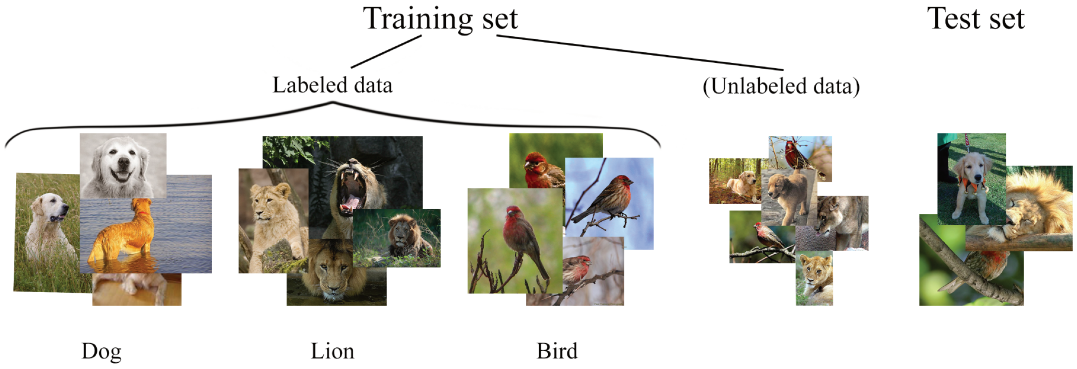


Figure 26 – Example of a 3-way 4-shot 1-query few-shot problem. When additional unlabeled data are also available during training, the problem becomes a few-label problem. Images come from mini-ImageNet [Vin+16].

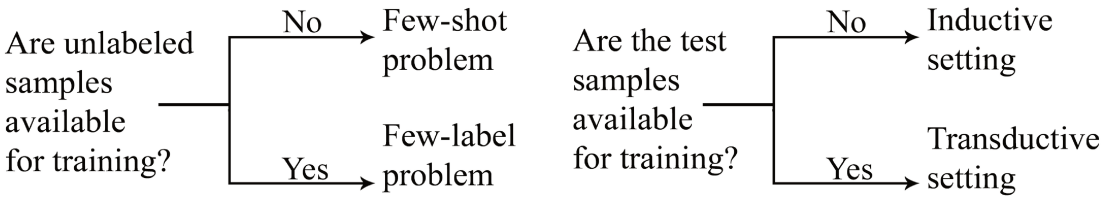


Figure 27 – Vocabulary used to characterize few-shot problems.

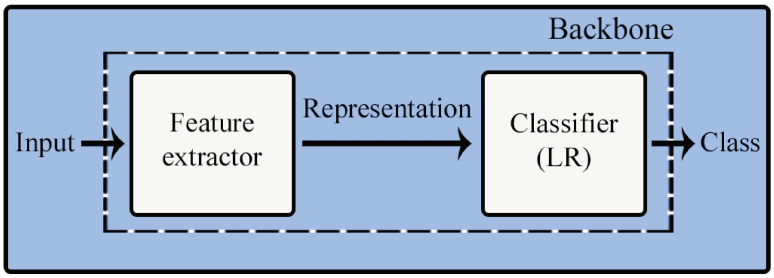
2.1.1 Formalization of the few-shot problems

In a few-shot classification problem, a certain number of classes has to be classified using a small number of examples. The training set contains a few labeled training examples. The training examples are called ***shots*** and the classes are called ***ways***. For academic research, problems with few examples are artificially generated from datasets containing more examples. In these cases, test sets are also generated to evaluate the generalization ability of the trained algorithms. The test examples are called ***queries***. A few-shot problem is said to be N-way K-shot Q-query when it contains N classes, K shots and Q queries. Figure 26 is an example of 3-way 4-shot 1-query problem (without considering the unlabeled examples in parentheses).

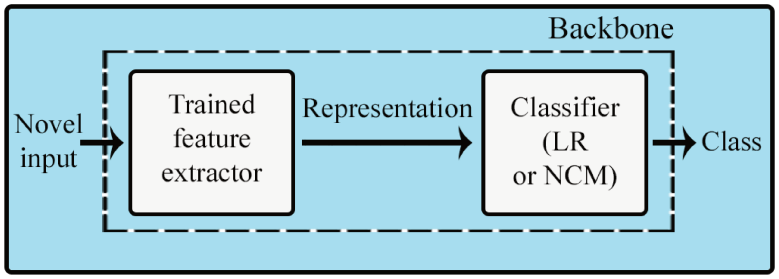
Problems with additional unlabeled examples are called few-label problems. In Fig. 26, the few-shot problem becomes a few-label problem when the unlabeled examples are used for training (in parentheses on the figure). The last notions to remember are the notions of ***inductive*** and ***transductive*** settings. The difference between the two settings lies in the question: are the test examples available during training (obviously without their labels)? If they are not, the setting is inductive. If they are, the setting is transductive. Vocabulary is recalled in Fig. 27.

2.1.2 Learning methods

In this subsection, we present some few-shot learning methods that are really efficient on natural image classification. Few-shot methods can be divided into three main families: a family inspired by transfer learning that we call embedding-based, a family inspired by meta-learning and a last family trying to artificially generate new examples



(a) Pre-training on the generic dataset.



(b) Adaptation to the few-shot problem.

Figure 28 – General framework of few-shot learning methods.

that we call hallucination-based. Note that in our experiments, we do not use methods based on artificial generation of examples. These methods are conceptually distant from the other families and they often require paying attention to the nature of the data.

The general idea behind embedding-based and meta-learning-based methods is to pre-train a DNN on a big generic dataset before using for the few-shot problem. Figure 28 illustrates this idea. In Fig. 28a, a DNN called ***backbone*** is trained on the generic dataset. This backbone can be decomposed into a feature extractor (i.e. first layers of the DNN) followed by a simple classifier (e.g. LR). The feature extractor is trained to extract features general enough, so that many classes can be classified by the simple classifier. As shown in Fig. 28b, the backbone is reused for the few-shot problems. A new classifier is used on top of the representations of the inputs extracted from the feature extractor with or without fine-tuning (i.e. a few epochs of additional training). Note that the generic dataset must contain data similar to that of the few-shot problem but not necessarily from the same classes.

Many few-shot learning methods have been designed from the general framework illustrated in Fig. 28. Their main difference often lies in the way the backbone is trained on the generic dataset. Below, we describe more precisely the three families and we also mention the names of the methods used in the experiments presented in this manuscript.

Meta-learning-based methods

The idea in meta-learning [FAL17; AES19] is to learn a good initialization of backbone parameters, so that it can easily adapt to a new few-shot problem with only a few epochs of training.
Method mentioned in this manuscript: MAML++ [AES19].

- ✓ Train to be efficient on few-shot problems.
- ✗ Hard to implement and to train.

Learning method	Architecture	Mini-ImageNet	
		1-shot	5-shot
SimpleShot [Wan+19]	ResNet-18	63.10 ± 0.20	79.92 ± 0.14
S2M2$_R$ [Man+20]	ResNet-18	64.06 ± 0.18	80.58 ± 0.12
MAML [FAL17]	ResNet-18	49.61 ± 0.92	65.72 ± 0.77
Transfer+Graph Interpolation [HGP21]	ResNet-18	72.40 ± 0.24	82.89 ± 0.14
SimpleShot [Wan+19]	WideresNet	65.87 ± 0.20	82.09 ± 0.14
S2M2$_R$ [Man+20]	WideresNet	64.93 ± 0.18	83.18 ± 0.11
MAML [FAL17]	WideresNet	$\times$	$\times$
Transfer+Graph Interpolation [HGP21]	WideresNet	76.50 ± 0.23	85.23 ± 0.13

Figure 29 – Accuracy of various few-shot learning methods averaged on 10000 5-way 15-query problems. The 95% confidence intervals are indicated. SimpleShot, S2M2$_R$ and MAML are inductive methods. Transfer+Graph Interpolation is transductive: it has access to the query samples without their labels during training. Transfer+Graph Interpolation follows the methodology of S2M2$_R$ to train the backbone. These results come from the paper [HGP21]. The results for MAML with a WideResNet are missing.

Embedding-based methods

Embedding-based methods [Wan+19; Man+20] are inspired from transfer learning. They aim at learning general-purpose data representations, general enough to be relevant for new problems. Often, the backbone is trained to classify all classes from the generic dataset at once. To evaluate the performance on a new few-shot problem, the features of the new examples are retrieved from the trained backbone and then, a simple classifier (e.g. LR, NCM) is used on top of the extracted features.
Methods mentioned in this manuscript: SimpleShot [Wan+19], S2M2$_R$ [Man+20] (pre-trained with self-supervision and manifold mixup).

✓ Possible to pre-train the backbone with state-of-the-art techniques such as self-supervision [GSK18; Bey+19; Dos+14] and manifold mixup [Ver+19].

Hallucination-based methods

Hallucination-based methods [ZZK19; Che+19; Wan+20] try to artificially generate more training examples linked to the specific few-shot problem. If enough examples representative of the classes diversity were generated, it would not be necessary to pre-train the backbone on a generic dataset. Some generative methods still require to be trained. For instance, the one composing the foreground objects of the few-shot images with the backgrounds of other images [ZZK19]. Other methods generate new images by modifying existing images with basic transformations such as rotations, cropping, zoom, brightness shift...

✓ Increase the number of training examples.

✓ Can be combined with embedding-based methods and meta-learning.

✗ May also require a generic dataset to train a generative model.

✗ Hard to capture the diversity of a class from a few examples.

We described several methods to pre-train the backbone from which useful representations are extracted from the new inputs of a particular few-shot problem. In most cases, a simple classifier is then trained on the representations to reduce the risk of overfitting. As shown in Table 29, methods based on embedding learning [Man+20]

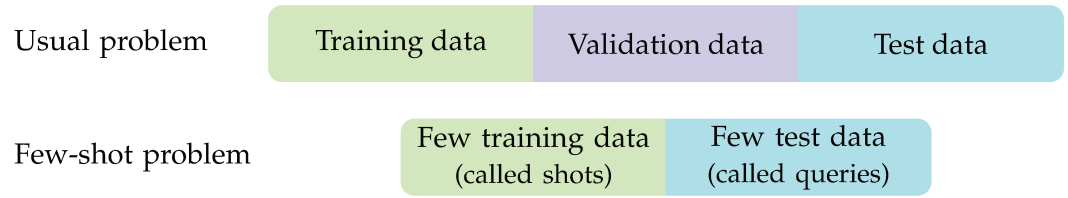


Figure 30 – Illustration of the two difficulties encountered in few-shot learning: training on few examples, evaluating on few examples.

are more efficient than meta-learning-based methods. Now, the question is: how can we control that the model is not overfitting? Traditional deep learning datasets contain many test examples. Here, in a few-shot problem, there is few training examples and so a few test examples at best. The following subsection is devoted to how few-shot learning methods are evaluated in the literature.

2.1.3 Evaluation of few-shot learning methods

In deep learning, few-shot learning methods are evaluated on particular datasets. The classes are split into 3 sets: a base set, a validation set and a novel set. The ***base classes*** are used to pre-train DNNs. The ***validation classes*** are used to select the trained DNN that enables to obtain the best representations for solving new classification problems (e.g. among various choices of architecture, by using early stopping...). The ***novel classes*** are used to generate multiple N-way K-shot Q-query few-shot problems on which the few-shot methods are finally evaluated.

For instance, mini-ImageNet is a common dataset in few-shot learning described in subsection 1.1.3 on page 44. It contains multiple images from 100 classes. 80 classes have been gathered in a generic dataset. Among them, 64 are base classes and 16 are validation classes. The last 20 classes are novel classes.

Usually in deep learning, when a DNN is trained to recognize some classes, its ability to generalize to new examples is evaluated on validation examples not used during training. In few-shot learning, the difficulty is two-fold. There are already not enough examples to train with. It is therefore difficult (if not impossible in 1-shot) to leave out a part of the training examples for validation.

As mentioned earlier and in Fig. 30, the issue is circumvented in academic research. Indeed, the datasets used in academic research contain a small training set and a small test set. The classifiers are trained on the few available shots. They are evaluated on the few test examples. However, in practice, we do not have access to these additional labeled examples. How in this case, can we succeed in evaluating the generalization ability of a trained model?

One of our major contribution is to put forward the question of the evaluation of the generalization ability of a classifier on a particular few-shot problem without test examples. In this chapter, we detailed the notion of few-shot problem and we described some methods used to solve them. We saw how these methods are evaluated in the literature. Now, we propose to explore alternative metrics to evaluate the generalization ability of a few-shot classifier.

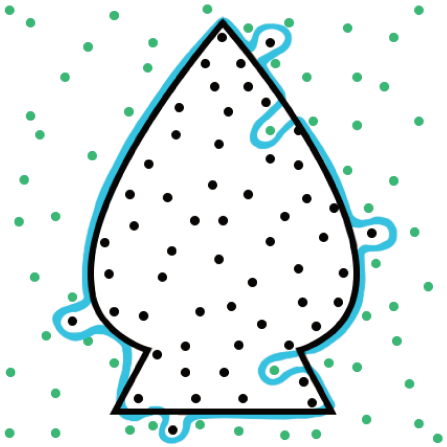


Figure 31 – Illustration of the notion of overfitting. Two classes (black and green) are represented by noisy points. The real boundary is shown by the black line. An overfitting classifier has learned the blue decision boundary which will introduce many classification errors.

2.2 Evaluating the Generalizability of a Few-Shot Classifier

The notion of generalization is crucial in machine learning. Learning is to be able to generalize: an algorithm learning to solve a problem on training examples is expected to perform well on new examples.

Until recently, researchers thought that an algorithm containing more parameters than training examples would probably overfit (i.e. make many errors on new examples). It would learn a solution classifying perfectly the training examples and as such, it would pay too much attention to irrelevant details to classify a new example. Such phenomenon is illustrated in Fig. 31.

However, the last few years have shown that most successful deep learning models contain far more parameters than training examples. We still do not know why they generalize correctly. Several theories exist (architecture of the models, implicit regularization…) but none of them is able to predict the generalization ability of models [Zha+21a].

Definition 28

Generalization is the ability to perform well on new data. The *generalization gap* is the difference between the error on the test set and the error on the training set. Note that the generalization gap is generally called variance in machine learning.

In general, the generalization of a DNN is evaluated on examples not used during training. As shown previously in Fig.30, some data is set aside (test set). For instance, CIFAR-10 contains 60000 images among which 10000 are kept for testing.

On problems with large datasets, this way of evaluating generalization already has some limitations [Jia+20b]. Indeed, we do not completely understand why and how a model generalizes [Zha+21a]. Understanding would lead to many advances. We could have guaranties on the generalization gap of a model for critical applications (medical field, self-driving cars…). We could also use this knowledge to design models generalizing better. Furthermore, the vulnerability of DNNs to adversarial attacks shows that the current evaluation of generalization with a set of new examples is not ideal.

For example, in the case of image classification, some adversarial attacks seek to imperceptibly change the pixels of certain images to completely distort the decisions of the DNNs.

On few-shot problems, evaluating the generalization on a subset of the dataset is more problematic [BBG21]. First, this reduces the already small amount of examples available for training. Then, even setting aside some of them, they are very unlikely to be representative of the diversity of classes in the real world. Yet, the notion of generalizability of a model in few-shot learning is even more critical than in deep learning: the backbone pre-trained on generic base classes, may not be relevant at all for the few-shot problem at hand. In particular, the generalization gap of a few-shot model is influenced by two factors:

- the variability between training examples and the real class distribution (as in traditional deep learning);
- the relationships between the classes of the few-shot problem and the base classes that have been used to pre-train the backbone (more information in the box on page 75).

Due to the scarcity of data in few-shot learning, the fundamental question we emphasize in this section is: *how to evaluate the generalization ability of a model without a validation or a test set?* This open question is really important for the entire deep learning community, and not just for the few-shot learning one. Highlighting the importance of this question in the few-shot case is one of our major contributions.

A lot of contributions have been made to try to define measures explaining generalization without a validation or a test set. Recently, several challenges on this theme have been proposed at major conferences (e.g. Predicting Generalization in Deep Learning at NeurIPS 2020).

Throughout this section, we first present some measures, called complexity measures, which have been proposed in deep learning to evaluate the generalizability of a model without using new examples. We also show how the relevance of these measures is empirically evaluated. Then, we focus on few-shot problems: we evaluate some measures on few-shot benchmark models and discuss the limitations of the current approaches.

2.2.1 Complexity measures

Many works propose to estimate the generalizability of a DNN on classification problems without using a validation or a test set. Generally, they define complexity measures theoretically or experimentally linked to the notion of generalization [Jia+20b].

Definition 29

A ***complexity measure*** is a measure linked to the notion of generalization. It is computed from the trained model, its optimization parameter and/or training data. It does not have access to new data samples.

Influence of Base Classes on Generalization in Few-Shot Learning

A few experiments on mini-ImageNet illustrate the influence of base classes on the generalization of a few-shot classifier. A backbone[1] is pre-trained on the base classes of mini-ImageNet. Few-shot problems are randomly generated from the novel classes of mini-ImageNet. For each problem, a LR is trained on top of the representations of the new examples extracted from the backbone.

The generalization ability of a classifier is influenced by the novel classes. Indeed, the accuracies on the few-shot problems vary from 54% to 96.4%[2]. The average standard deviation due to the choice of shots within each class is 3.36% whereas the average standard deviation due to the choice of classes is 5.99%[3].

The impact of novel classes depend on their relationships with the base classes. The largest overlaps[3] between the novel classes of mini-ImageNet and the largest overlaps between its base and novel classes are represented in the graphs above. The average overlap between the 5 classes of a few-shot problem is highly correlated with the generalization error of the LR as shown in the scatter plot above (Pearson correlation coefficient of 0.78). Three phenomena appear.

- *One-to-one: a novel class matches uniquely a base class.* E.g. ant or bookshop. It seems ideal as these classes do not strongly overlap other novel classes.
- *Many-to-one: several novel classes match the same base class.* E.g. golden retriever and malamute. These novel classes highly overlap with each other.
- *One-to-many: a novel class matches many base classes.* E.g. electric guitar. It seems that such a class is more likely to overlap with another class.

1. DenseNet architecture trained with SimpleShot (embedding-based method).
2. As training errors are close to 0, test accuracies represent the generalization gaps.
3. The detailed algorithms are given in Appendix B.

Many complexity measures have been proposed in deep learning [Jia+20b] and, although it it more recent, in few-shot learning [BBG21].

Some measures are theoretical: measures defining a bound on the probability that the error is smaller than a certain value (such as PAC-Bayesian bounds [McA99; DR17] or other bounds based on VC-dimension [VC15; Bar+19]). Other measures have been experimentally motivated: measures linked to the parameters of the model (such as number of parameters or Frobenius norm of the parameters), measures linked to the robustness of the training loss with respect to a perturbation of the parameters (such as sharpness-based measures [Kes+17]), measures looking at the distribution of the representations at the output of the model (such as cross-entropy, entropy, margin [Jia+19] or relative position of the examples in the output space [Las+20a])...

As theoretical measures are often not applicable to real world problems (either they are defined on very controlled models or their bounds are not tight enough), much work remains to be done to find and evaluate new measures of complexity. In the next subsection, we show that it is not straight-forward to empirically evaluate the relevance of a complexity measure to predict the generalizabity of a model.

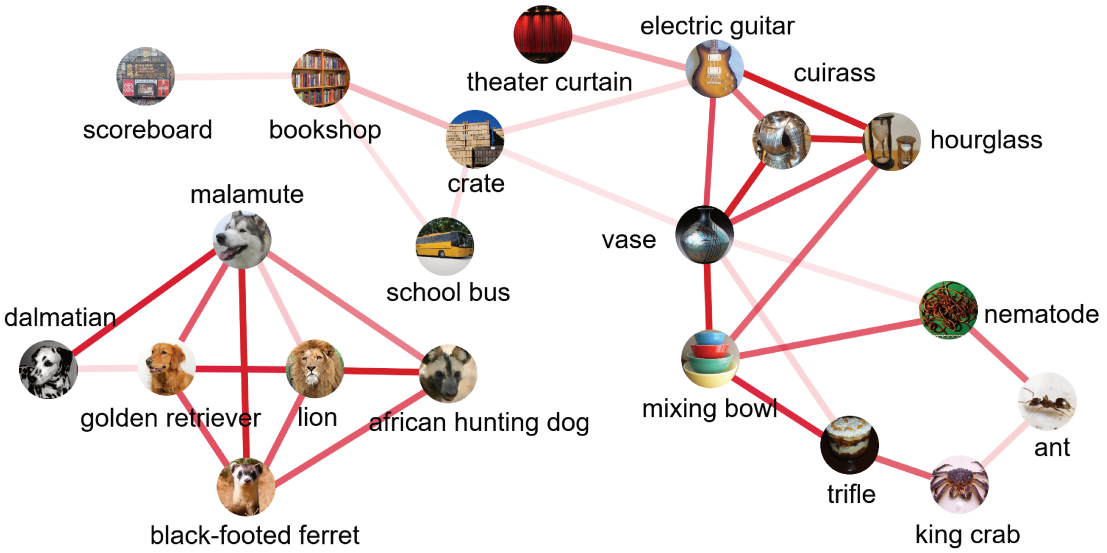


Figure 32 – Overlap between novel classes of mini-ImageNet. A darker edge means a larger overlap.

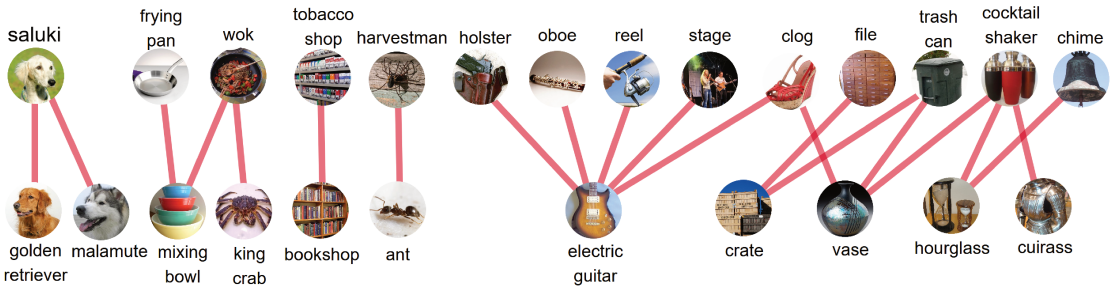


Figure 33 – Overlap between base and novel classes of mini-ImageNet. Top row: base classes. Lower row: novel classes. For clarity, only a subset of classes is plotted.

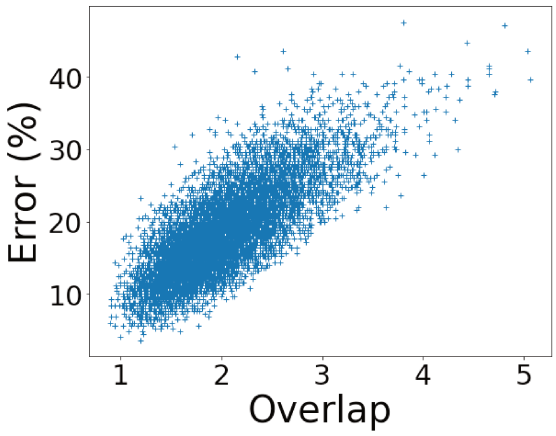


Figure 34 – Generalization error as a function of class overlap in few-shot problems.

2.2.2 Evaluating the relevance of complexity measures

A simple way to empirically evaluate the relevance of a complexity measure is to look at its correlation with the generalization gap of several models on test sets [Jia+20b].

Looking at correlation coefficients such as the Pearson correlation coefficient has the advantage of being easy to implement on standard deep learning and few-shot learning benchmark datasets. Yet, it has some disadvantages.

Sometimes, the Pearson correlation coefficient does not reflect the real relationships between two variables. It can miss a non-linear relationship or be corrupted by outliers as shown in Chapter 1.3, Fig. 23. This can sometimes be handled by using the Kendall's rank correlation coefficient [Ken38] or by plotting the relationships between the two variables.

Correlation coefficients may also reflect spurious relationships between models and complexity measures. For instance, consider the case where the correlation is computed by varying a hyperparameter of the model. If the complexity measure is relevant, when the variation of the hyperparameter induces a decrease in the generalization gap, the decrease in the generalization gap also induces a decrease in the complexity measure. However, it is possible that varying the hyperparameter induces a decrease in both the generalization gap and the complexity measure without any link between the generalization and the complexity measure.

In a recent study [Jia+20b], most of the previously mentioned complexity measures have been empirically evaluated and compared. To counter-act spurious correlations linked to hyperparameters, they chose to vary several hyperparameters of the models (batch size, dropout probability, learning rate, network depth, weight decay coefficient, network width, optimizer) and to compute an average correlation coefficient in two steps.

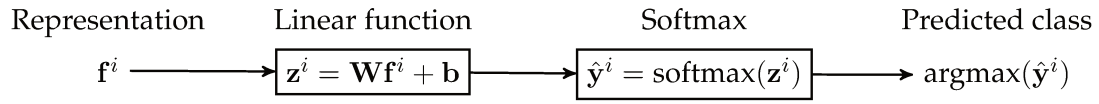
1. An averaged Kendall's rank correlation coefficient representative of the influence of one hyperparameter is computed.
   - $\rightarrow$ The Kendall's rank correlation coefficient is obtained by varying only this hyperparameter and fixing the values of the others.
   - $\rightarrow$ All coefficients obtained after varying the values of the other hyperparameters are averaged.
2. All averaged Kendall's rank correlation coefficients obtained after varying the studied hyperparameter are averaged.

This solution is a first step towards a better evaluation of complexity measures but, unfortunately, it requires a lot of computational power to train many models (more than 10000 in their study).

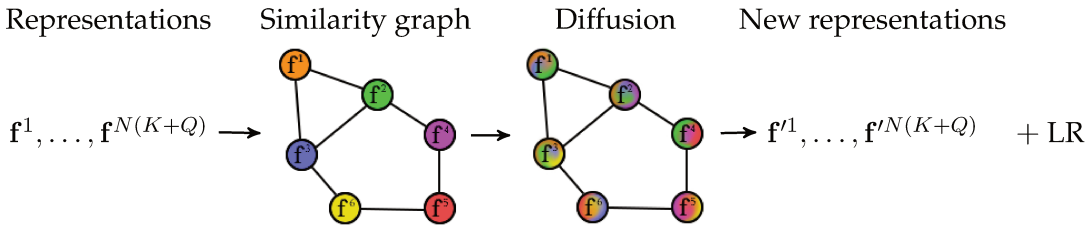
In these first two subsections devoted to the notion of generalization in deep learning, we have presented several directions around which current research is developed. The following subsection focuses on the same generalization problem in the particular context of few-shot learning.

2.2.3 Complexity measures in few-shot learning

We conducted a study [BBG21] in which we evaluated the ability of several complexity measures to predict the generalizability of a few-shot classifier without using a validation or a test set. The setup for few-shot learning is slightly different from traditional



(a) Inductive setting. A LR is trained on the representations of the K shots per class.



(b) Transductive setting. The representations of the examples (shots + queries) are smoothed along a similarity graph before being used in the LR.

Figure 35 – Description of the classifiers used to solve few-shot problems.

deep learning: the studied classifier is trained on top of the representations of a few training examples extracted from a backbone pre-trained on a generic dataset.

To limit the context of the study, we made a series of assumptions about the setup of the few-shot learning problems (few-shot or few-label, inductive or transductive), the generic dataset (accessible or not to compute a complexity measure) and the few-shot learning method (backbone used with or without fine-tuning on the new examples).

In this study, only two settings are considered: inductive few-shot learning and transductive few-shot learning. Thus, studied few-shot problems contain N classes, K shots per class and Q queries per class. The Q queries examples are available without their labels in the transductive setting. Moreover, only few-shot methods whose backbone is not fine-tuned are considered. Two arguments support this choice. First, such methods can achieve state-of-the-art results on benchmark datasets [Man+20]. The second reason is computational. Without fine-tuning, the same DNN can be reused without additional training in all experiments. Thus, the time and the memory required to evaluate the complexity measures on numerous few-shot problems are reduced. Finally, we do not use the generic dataset in the complexity measures as it may not be available in practice.

In the inductive setting, the classifier chosen to be trained on top of the representations of the examples of a few-shot problem is a LR. In the transductive setting, it is also a LR but the representations are smoothed according to the Transfer+Graph Interpolation method [HGP21]. Figure 35 contains a description of the classifiers, where $\mathbf{f}^i$ is the representation of the example i extracted from any pre-trained backbone. The LR is detailed more precisely in Chapter 1.1.4 and the equations of the diffused LR are given in Appendix A.1. In both cases, the LR is trained on 50 epochs on the labeled training examples.

In the following, several complexity measures are empirically evaluated and compared.

Measure \ Based on	Shots	Shot labels	Queries
Cross-entropy	✓	✓	
Similarity	✓	✓	
Graph	✓		Optional
Clustering	✓		Optional
LR confidence			✓

Table 1 – Complexity measures evaluated on few-shot problems.

Complexity measures

Five complexity measures are tested: the cross-entropy, a similarity-based measure, a graph-based measure, a clustering-based measure and the LR confidence. All these measures are computed from the representations of the data samples. They do not directly use the parameters of the backbone because these parameters are learned independently of the few-shot problem. The measures are summarized in Table 1 and detailed below.

Cross-entropy The cross-entropy is computed from the shots and their labels to see how well the LR output matches the real classes. The cross-entropy is also the loss minimized by the LR during training. As the LR is trained on a fixed number of epochs, it is believed that the harder the few-shot problem is, the greater the loss will be at the end of the training.

Given $\mathbf{y}_c^i$ the number (0 or 1) indicating whether the label of the example i is c and $\hat{\mathbf{y}}_c^i$ the number indicating the probability of the example i to be labeled c by the LR. The cross-entropy is defined as:

$$\text{cross-entropy} = \frac{-1}{NK} \sum_{i=1}^{NK} \sum_{c=1}^{N} \mathbf{y}_c^i \log \hat{\mathbf{y}}_c^i \,. \tag{21}$$

Similarity-based measure The similarity measure is also computed from the shots and their labels. It compares the intra-class similarities with the inter-class similarities.

The similarity within a class c and the similarity between two classes c and c' are respectively measured by:

$$\text{intra}(c) = \frac{1}{K(K-1)} \sum_{\substack{i,j \neq i \\ \mathbf{y}_c^i = 1 \\ \mathbf{y}_c^j = 1}} \cos(\mathbf{f}_i, \mathbf{f}_j) \text{ and } \text{inter}(c, c') = \frac{1}{K^2} \sum_{\substack{i,j \\ \mathbf{y}_c^i = 1 \\ \mathbf{y}_{c'}^j = 1}} \cos(\mathbf{f}_i, \mathbf{f}_j) \,. \tag{22}$$

The proposed similarity measure is:

$$\text{similarity} = \frac{1}{N} \sum_{c=1}^{N} \left(\text{intra}(c) - \max_{c \neq c'} (\text{inter}(c, c')) \right) . \tag{23}$$

Graph-based measure The graph-based measure is only computed from the examples (shots and, if available, queries). It does not use the labels. It indirectly measures the difficulty to separate the representations of the examples into N groups (N being the number of classes).

Consider a graph where each vertex is an example and where edges link the examples having the most similar representations. When the representations of distinct class examples are very dissimilar, this graph is expected to produce at least as many connected components as the number of classes. The N-th lower eigenvalue of the combinatorial Laplacian of a graph containing at least N connected components is zero [Shu+13]. Consequently, we chose to define the graph-based complexity measure as the amplitude of this N-th lower eigenvalue.

Clustering-based measure The clustering-based measure is also computed only from the examples. It measures the relative similarity between the N clusters obtained after a N-means algorithm.

Assuming that the examples within classes are sufficiently similar, we expect each learned cluster to represent a class. A measure of relative similarity within and between clusters, such as the Davies-Bouldin (DB) score [DB79], would provide insight into the difficulty of clustering. This measure could be relevant to estimate the ease with which a classifier will be able to generalize to new samples.

Denote the centroid of a cluster C $\boldsymbol{\mu}_C$, such that $\boldsymbol{\mu}_C = \frac{1}{|C|}\sum_{i\in C}\mathbf{f}_i$. The average distance between the samples in C and the centroid of their cluster $\boldsymbol{\mu}_C$ is:

$$\delta_C = \frac{1}{|C|}\sum_{i\in C}\|\mathbf{f}_i - \boldsymbol{\mu}_C\|_2 \,. \tag{24}$$

The DB score is defined as:

$$\text{clustering} = \frac{1}{N}\sum_{C=1}^{N}\max_{\tilde{C}\neq C}\left(\frac{\delta_C + \delta_{\tilde{C}}}{\|\boldsymbol{\mu}_C - \boldsymbol{\mu}_{\tilde{C}}\|_2}\right). \tag{25}$$

Confidence-based measure The last measure computes the confidence of the LR decision on the queries. It can only be used in the transductive setting, in which the queries are available without their labels during training.

The LR confidence on a query sample is based on the distance between the LR output and a one-hot-bit encoded version of this output. In more details, for each query, the LR outputs the probability it has to belong to a particular class. As we do not know the label of the query, we cannot look at the probability the classifier gives to this label. However, we propose to look at the maximal probability the LR attributes to one of the classes. If the maximal probability is far from one, we can assume that the LR is unsure of its output. The lower the maximal probability is, the riskier the decision must be for the query sample.

$$\text{LR confidence} = \frac{-1}{NQ}\sum_{i=1}^{NQ}\log\max_{c}(\hat{\mathbf{y}}_c^i)\,. \tag{26}$$

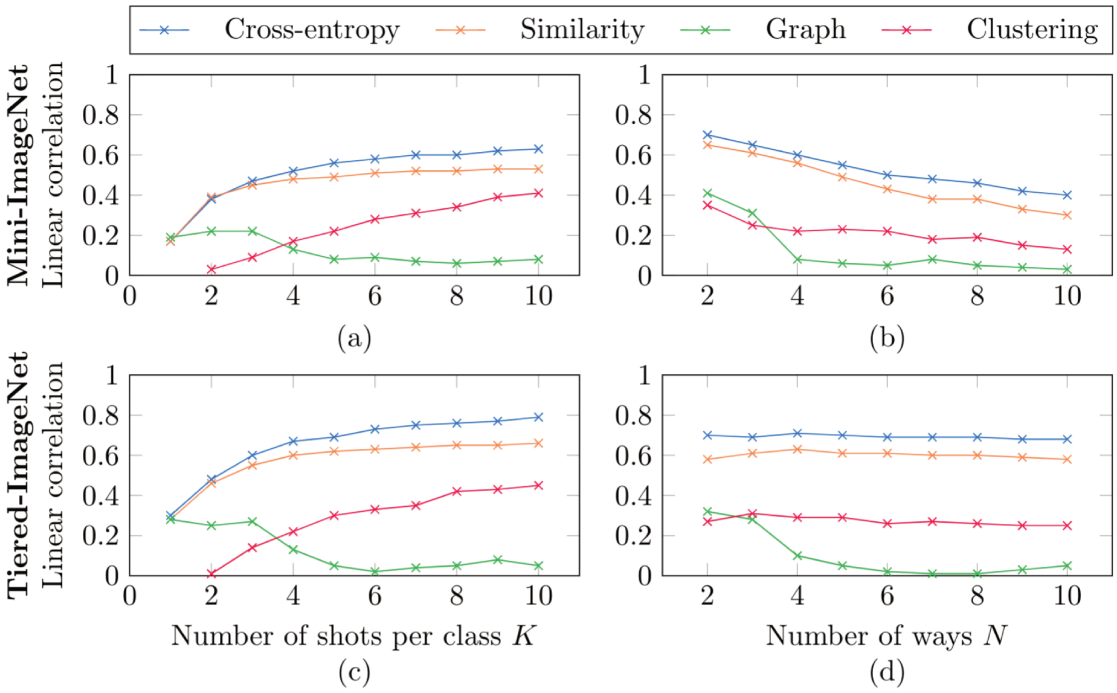


Figure 36 – Inductive setting. Each curve represents the linear correlation of a complexity measure with the accuracy of a LR on few-shot problems. For each point, the correlation is computed over 10000 few-shot problems randomly generated from mini-ImageNet (a,b) or tiered-ImageNet (c,d). The few-shot problems are either 5-way 50-query problems with a variable number of shots (a,c) or 5-shot 50-query problems with a variable number of classes (b,d).

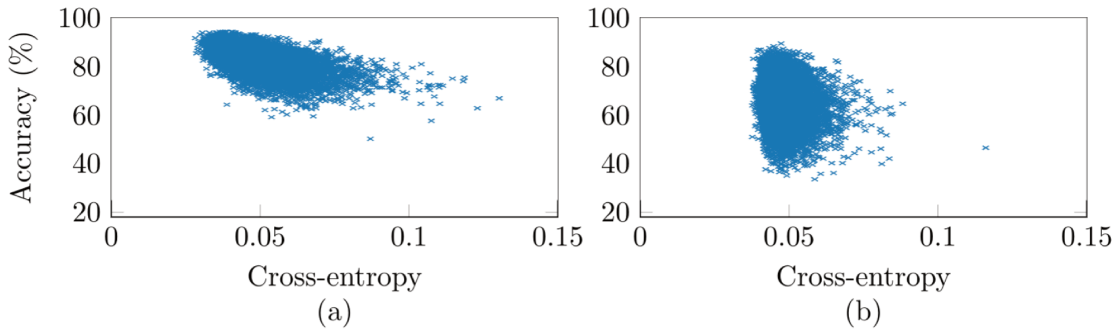


Figure 37 – Inductive setting. The accuracy of a LR on the queries is plotted as a function of the cross-entropy loss of the LR on the shots. Each point is a 5-way 50-query few-shot problem generated from mini-ImageNet with either 5 shots (a) or 1 shot (b).

Results

In the literature, few-shot learning methods are evaluated on the few queries of a large number of few-shot problems randomly generated from a set of classes, called novel classes. Similarly, to evaluate the relevance of a complexity measure, we generate many few-shot problems from the set of novel classes. We first look at the correlation between the value of the complexity measure on each few-shot problem and the accuracy of a LR trained on the same problem. The correlation measure used here is the Pearson correlation coefficient as the relationships are rather linear (see scatter plots in Fig. 37 and Fig. 38). The accuracy of a LR on a few-shot problem is computed on the query samples. Several correlation coefficients are reported by varying the datasets/architecture of the backbones/few-shot learning methods (either a *WideResNet trained on the base classes of mini-ImageNet with S2M2$_R$* and evaluated on the novel classes of mini-ImageNet or a *DenseNet trained on the base classes of tiered-ImageNet with SimpleShot* and evaluated on the novel classes of tiered-ImageNet) and the few-shot learning setting (number of ways, number of shots and number of queries). Note that mini-ImageNet and tiered-ImageNet are detailed in chapter 1.1.3 and that S2M2$_R$ and SimpleShot are detailed in Appendix A.

In the inductive setting, Fig. 36 shows the linear correlation of several complexity measures with the accuracy of a LR on several few-shot problems. Each point is obtained by computing a Pearson correlation coefficient over 10000 randomly generated few-shot problems. In all settings, the cross-entropy measure is the most correlated with the test accuracy.

In the transductive setting, Fig. 38 shows that for all setting, the LR confidence is the most correlated measure. The cross-entropy and the similarity measures are less correlated but this is not really surprising: they do not use any information from the query samples whereas the other measures do.

As several measures are correlated with the generalization gap on few-shot problems, the next question is: can we predict the generalization gap on a new few-shot problem by only looking at the value of one of these complexity measures? In our article [BBG21], we provide insight into the ability of complexity measures to predict the performance of a classifier on new data. We arbitrarily split few-shot problems into two categories: problems with an accuracy below 80% are hard and the ones above 80% are easy. Thus the question is simplified in: do complexity measures enable to distinguish between hard and easy tasks?

A first experiment is made to choose a threshold value for a given complexity measure in order to distinguish hard and easy problems. A ROC curve is computed from 10000 5-way 5-shot 30-query few-shot problems generated from a subset of 10 classes among the 20 novel classes of mini-ImageNet. To obtain the blue points, the threshold value varies along sensible values of the complexity measure. Here, ***sensitivity*** is the proportion of problems classified as being hard among the hard problems. 1 - ***specificity*** is the proportion of problems classified as being hard among the easy problems. A threshold value is chosen from the ROC curve with a given sensitivity/1 - specificity couple.

A second experiment is made to see if this threshold value is relevant for new few-shot problems. A confusion matrix is computed on 10000 5-way 5-shot 30-query few-shot problems generated from the remaining 10 novel classes of mini-ImageNet. Each row of the confusion matrix is normalized, so that its first (resp. second) row indicates

the percentage of tasks predicted hard or easy among the hard (resp. easy) tasks. Given the chosen threshold, the sensitivity (upper left cell) and specificity (bottom right cell) are computed on these new problems. The measure could not be used as a reliable predictor if the new sensitivity/specificity values differ too much from previous values.

In our study, two complexity measures are evaluated: cross-entropy in the inductive setting (a) and LR confidence in the transductive setting (b).

The threshold on cross-entropy is chosen with a sensitivity of 0.81 and a 1 - specificity of 0.29. On new problems, the sensitivity and specificity become respectively 0.45 and 0.14. As the values are really different, cross-entropy cannot be considered as a reliable predictor.

With LR confidence, the chosen sensitivity is 0.81 and 1 - specificity is 0.16, and they respectively become 0.76 and 0.18 on the new problems. LR confidence is thus more promising.

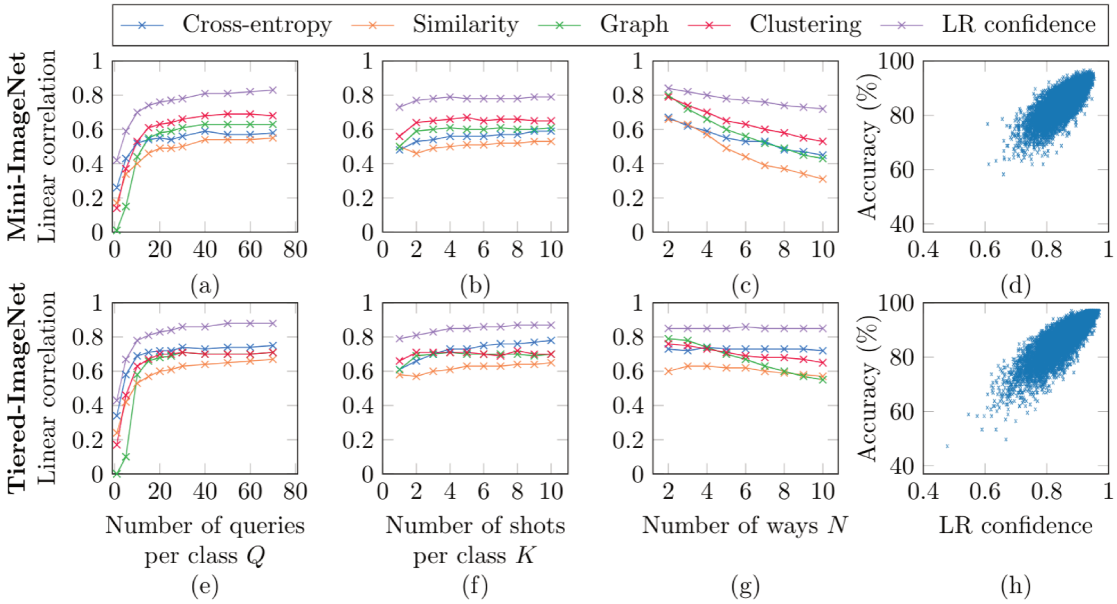
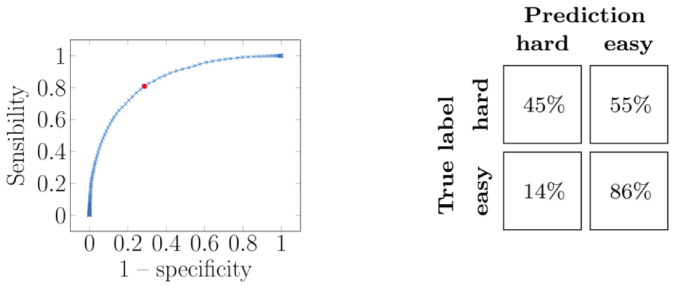
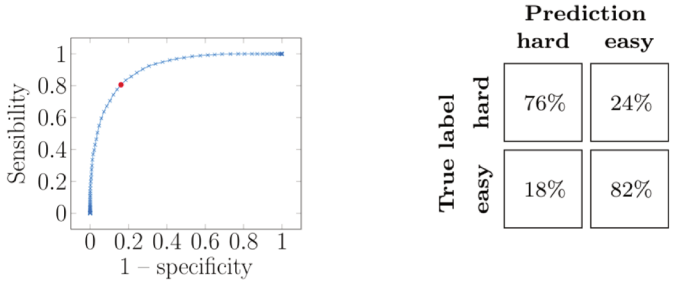


Figure 38 – Transductive setting. Each cross is a Pearson correlation coefficient between the values of a complexity measure and the accuracies of a LR, computed on 10000 few-shot problems generated from mini-ImageNet (a,b,c,d) or tiered-ImageNet (e,f,g,h). By default, 5-way 5-shot 30-query problems are generated. The numbers of queries, shots, classes vary respectively in (a,e), (b,f) and (c,g). In (d,h), the accuracy of the LR is plotted as a function of the LR confidence. Each point corresponds to a few-shot problem.



(a) Inductive setting with the cross-entropy complexity measure.



(b) Transductive setting with the LR confidence complexity measure.

Figure 39 – Prediction of the difficulty of a few-shot problem (hard or easy) from a threshold value on a complexity measure. A threshold is chosen from a ROC curve plotted by varying the threshold of the complexity measure on many few-shot problems generated from a subset of novel classes of mini-ImageNet. Sensitivity is the proportion of problems classified as being hard among the hard problems. 1 - specificity is the proportion of problems classified as being hard among the easy problems. Then, a confusion matrix is computed from the chosen threshold value (red points) on few-shot problems generated from the remaining novel classes of mini-ImageNet.

Summary

This chapter brings together two very important issues in machine learning: learning from few examples and estimating the generalization ability of an algorithm.

In few-shot learning, proposed learning methods are inspired from several domains (meta-learning, transfer, self-supervised learning...). In this manuscript, we do not propose a new solution but we adapt some of them in the next chapter dedicated to brain decoding.

In machine learning in general, algorithms learn to solve a problem from a training dataset. Understanding why they generalize and estimating their accuracy is crucial before they can be developed in many technologies. Currently, the generalization of an algorithm is evaluated on a validation set (and sometimes also on a test set) not used for training. Nevertheless, it would be interesting to be able to evaluate it otherwise. Many works define theoretical or empirical complexity measures for DNNs although it is not straight-forward to empirically evaluate the causal relationship between a complexity measure and generalization. As no measure is yet satisfactory, many challenges are organized on this topic in deep learning conferences. For example, at NeurIPS, we participated in the competition called Predicting Generalization in Deep Learning [Jia+20a]. We proposed a complexity measure looking at whether the same classes were assigned to nearby training examples in the penultimate layer of DNNs. This measure, ranked third in the competition, is detailed here:

Carlos Lassance et al. "Ranking Deep Learning Generalization using Label Variation in Latent Geometry Graphs". In: *arXiv preprint arXiv:2011.12737* (2020)

Finally, these two issues have been brought together in the same chapter for a specific purpose. We wanted to show that finding alternative solutions to measure the generalization of a model is even more crucial in few-shot learning [1]. When the access to a training dataset is complicated, the access to a test dataset is also complicated. To highlight this problem, we published an article containing the measures that are presented here in this chapter and detailing the framework in which they are compared:

Myriam Bontonou, Louis Béthune, and Vincent Gripon. "Predicting the Generalization Ability of a Few-Shot Classifier". In: *Information* 12.1 (2021), p. 29

The next chapter is devoted to a practical application: brain signals decoding. We will see how to classify complex signals from a few examples.

1. Codes to generate the plots in the box on page 76 and the results of the article are available at `https://github.com/mbonto/fewshot_generalization`.

Chapter 3

Few-Shot Learning for Decoding Brain Activity

Contents

THIS chapter is entirely dedicated to the decoding of brain activity. Before getting into the heart of the matter, we start by detailing how to measure brain activity, by defining the notion of decoding and especially by motivating the interest of this application for the machine learning community.

The brain is a complex biological organ, which is the center of the central nervous system of many living species. The brain receives signals from the outside world through millions of sensors located in various places on the body, and processes them to adapt the behavior of the organism. A human brain is relatively small, however its capacities of reaction, anticipation and adaptation are still unmatched by today's computers. This gap is for instance visible in a very active area of AI research called reinforcement learning. In reinforcement learning [Hen+18], a machine learned to perform tasks by analyzing external signals rewarding or penalizing its decisions. Unlike humans which are in general able to adapt quickly, current machines are trained on very specific tasks in restricted environments.

As a biological organ, the brain processes information differently than traditional computers. Current computers contain units dedicated to distinct functions (CPU, hard disk...). In contrast, cognitive functions are massively distributed in the brain [CD16]. Unlike a computer, the memorization and the processing of signals are carried out simultaneously by the same cells: the neurons [Kan+00].

The neuron is the fundamental computing unit of the brain. The human brain is composed of about 86 billion neurons and about 100 trillion connections between these neurons. There are different types of neurons, but their behavior can be summarized as follows. Each neuron receives information from hundreds of neurons, processes it and in turn transmits information to other hundreds of neurons. Learning and memory capacities are linked to brain plasticity: connections can be reinforced between certain neurons according to our needs. All along this chain of transmission, action potentials are generated inside neurons and propagate.

Brain activity can be measured non-invasively (i.e. no surgery is required to open the skull). Different brain imaging techniques measure different signals: Electroencephalography (EEG) records scalp electrical potential generated by brain activity while Magnetoencephalography (MEG) records the magnetic field generated by this same brain activity. Functional Magnetic Resonance Imaging (fMRI) measures the blood flow resulting from activity in a brain region. These neuroimaging techniques make it possible to decode brain activity.

Definition 30

Brain decoding consists in "extracting brain activity patterns that discriminate conditions and predict them from unseen neural activity" [VP19].

For example, in controlled environment experiments, participants perform different tasks (moving, reading, counting, feeling an emotion...). Their brain activity is simultaneously recorded through a machine (EEG headset, MEG scanner, fMRI scanner...) and algorithms are designed to predict the task performed from the measured activity.

The field of brain decoding is very active because some of the improvements linked to Brain-Computer Interfaces (BCIs) relies on it. BCIs are used in many technologies. From a medical point of view, paralyzed or amputated people could mentally control robotic prostheses to grasp objects [Hoc+12]. They could also steer a wheelchair [Reb+10] or surf the net with their mind [Ben+07]. In other fields, the state of alertness of people

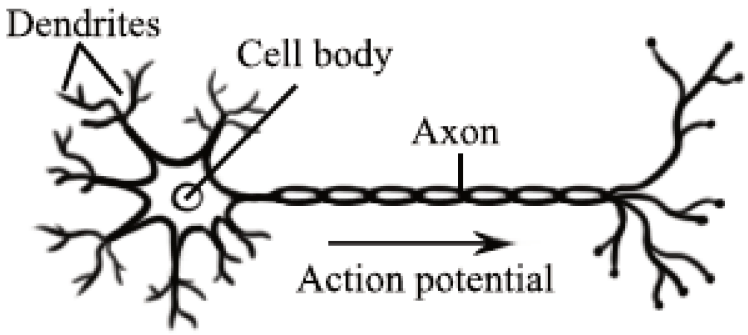


Figure 40 – Structure of a typical neuron.

driving a car could be estimated [Bla+10] or video game could be based on mentally controlled avatars [Cat21].

Decoding brain activity is challenging. Certain brain regions are more or less specialized in certain tasks but it is difficult to find a brain signature associated with a task for several reasons:

- several signals can correspond to the same task;
- several tasks can activate the same regions;
- there is significant inter-subject variability;
- the brain adapts over time, so there is also significant intra-subject variability;
- current studies do not contain a very large number of measurements per subject;
- there are complex dependencies between signals measured simultaneously in different brain regions.

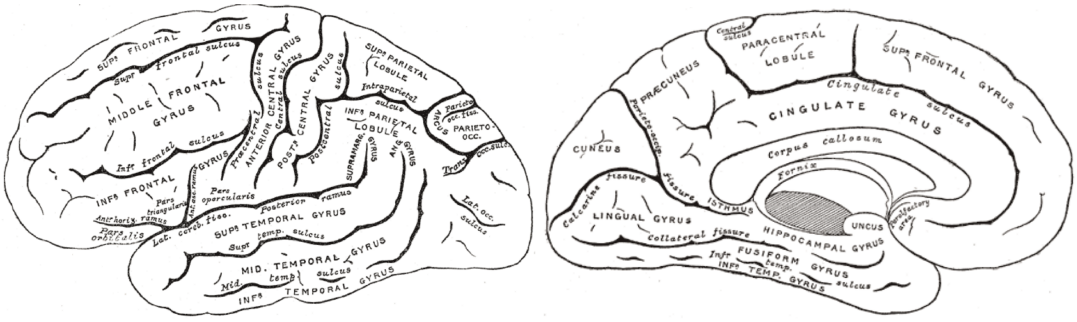
In this thesis, we are interested in two aspects that are very important for learning: the lack of data and irregularly structured data. These two critical notions appear in brain activity decoding. We have chosen to write this last chapter in three parts. The first section provides more explanation on brain activity, neuroimaging techniques and current decoding methods. The second section addresses the challenge of relatively small datasets, showing that few-shot learning can help to solve classification problems on neuroimaging datasets. The last section focuses on the irregular structure of the data. Specifically, as the network of connections between brain regions can be measured, we propose to exploit this additional information to better classify brain activity.

3.1 Introduction to Neuroimaging and Brain Activity Decoding

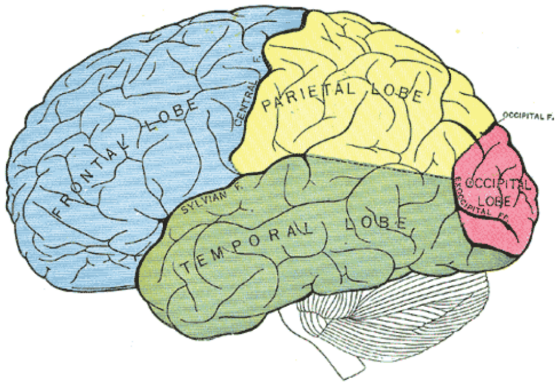
This section puts forward the complexity of human brain decoding. Starting from the basic notions of neuroscience, we present some technologies for recording brain activity and some approaches for decoding the measured activity.

3.1.1 Primer on neuroscience

This first subsection provides a general overview of how our behavior and our cognitive abilities emerge from the activity of neurons. The human brain contains about 86 billion neurons and weights around 1.5 kg [HH09]. Neurons are organized in networks, themselves involved in larger systems interacting with each other. The neuron is a particular cell able to communicate with other neurons through electrical and chemical signals.



(a) Lateral surface of left cerebral hemisphere. **(b)** Medial surface of left cerebral hemisphere.



(c) Principal fissures and lobes of the cerebrum.

Figure 41 – Anatomy of the brain. The images come from [GL18] [1].

Here is a very general way of describing the communication between neurons. A neuron contains several elements presented in Fig. 40. Dendrites (which are branch-like extensions of the neuron) contain receptors sensible to certain molecules called neurotransmitters. When these molecules are present, they interact with the receptors, which open channels across the neuron membrane. Several types of ions flow through the channels, inducing changes in the membrane potential of the neuron. Once this potential reaches a given threshold, a cascade of reactions occurs and generates a potential peak near the cell body. This potential peak, commonly called action potential or spike, is then conveyed through the axon. Finally, at the extremities of the axon, the presence of a spike leads to neurotransmitters release. They will in turn interact with the receptors of nearby neurons and the cycle will start over.

Action potentials convey information all the way from sensory nerve cells to the brain, which can then adapt behavior by sending commands to muscles. For instance, when we hear a dog bark, some receptors in our inner ear react to the frequencies of that sound and emit action potentials, which will ultimately reach the auditory cortex where the sound will be analyzed. It could also travel to other brain regions implicated in memory processes, in emotional processes such as fear or in motor movement to run away.

The organization of the brain is similar across typically developed healthy adults. A number of anatomical structures are similar across healthy adults. For example, the localization of gyri (i.e. ridge) and sulci (i.e. depression). As shown in Fig. 41, the brain is divided into 2 hemispheres (left and right), themselves split into 4 lobes: the frontal, the parietal, the temporal and the occipital. In 1909, the anatomist Brodmann delimited

1. The images are accessible on the website `https://www.bartleby.com/107/`.

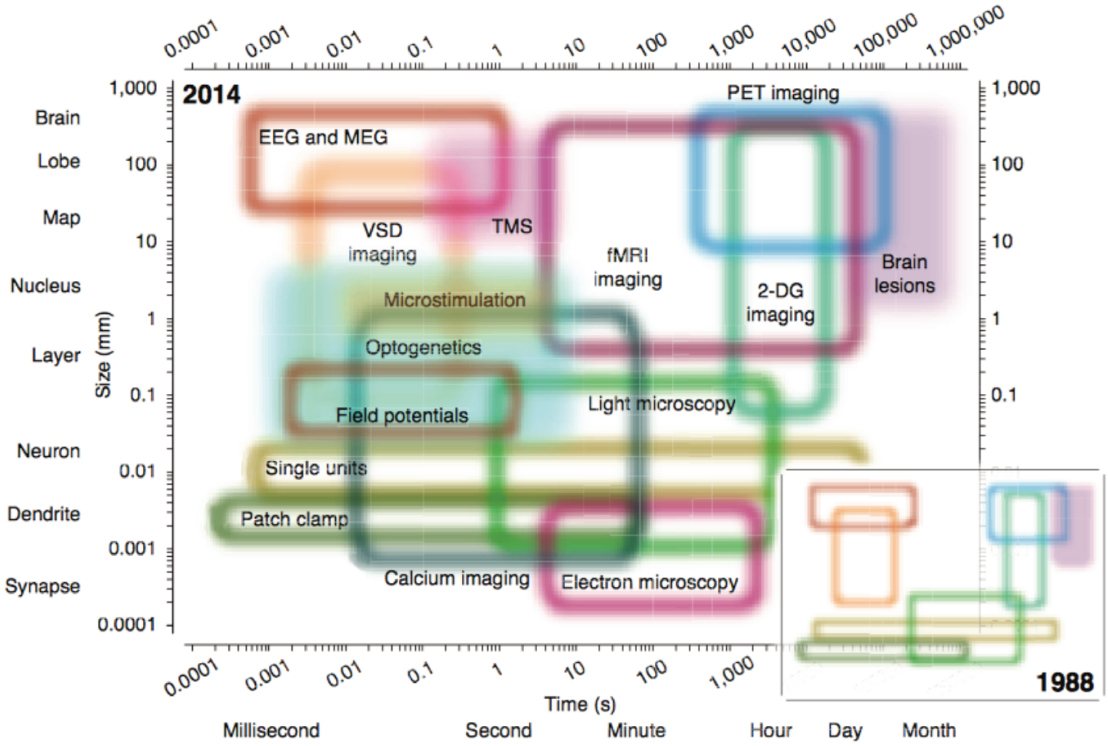


Figure 42 – Main methods available to record brain activity. Reprinted by permission from Springer Nature: Nature Neuroscience, Putting big data to good use in neuroscience, Sejnowski et al. (2014) [SCM14].

the brain into 52 smaller areas with respect to the organization and structure of neurons in these areas [Bro09].

Cognitive functions encompass the set of higher order mental functions such as speech, memory, language, attention, spatial reasoning or decision-making as well as sensory perception and motor commands. Although each cognitive function relies on many different regions across the brain, they are more or less localized in certain lobes. For instance, the primary visual cortex, which receives visual information from the eyes, is localized into the occipital lobe. The primary auditory cortex and the hippocampus, a region implied in memory processes, are localized in the temporal lobe and so on [Ken38].

These two properties (similarity between human beings and more or less localized cognitive functions) make it possible to decode the brain. However, as the same regions can be activated by multiple conditions [Pol06], decoding brain act is a very challenging task. In this chapter, we are interested in decoding cognitive functions using brain activity. We show that it is possible to decode brain activity in carefully designed experiments using statistical techniques.

3.1.2 Recording brain activity

Over the years, non-invasive technologies have been designed to record brain activity without having to surgically open the skull. Several techniques, such as EEG and MEG, measure scalp-level electrical activity related to action potentials. Others, such as fMRI, indirectly record brain activity by exploiting physiological properties linked to brain activity. These different technologies have their pros and cons, notably regarding

Figure 43 – MRI scanner [2].

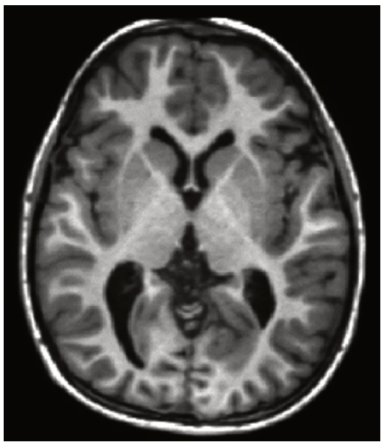


Figure 44 – T1-weighted image.

their various temporal and spatial resolutions (see Fig. 42). To decode brain activity evoked from a variety of cognitive tasks, one of the most used techniques is fMRI. Most research studies currently uses MRI scanners with a spatial resolution in the order of the millimeter (which is rather high, as a neuron measures about 100 micrometers but networks of neurons evolving jointly are in the order of the millimeter) and a temporal resolution of several seconds (which is low, as the human reaction to a stimulus is around 1 second). In this subsection, we introduce MRI acquisition techniques, as we subsequently use this type of data to attempt to decode brain activity. This will provide insight into the challenges of decoding fMRI data.

MRI is a very versatile technique [McR+17]: the same technology can image different tissues in the brain, the white matter architecture connecting different neurons and even brain activity. Fundamentally, MRI is based on two properties of the protons (of which the nuclei of the atoms are composed). Without external intervention, the nuclei spin around their axis in random directions (like a spinning-top in a zero gravity field). However, when placed in a strong magnetic field, their rotational axis tends to align with the magnetic field. This first property alone does not enable to distinguish different elements in the brain. The second crucial property is that when an electromagnetic field of a certain energy is sent to these nuclei at a certain angle, the rotation of the nuclei synchronizes and their rotation angle increases. When this electromagnetic field is removed, the nuclei begin to desynchronize and realign to the main magnetic field. This phenomenon, known as relaxation, induces a high frequency magnetic field (caused by the movement of charged particles) that can be measured. The energy required to make the protons precessing depends on the atoms and on the magnetic field intensity. In MRI, the studied atom is often the hydrogen atom. Its relaxation time depends on the tissue in which it is nested. Thus, different signals are measured for different tissues. What is interesting is that we can know the locations from which the recorded signals are emitted. First, the required energy to make the nuclei precessing also depends on the intensity of the main magnetic field. This allows the recording of brain signals slice by slice by applying an increasing magnetic field from the front to the back of the head or vice versa. Then, to localize the position of the pixels in each slice, several properties linked to the phases and the frequencies of the signals are exploited. Figure 44 shows an example of images that can be obtained from a MRI scanner similar to the one in Fig. 43. This type of image, formally called T1-weighted, has the advan-

2. Image by ©Jan Ainali, Wikimedia Commons, CC-BY-3.0.

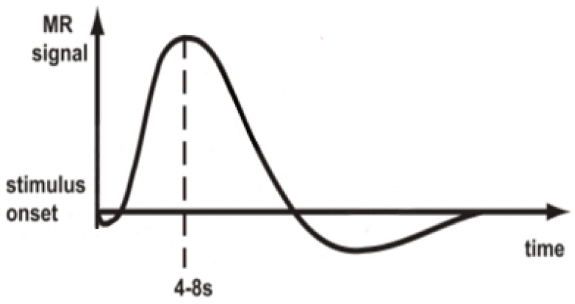
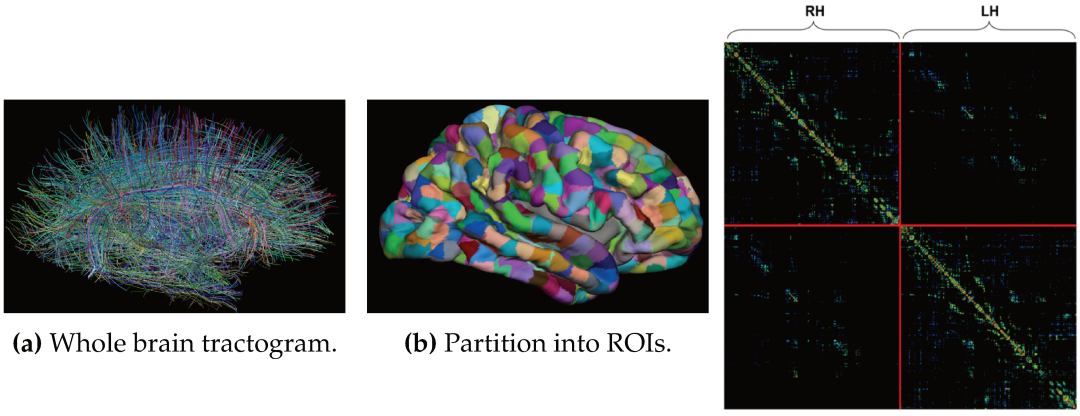


Figure 45 – Haemodynamic response function.

tage to make fat brighter and water darker. Thus, it provides a good contrast between the gray matter regrouping numerous cell bodies of neurons (dark gray) and the white matter indicating the axons (lighter gray). Brain images recorded with MRI are often called ***structural MRI*** images.

Brain activity can be measured with a technique derived from MRI called functional Magnetic Resonance Imaging (fMRI) [OEJ01]. This technique emerged from three successive discoveries in the 90s. The first one is a chemical discovery implying the hemoglobin molecule which carries oxygen in the blood to the cells. The oxygenated hemoglobin is not magnetic: it does not react to a magnetic field. However, the deoxygenated hemoglobin is paramagnetic: it becomes magnetic when it is placed in a magnetic field. Secondly, scientists found a way to record the ratio of oxygenated versus deoxygenated hemoglobin in the blood with MRI. Finally, it has been shown that the ratio of oxygenated versus deoxygenated hemoglobin is linked to neuronal activity: as active neurons need more oxygen and glucose, the blood flow increases locally around these neurons. It appears that the blood carries more oxygen than necessary. Consequently, the MRI signal increases when the neurons are more active. This MRI signal has been called Blood Oxygenation Level Dependent (BOLD) signal. It follows the haemodynamic response function characterizing the rapid delivery of oxygenated blood to activated neurons. As shown in Fig. 45, the BOLD signal peaks between 4 and 8 seconds after the stimulus and remains active for several seconds. The duration of the haemodynamic response is one of the major reason explaining the low temporal resolution of fMRI. This low resolution is also due to the fact that brain activity is recorded slice by slice: in typical experimental settings using a 3 tesla permanent magnet, the whole brain is imaged in about 2 seconds.

Using MRI, the different tissues composing the structure of the brain can be imaged as shown previously in Fig. 44. In this figure, the gray matter appears in darker than the white matter. However, it is difficult to infer the directions of the white fibers, and thus the underlying connectivity. White matter fibers can be visualized more accurately using another technique based on MRI, namely diffusion MRI [LB03]. This technique takes advantage of one of the chemical properties of axons: they repel water molecules. As a result, water molecules flow along the white fibers. The images resulting from diffusion MRI are called tractograms (see Fig. 46a). T1-weighted images are often recorded along with tractograms and parceled into a small group of predefined Region Of Interest (ROI) (Fig. 46b). Thus, the number of fibers between each region can be inferred and summarized into a structural connectivity matrix also called connectome [STK05] (Fig. 46c).



(a) Whole brain tractogram. **(b)** Partition into ROIs. **(c)** Connectivity matrix.

Figure 46 – Connections between all pairs of ROIs obtained from tractography and T1-weighted MRI. RH stands for right hemisphere and LH for left hemisphere. Adapted from ©2008 Hagmann et al., CC-BY-4.0 [Hag+08].

All these images, in particular functional ones, are preprocessed to remove unwanted artifacts (such as head movements) and to align all human brains to a standard template (such as the standard MNI-152 template [Fon+09]). Prepossessing techniques are more extensively described in this paper [Est+19] and on this website `https://fmriprep.org/en/stable/`. As stated before, human brains are similar but slightly different. To compare human brains, it is necessary to distort the recorded images to fit a standard model. In this way, decoding models can be trained on groups of individuals rather than on single individuals, thus greatly improving the statistical power of the studies.

3.1.3 From raw fMRI signals to brain activation maps

During a fMRI scan, 3D images of the brain are obtained every 2 seconds approximately; the temporal resolution depends on the scanner and the spatial resolution. Each 3D image is composed of cubes called ***voxels*** of about 1.5mm on each side. Each voxel contains a few million neurons. The purpose of brain decoding is to find the condition in which a person is set in from its brain activity. In a fMRI session, the experiments proposed to a subject generally follow 2 paradigms. In an event-related paradigm, a single stimulus is presented to a subject. In a block design, stimuli are repeated several times over a short period of time to increase the amplitude of the signal. Between events or blocks of events, the subject is at rest (e.g. fixing a white cross on a black screen).

Brain activity can be decoded directly from temporal fMRI signals. This type of decoding requires identifying stimuli onsets during the experiments and extracting relevant sequences for training a decoder. Since all the information on the experimental paradigms is made public along with the data, this is feasible but arduous. Additionally, as all conditions do not last the same amount of time, it is really complex to train a decoder with a variable size input. Another solution consists in decoding 3D brain activation maps.

3. Visualization of a contrast map from [Pin+20] with nilearn [Abr+14]. Voxels are averaged into 360 ROIs defined by Glasser atlas [Gla+16].

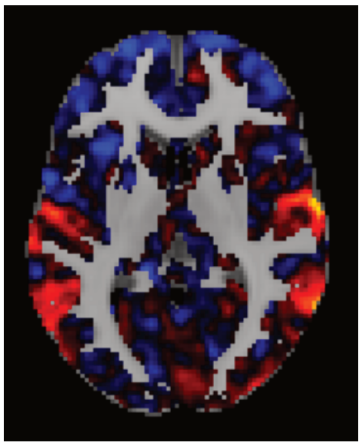


Figure 47 – Slice of a contrast map.

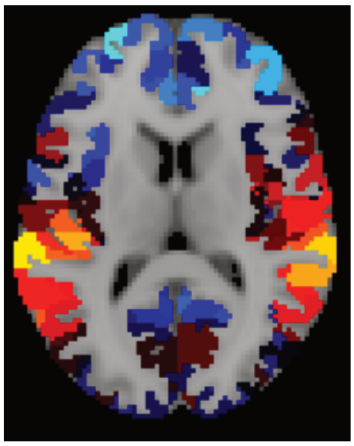


Figure 48 – Averaged contrast map.

Figure 49 – Example of fMRI data [3]showing regions of activation in a comparison between a task involving a complex stimulus (here, a body image 2-back task) and rest condition (here, fixing a cross on a black screen). The activations (colors) are shown against a background based on an average structural image.

Definition 31

A ***brain activation map*** is a statistical estimate modeling the extent to which brain activity in a voxel is related to an experimental condition during a session. It no longer contains a temporal dimension.

Brain activation maps are computed with a General Linear Model (GLM) [Mon11]. A GLM is applied on each voxel independently. It determines how the voxel responds to various conditions. The principle of the GLM is described in Fig. 50. The activity measured for the voxel over time is stored in the vector $\mathbf{Y}$. This activity depends on the stimuli and the task achieved by the subject but also on confounds of no particular interest (e.g. head motion). The purpose of the GLM is to model the activity in a voxel as a linear combination of these factors, called regressors. The essential regressors are called regressors of interest. They represent a task or a condition of interest (e.g. moving right hand, looking at a face). In Fig. 50, the first columns of the design matrix ($\mathbf{X}_1$, $\mathbf{X}_2$ and $\mathbf{X}_3$) are regressors of interest. They contain the idealized prediction of what the fMRI signal should look like if the voxel was activated by the stimulus. Other regressors, called nuisance regressors, represent experimental factors that confound the analysis. They can be empirically determined thanks to measurements taken during the fMRI session. In Fig. 50, they are represented by the last seven columns of the design matrix $\mathbf{X}$. According to the model, the coefficients β of the linear regression are derived to minimize the sum of the errors ϵ. All β coefficients associated with different voxels are then gathered to create a 3D ***beta map*** determining how the brain responds to a particular condition.

To know if a voxel is truly activated by a condition, the values of the beta maps are tested for statistical significance. The coefficients are first divided by a standard deviation depending on the design matrix and the sum of the errors. As such coefficients are assumed to follow a standard normal z distribution, the resulting 3D maps are called ***Z maps***. The voxels with the highest values are more likely to be activated by the condition. Voxels with a value lower than a certain threshold do not significantly respond to the stimulus. To perform brain decoding, unthresholded Z maps are considered.

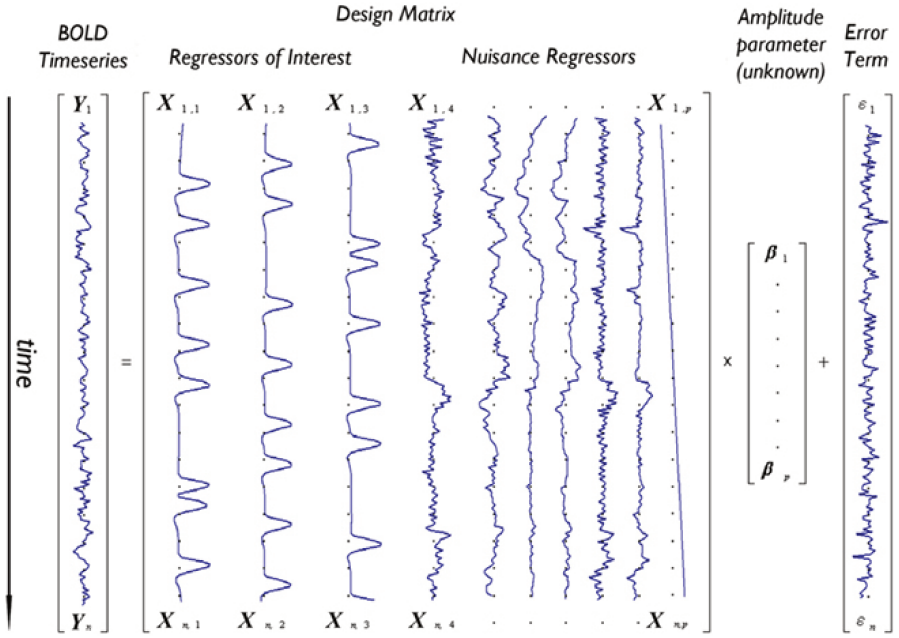


Figure 50 – Description of the GLM used to generate brain activation maps from fMRI signals. The time series within a voxel is represented by $\mathbf{Y}$. The model includes 10 regressors, 3 regressors of interest linked to three conditions studied in a fMRI session (e.g. moving tongue, left arm, right arm) and 7 nuisance regressors linked to experimental factors confounding the signal. Illustration from [Mon11], under CC BY license.

Finally, to detect patterns of activity linked to a particular stimulus, researchers try to remove from the measured activity any activity that is not particularly linked to this stimulus. For instance, to detect patterns linked to the identifications of written words, we can show the subject a word and look at its brain activity. However, to recognize the word, many other processes are engaged. The subject has to look at the screen, read the letters and then determine that the sequence of letters is a word. To remove unwanted brain activity, researchers employ a technique called subtraction method. For the previous example, the subject will be also asked to look at a sequence of random letters. Two Z maps will be calculated. One related to word recognition. The other related to reading a random sequence of letters. The Z map of the condition of interest will be the first one minus the second one. Such maps are called ***contrast maps***. Contrast maps are often computed between a condition of interest and a baseline state (i.e. rest), or between two conditions of interest. In this chapter, when we talk about brain activation maps or activation maps, we are referring to contrast maps.

The interest behind the decoding of statistical maps is not the same as the decoding of raw signals. Decoding short sequences of fMRI signals can be useful for technologies based on BCIs. For instance, this could help to mentally control prosthetic limbs in real time. On the other hand, by trying to decode brain activation maps, we try to know if certain cognitive functions are characterized by activation patterns.

In this final subsection, we detail a few fMRI studies, indicate where to find their data, and explain how they are analyzed.

3.1.4 fMRI studies: datasets and analyses

First, it is important to know that more and more neuroimaging data are freely shared on dedicated databases. For example, on OpenNeuro [Gor+17b], one can find raw data from several dozen studies and on NeuroVault [Gor+15], one can find non-threshold brain activation maps.

fMRI studies are conducted for specific purposes. They study the impact of different experimental conditions on the brain activity of a variable number of individuals. Some of the larger studies are listed in Table 2. The Human Connectome Project, the ARCHI Project and Brainomics contain a large number of individuals which are scanned while performing a certain number of tasks once. These studies aim at studying brain functions and inter-subject variability in a number of mental processes and sometimes also at understanding the origin of this variability using other sources of data (e.g. behavioral data, genetic data). Other studies such as Midnight Scan Club are more focused on intra-subject variability. Differences between subjects tend to be erased in group-average estimates whereas they contain meaningful information, especially in clinical settings. To study intra-subject variability, a small group of individuals can be scanned performing the same tasks several times. Finally, the ongoing study named Individual Brain Charting aims at investigating the functional principles governing a large range of cognitive, motor and sensory functions in healthy adults. In this study, tasks involving diverse cognitive, motor or sensory functions are performed by subjects. Data from fMRI studies are often analyzed with the statistical methods described above to obtain brain activation maps. Brain areas that are particularly activated in response to an experimental condition are reported in scientific papers.

Since many studies contain little data, the results of the studies often have a high false positive rate. To obtain more reliable areas, meta-analyses can be conducted on the results of a large number of studies. For example, the online tool Neurosynth [Yar+11] automatically generates a meta-analysis from numerous neuroimagining studies. After locating neuroimaging studies using keywords, the tool extracts the coordinates of brain regions reported in the studies to be significantly activated during a cognitive task. By gathering the different results, it infers a brain map on which the value of each voxel indicates the probability of the experimental condition if the voxel was activated. The tool can then be used to perform forward inference, i.e. to associate a brain activation map with a given experimental condition. It can also be used for reverse inference, i.e. to associate plausible cognitive functions with an activation map. Another tool for meta-analysis is NeuroQuery. It takes into input a word describing a cognitive ability and it returns a map showing the activations highlighted by most neuroimaging studies related to the input word.

It should also be noted that more and more studies are trying to focus on the analysis of individuals instead of averaging results over groups of individuals [Fed21]. This increases the resolution of the results and the sensitivity of detection.

In this manuscript, we do not seek to detect significantly activated regions under certain experimental conditions. Instead, we try to decode conditions from brain activation maps. In the next section, we show that training such a decoder is not straightforward as we face a significant problem: only a small number of contrast maps per condition are available in most fMRI studies.

Table 2 – Brain activation maps available in various task fMRI studies.

fMRI study	**# classes**	**# subjects**	**# samples per subject and per class**	**# samples per class** (total)
Human Connectome Project [4] [VE+12]	23	788	1	788 (18070)
Midnight Scan Club [5] [Gor+17a]	12	10	10	100 (1197)
Individual Brain Charting [6] [Pin+20]	197	13	2 to 6	22 to 78 (9532)
ARCHI Project [7] [Pin+07]	30	78	1	78 (2340)
Brainomics [8] [Pap+17]	19	94	1	94 (1786)

3.2 Decoding Brain Activity with Few-Shot Learning

3.2.1 Motivation

In this chapter, we are interested in the decoding of brain activation maps. We consider that decoding brain activation maps is a particular classification problem. Currently on natural images, the most efficient classifiers are DNNs [KSH12]. These DNNs are usually trained on large amount of data, so that they can generalize their decisions to new inputs.

Here, we want to explore whether a decoder trained on a subset of cognitive tasks can be easily adapted to detect new cognitive tasks with only a few training examples. We consider both intra-subject and inter-subject variability as a source of noise inherent to each class. However, even in this context, fMRI datasets often contain a relatively small amount of data. This is because fMRI studies are long and expensive. In Table 2, we describe the datasets released by some of the biggest fMRI studies on healthy adults. With the exception of the Human Connectome Project, which received a 5-year grant of $30 million, the other studies contain less than 100 samples per class.

In chapter 2, we have seen that the lack of training data is a recurring issue in machine learning. This makes the training of DNNs from scratch difficult. However, in chapter 2, we also presented a few alternatives to take advantage of the power of DNNs while circumvented the need for large amounts of data. These solutions, applied in the context of image classification, are often inspired from transfer learning. To summarize, a DNN, called backbone, can be pre-trained on data coming from many different sources and than, be adapted to the data of interest.

In the context of decoding the brain, we wonder how useful these methods are. It is very unlikely that a backbone trained on natural images helps improving the performance on brain activity maps, as the data structure is very different. Nevertheless, transfer could be achieved from activation maps representing certain conditions to new

4. https://identifiers.org/neurovault.collection:4337
5. https://identifiers.org/neurovault.collection:2447
6. https://identifiers.org/neurovault.collection:6618
7. https://identifiers.org/neurovault.collection:4339
8. https://identifiers.org/neurovault.collection:4341

activation maps representing other conditions, although these maps contain important levels of variability appearing between individuals, within a same individual and between experimental protocols.

Note that in our study, we choose to decode contrast maps instead of raw fMRI signals for two reasons. First, brain activation maps no longer have a temporal dimension. Therefore, the DNNs do not have to take the temporal dimension into account in their architecture. Second, the inputs are smaller, which allows us to train the decoding models faster. This allows us to evaluate more easily if the few-shot learning methods can be adapted to a different type of data and in particular if brain maps representations are transferable across cognitive domains.

In the following subsections, we describe related works that propose to decode fMRI brain activation maps or that use transfer methods on fMRI signals before presenting our own work.

3.2.2 Related works

Many recent works study brain activity decoding. Here, we present some of them that train decoders on structural images, on fMRI images or on preprocessed features. We also mention some works based on transfer learning.

Deep learning for structural MRI, functional MRI and preprocessed fMRI features

In neuroimaging, DNNs are more and more used. For instance, in this paper [Abr+20], a CNN learning 3D convolutional filters is trained to find the age and the sex of a person from structural MRI images. This work shows that the features extracted by DNNs enable to obtain better performance than standard machine learning methods. In another paper [Zha+21b], a DNN is trained from scratch to decode fMRI signals coming from the Human Connectome project [VE+12], which contains a relatively large number of samples per class. The classification problem amounts to distinguish 21 conditions from 10s windows of fMRI signals. After training, the conditions are identified with an average test accuracy of 90%. In this paper [Li+19], GNN have also been used to discover biomarkers handcrafted features from task-fMRI.

Knowledge transfer

As most fMRI studies contain a small number of examples per class, it is interesting to take advantage of the information contained in other studies to better solve a given problem. This paper [ZB20] uses transfer learning on time-varying fMRI signals. They retrieve the previously mentioned DNN of [Zha+21b] and use it as a pre-trained backbone for two novel datasets. One of these datasets is the Individual Brain Charting dataset. The other one contains new subjects of the Human Connectome Project which are not used for pre-training. The authors investigate the transferability of brain decoding across different settings: inter-subject transfer on the same tasks, cognitive domain transfer and inter-site transfer. In inter-subject transfer, knowledge is transferred from the DNN pre-trained on a set of subjects performing specific tasks to another set of subjects performing the exact same task. In cognitive domain transfer, the pre-trained DNN is adapted to classify new cognitive classes performed by the same subjects. Last but not least, in inter-site transfer, the pre-trained DNN is adapted to decode similar tasks stimulating the same cognitive domains but whose experimental design is significantly different. For these three scenarios, the reported results show a significant performance

boost with transfer learning compared to DNNs trained from scratch although transfer is very sensitive to inter-site variations.

Transfer learning has also been tested on brain activation maps. In a recent study [Men+21], brain activation maps have been gathered from several studies to jointly learn to decode them. The transfer method consists in jointly training a linear model to project the brain activation maps into a common subspace before being using their representations to train a logistic regression per study. A different logistic regression is learned for each study but the same subspace is used for all study. Precisely, the method can be described in three steps. The first two steps enable to project the maps into a common subspace, the last step consists in training all logistic regressions. For the linear projection, resting state data from the Human Connectome Project are first used to reduce the 3D brain activation maps to a combination of functional networks (linear combination of voxels). This enables to reduce the size of the samples (from 200000 voxels to 465 resting-state networks). Then, more than 30 fMRI studies are used to learn a relevant combination of these networks, where relevant means enabling to obtain better classification accuracies with the subsequent logistic regressions. This linear projection and the logistic regressions are jointly trained. As a result, the classification accuracy is increased for most datasets. The method learns to project brain activity onto a set of functional networks whose relative weights are relevant for decoding. One significant advantage is that these task-optimized networks can be easily interpreted: they can be displayed on a standard brain and the most important networks for a given class can be inferred.

Interpreting the results

The question of the interpretability of the trained models is relevant in neuroscience. Do we want to train classifiers with the highest accuracy as possible or do we prefer to gain understanding about how the brain works? In the latter case, we need to be able to explain why a particular sample was classified the way it was and what characteristics are important to recognize one class from another. This paper [HB18] contains an interesting anecdote about one of the first competitions launched to decode brain activity. The neural regions on which the winning strategy learned to focus on were regions related to physiological noise. So they were really useful to get good predictions but completely useless to understand how the brain works.

With non linear models such as DNNs, it is challenging to interpret the results. One may want to find out which features have been used to assign this class to this example. One may also want to know which features matter to differentiate a set of classes. In [Zha+21b; ZB20], saliency maps [Spr+15] are displayed to answer to this last question. For instance, after having trained a decoder to distinguish 5 different motor tasks (tongue, left hand, right hand, left leg, right leg), one can infer the input components most likely to modify the decoder's decision. The input on which the importance of these components is represented is called a saliency map. The field of interpretable deep leaning is really active and a few methods already exist to interpret results but many researchers argue that linear models are more easily interpreted [Men+21].

In the following subsection, we explain how we adapt few-shot learning methods to brain activation maps.

3.2.3 Adaptation of few-shot learning methods

A common few-shot learning strategy, presented in chapter 2 consists in pre-training a DNN on a large set of diverse data before adapting it to the problem with few examples. To test the efficiency of this strategy, we create a benchmark dataset similar to those currently used in few-shot learning. Recall that the image datasets on which the few-shot methods are tested have a particular structure: they contain many different classes. These classes are divided into three sets: base classes, validation classes and novel classes.

Some studies publicly release their brain activation maps on NeuroVault [Gor+15]. This is for instance the case of all datasets mentioned in Table 3. For our study, we choose to create a few-shot neuroimaging benchmark from the Individual Brain Charting (IBC) dataset. The brain activation maps of IBC are interesting for creating a large generic dataset on which the DNN will be pre-trained. Indeed, the IBC project was carefully designed to ensure that the experimental tasks performed by participants during the fMRI sessions activate a wide range of brain functions. Resulting brain activation maps represent a wide range of classes to decode. We assume that the diversity of classes mitigates the problem potentially posed by the fact that classes are represented by relatively few samples.

In this section, we present the conditions under which the IBC dataset was acquired before detailing the data that we selected to create a benchmark. Then, we present the few-shot learning methods that are tested in [Bon+20b].

IBC dataset

In the IBC project, the tasks performed by the participants are carefully selected. To perform each task, a screen is placed in the scanners. The subjects are asked to follow instructions indicated on the screen while the scanner records their brain activity. The whole experimental protocol as well as the preprocessing of the recorded signals is detailed in the articles [Pin+18] (release 1) and [Pin+20] (release 2). The videos used are available on GitHub [9].

For each task, two different acquisitions are performed using two opposite phase-encoding directions: Posterior to Anterior (PA) and Anterior to Posterior (AP). A direction has to be chosen as the fMRI scanner records a 3D image of the brain slice by slice. The chosen order for the slices may introduce a distortion. Along with the fMRI scanner, T1-weighted images are recorded to obtain an anatomical image of the brain of each participant. These images are used to align the brains of all participants and normalize them to the MNI-152 template representing a standard brain. The raw time-varying fMRI signals can be downloaded directly on NeuroVault [10]. For simplicity, we prefer to focus on decoding contrast maps as a first benchmark for few-shot learning. The data are easier to manipulate and various methods can be tested quickly.

As explained in the previous subsection, contrast maps result from GLMs inferred from the fMRI signals. The GLMs output 3D beta maps representing voxel-wise activity for each acquisition and for each studied condition. In a second step, various contrasts are defined by subtracting some of the previous maps, showing either the difference between two conditions of interest or the difference between a condition of interest and a baseline (e.g. fixing a white cross on a black screen). For our few-shot neuroimagining

9. https://github.com/hbp-brain-charting/public_protocols
10. https://openneuro.org/datasets/ds002685

Table 3 – Characteristics of the benchmark dataset for few-shot learning on brain activation maps. Original data coming from Individual Brain Charting.

# classes *Contrasts between a condition of interest and a baseline.*	**# samples** *Contrast maps averaged per brain regions.*	**# samples per class**
106	21 to 78	3848

benchmark, we only consider contrast maps resulting from the difference between a condition of interest and a baseline. Note that as some contrasts have been computed from an AP acquisition and other from a PA acquisition, fixed-effects (FFX) contrasts have been generated to compare both versions.

All contrasts have been released by the IBC team in NeuroVault [11]. In a nutshell, in the dataset on NeuroVault, 13 subjects performed 25 tasks, resulting in 9532 contrasts maps representing 197 different contrasts.

Few-shot neuroimaging benchmark

To gather a neuroimaging benchmark dataset for few-shot learning, we make several choices. First, we only select contrasts showing the difference between a condition of interest and a baseline (e.g. left hand button presses upon audio instructions, watch vertical checkerboard, mental subtraction upon video instruction) and we discard the contrasts involving two conditions of interest already studied separately (e.g. read or listen to sentences, move left foot vs right foot, relational comparison vs matching). We also remove two contrasts from a task called Bang as each one only contains 11 examples per class. Then, as said previously, some contrasts are encoded in an AP/PA direction and others take into account fixed effects (FFX). In the original release paper [Pin+18], they studied how well the signal could be explained by the phase-encoding with a per-voxel Analysis of Variance (ANOVA). They showed that the difference between phase encoding directions is significant in certain regions such as in the frontal areas. However, they also show that the subject identity had a stronger impact. Our purpose being to train a brain decoder working at the group level, and not at the subject level, we make two assumptions: 1) inter-subject variability is a source of noise inherent to each class; 2) AP/PA acquisition is also a source of noise. Thus, we keep the contrasts in the AP/PA directions, and we remove the contrasts corrected for fixed effects. This strategy enables us to keep more data.

We also decide to average the values of the brain activation maps over large brain regions. The parcellation comes from the Glasser atlas [Gla+16], which compresses several hundred thousand voxels into 360 ROIs.

Finally, the few-shot neuroimaging benchmark contains 106 types of contrasts (classes to decode). Each class contains between 21 and 78 samples, with a median of 33. The total number of data samples is 3848. Each data sample contains 360 values. A description of the original dataset on NeuroVault and all our preprocessing steps are described in two notebooks on GitHub [12].

11. `https://identifiers.org/neurovault.collection:6618`
12. `https://github.com/mbonto/fewshot_neuroimaging_classification`

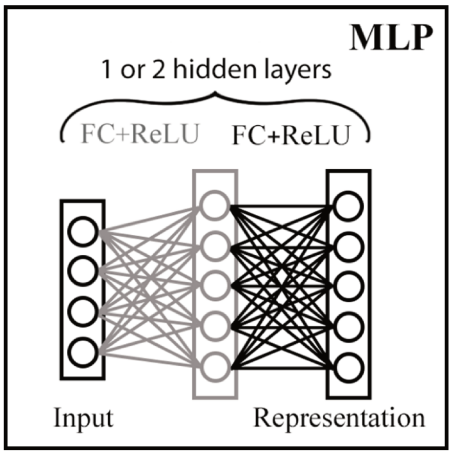


Figure 51 – Architecture of the feature extractor. FC stands for fully connected layer.

Few-shot learning methods

As for the few-shot learning methods evaluated on the neuroimaging benchmark dataset, we experiment an embedding-based method and a meta-learning based method.

For the embedding-based method, we choose to reproduce SimpleShot [Wan+19]. SimpleShot implements the simplest transfer learning scenario. A DNN is trained to recognize all base classes at once. To solve a few-shot problem, the representations of the new inputs are retrieved from the penultimate layer of this DNN. Then, a NCM is trained on top of the representations. For more details, see A.1.

For the meta-learning based method, we choose to reproduce MAML++ [AES19]. The method consists in training a DNN to solve numerous few-shot problems from the same initialization. What is learned during training are the initial values of the DNN's parameters. To solve a few-shot problem, the DNN with the learned initial parameters is simply fine-tuned for a few epochs on the new data. For more details, see A.1.

As for the architecture of the DNN, we adapt it to suit our inputs. For natural images, CNNs following standard architectures such as ResNet or WideResNet are used. In our case, the input data is smaller (only 360 coefficients) and they are not regularly structured (so the use of convolutional filters is not relevant). Consequently, we consider MLPs made of 1 or 2 hidden layers followed by a logistic regression. The architecture is illustrated in Fig. 51. During training, we vary 2 hyperparameters: the number of hidden layers (1 or 2) and the number of features per layer (64, 128, 256, 360, 512 or 1020). In the next subsection, the results obtained with few-shot learning methods are compared with a simple NCM directly trained on the inputs.

3.2.4 Results

To evaluate few-shot learning methods from a benchmark dataset, we need to divide it into the three sets: a base set, a validation set and a novel set. Recall that the backbone is pre-trained on the base classes, that its hyperparameters are chosen using the validation classes and that the performance of the few-shot methods is averaged on many few-shot problems generated from the novel classes. As shown in Fig. 55, we vary some hyperparameters of the MLP and select the ones obtaining the best accuracy on problems generated from the validation classes. The results in Table 4 are reported on problems generated from the novel classes.

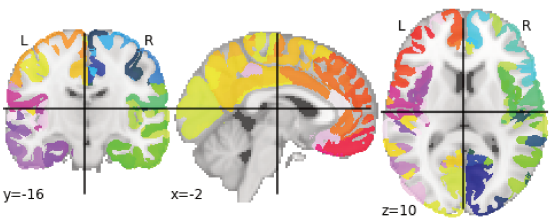


Figure 52 – 360 ROIs defined by GLASSER Atlas [Gla+16].

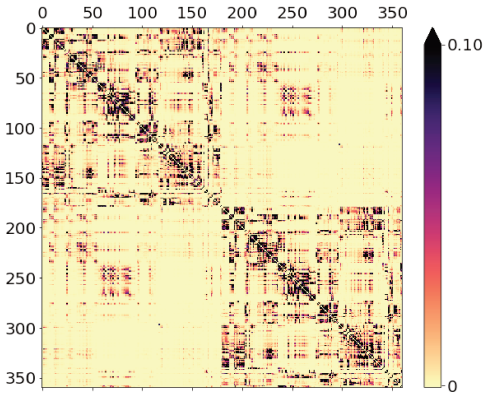


Figure 53 – Adjacency matrix of the anatomical graph on the ROIs [PVDV19].

The results are promising. They show that few-shot learning methods enable a huge boost in performance (more than 10% gain in accuracy). SimpleShot is most efficient than MAML++, as it is usually the case for standard image classification benchmark [Wan+19; AES19]. Thus, the variability in a set of cognitive tasks can be exploited to decode new cognitive functions with few examples.

However, from a neuroscience perspective, it would be interest to focus more on interpretability. We could look at different saliency maps obtained with and without transfer. It would also have been interesting to compare the small SimpleShot backbone pre-trained on some classes of the IBC project with the linear model of [Men+21] pre-trained on a larger amount of data. Additionally, from a BCI perspective, it would be interesting to try few-shot learning methods directly on time-varying fMRI signals as it could allow to decode brain activity in real time.

A last limitation of this results is that they do not exploit the underlying structure of the activation maps at all. Brain activation maps averaged per brain regions are not regularly structured as natural images, but brain regions are connected to each other through the connectome. As the connectome can be inferred from MRI techniques, we propose in the next section to investigate the relevance of the underlying brain structure for brain decoding.

3.3 Exploiting Structural Priors to Better Classify Brain Activity

3.3.1 Motivation

Our objective is to decode brain activation maps. Brain activation maps are noisy representatives of the brain activity induced by cognitive tasks. The noise is linked to the data acquisition process and also to the fact that a brain processes a lot of information that is not necessarily linked to the task to decode. Additionally, each map is a multivariate signal whose coefficients are associated with distinct brain regions. As numerous neurons connect these regions, the activity on different regions can be strongly correlated.

Relationships between brain regions can be inferred with different methods. As explained in section 3.1.2, diffusion MRI and tractography allow us to image the connectome. The resulting network can be assimilated to an ***anatomical graph*** [PVDV19], in which the nodes are brain regions and the edges represent the strength of connec-

tions between the regions. Graphs are ubiquitous to represent such irregular structures. Several tools have been designed in the Graph Signal Processing (GSP) field to extract meaningful information from graph signals. Additionally, connectivity between brain regions can also be inferred from resting-state fMRI data. A person's brain activity is recorded at rest in a scanner (e.g. looking at a white cross on a black background for several minutes). The correlation between signals recorded on pairs of regions can be calculated over time and used to create a functional graph. Last but not least, the anatomical and functional graphs can be averaged over several people and used as a reference for many others.

In this section, we try to leverage prior knowledge on the data structure to solve few-shot classification problems generated from our benchmark dataset. Interestingly the success of DNNs on image classification problems is due to several factors, including the exploitation with convolutional layers of the underlying structure of signals in images. As images have a regular structure (an object translated on an image is still the same object), successive convolutional layers learn to detect complex patterns on image portions, of interest for the classification problem. In the case of brain activity, the underlying spatial structure is irregular. Information is conveyed across neurons through white matter. It seems less evident that brain activity induced by a class be well represented by filters learning local spatial patterns. It is thus even more stimulating to search for alternative solutions.

3.3.2 Exploiting graph-structural priors

We propose to explore two different approaches to take advantage of the structure prior in a few-shot context, one which is more oriented deep learning and another one which is more oriented signal processing. The deep learning approach consists in inserting the a priori on the data structure in a classical MLP to see if we can improve the previous performance obtained with few-shot learning methods. The signal approach considers that the information is masked by two sources of noise: a random noise linked for example to the data acquisition and a noise depending on the connectivity of the cerebral regions that models the dependencies between these regions. We thus propose a simple model of the data distribution as well as an optimal classifier to solve a classification problem on such a model.

As a first approximation, for both approaches, the data structure prior is given by an averaged anatomical graph coming from an independent study [PVDV19]. The nodes of the graph are associated with the brain regions defined in the Glasser atlas. Thus, we use the same input samples as in the previous section. The brain atlas and the averaged adjacency matrix of the anatomical graph are respectively shown in Fig. 52 and Fig. 53.

In the following, we first formally present our two approaches before comparing them in a final subsection.

Deep learning approach

We continue to explore the potential of few-shot learning. To that end, we incorporate a graph filter inside the MLP we used previously. The resulting architecture is a simple GNN called SGC [Wu+19]. As shown in Fig. 54, the input is diffused on the graph before being handled by the MLP.

Formally, we denote $\mathcal{G}$ the anatomical graph, $\mathbf{A}$ its adjacency matrix where $\mathbf{A}_{i,i} = 0$ and $\mathbf{A}_{i,j} \geq 0$ is proportional to the number of white fibers between two brain regions i

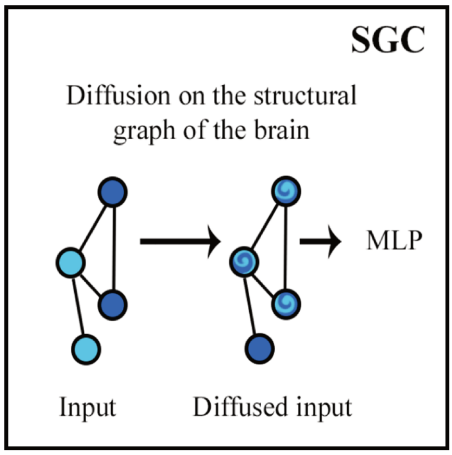


Figure 54 – Architecture of the feature extractor.

and j and $\tilde{\mathbf{D}}$ the degree matrix of $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$. Given $\mathbf{x}_i$ the i^{th} input (shot or query), we define the diffusion of the input on the anatomical graph by a matrix multiplication:

$$\mathbf{x}_{\text{diff}} \coloneqq \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{x} \,. \tag{27}$$

Signal processing approach

We want to try to exploit the known anatomical graph to improve decoding performance. In the literature, many articles propose to model graph signals as a function of a GSO. For instance, a graph can be used to model direct relationships between the coefficients of a signal [Seg+17]. The observed signals can also be seen as the result of a graph filter scrambling the elements of an originally sparse signal [Seg+16]. Here, we assume that the graph represents noisy interactions between brain regions and we propose a simple model related to this assumption [BFG21]. Each class is characterized by a constant vector called centroid. The contrast maps are supposed to result from the sum of three components: the centroid of their class, an isotropic noise and a graph-dependent noise. The isotropic noise reflects external sources of noise (e.g. data acquisition process, movements, irrelevant mental processes…). The graph-dependent noise accounts for intrinsic relationships between brain regions.

We denote $\mathcal{G}$ the anatomical graph composed of N nodes $\mathcal{V}$, $\mathbf{A}$ its adjacency matrix, $\mathbf{D}$ the degree matrix of $\mathbf{A}$ and $\mathbf{S}$ a symmetric Graph-Shift Operator (GSO) defined by $\mathbf{S} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$. We also denote $\boldsymbol{\mu}_c$ the centroid of the class c. Formally, the model requires two parameters $\alpha > 0$ and $\beta > 0$ associated with two random vectors $\mathbf{N}_1$ and $\mathbf{N}_0$ following multivariate normal distributions with mean $\mathbf{0}$ and covariance matrix $\mathbf{I}$. The random vector $\boldsymbol{X}_c$ characterizing the distribution followed by a contrast map $\mathbf{x}$ belonging to the class c is modeled by:

$$\boldsymbol{X}_c = \boldsymbol{\mu}_c + \alpha\mathbf{S}\mathbf{N}_1 + \beta\mathbf{N}_0 \,. \tag{28}$$

In the same paper [BFG21], we also provide an optimal classifier for data following such a model. Such data can be optimally classified with a LDA. Indeed, all classes follow a multivariate normal distributions with mean $\boldsymbol{\mu}_c$ and with the same covariance matrix $\boldsymbol{\Sigma} = \alpha^2\mathbf{S}^2 + \beta^2\mathbf{I}$. Since we assume that the classes are balanced, the LDA amounts to ***whitening*** the features (i.e. linearly transforming the values associated with

a brain region so that their covariance matrix becomes the identity matrix) before applying a NCM. For more details on LDA and NCM, see the preliminary chapter 1.1.4. The solution in the particular case of the model is called *graph-LDA*. It exploits the eigendecomposition of the GSO $\mathbf{S}$ into $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\intercal$, $\mathbf{\Lambda}$ being a diagonal matrix containing the eigenvalues $\lambda_1, \dots, \lambda_N$ in ascending order and $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_N]$ an orthonormal matrix. Graph-LDA requires three steps:

1. Projecting all contrast maps. *Given a contrast map* $\mathbf{x}$, $\hat{\mathbf{x}} = \mathbf{U}^\intercal\mathbf{x}$.
2. Normalizing their features. *Given* $\mathbf{\Delta} = \mathbf{\Lambda}^2 + \left(\frac{\beta}{\alpha}\right)^2 \mathbf{I}$, $\breve{\mathbf{x}} = \mathbf{\Delta}^{-\frac{1}{2}}\hat{\mathbf{x}}$.
3. Classify the normalized signals $\breve{\mathbf{x}}$ with a NCM.

The properties originally presented in [BFG21] proving the optimality of graph-LDA in the framework of our model are provided below.

Definition 32

Given $\boldsymbol{X}$ is the random vector characterizing a data sample $\mathbf{x}$ and Y the random variable characterizing its label y, an ***optimal classifier*** $h^\star$ is a classifier h maximizing the expected accuracy
$\mathrm{E}_{\boldsymbol{X},Y}(\mathrm{acc}(h(\boldsymbol{X}), Y))$ where,

$$\mathrm{acc}(h(\mathbf{x}), y) = \begin{cases} 1 & \text{if } h(\mathbf{x}) = y \\ 0 & \text{otherwise} \end{cases} . \tag{29}$$

Property 1

An optimal classifier $h^\star$ associates a sample $\mathbf{x}$ with the class c reaching the highest posterior probability $p_{Y|\boldsymbol{X}}(c|\mathbf{x})$.

Proof of property 1. We look for a classifier h maximizing $\mathrm{E}_{\boldsymbol{X},Y}(\mathrm{acc}(h(\boldsymbol{X}), Y))$. We denote $f_{\boldsymbol{X}}$ the probability density function of the random vector $\boldsymbol{X}$, p_Y the probability mass function of the random variable Y and $f_{\boldsymbol{X},Y}$ the joint probability density function of the couple $(\boldsymbol{X}, Y)$. By noticing that $\forall \mathbf{x}, \forall y$, $f_{\boldsymbol{X},Y}(\mathbf{x}, y) = f_{\boldsymbol{X}}(\mathbf{x})p_{Y|\boldsymbol{X}}(y|\mathbf{x})$, we have:

$$\begin{aligned} \mathrm{E}_{\boldsymbol{X},Y}(\mathrm{acc}(h(\boldsymbol{X}), Y)) &= \int_x \sum_y f_{\boldsymbol{X},Y}(\mathbf{x}, y)\mathrm{acc}(h(\mathbf{x}), y)dx \\ &= \int_x f_{\boldsymbol{X}}(\mathbf{x}) \sum_y p_{Y|\boldsymbol{X}}(y|\mathbf{x})\mathrm{acc}(h(\mathbf{x}), y)dx \\ &= \int_x f_{\boldsymbol{X}}(\mathbf{x})\mathrm{E}_{Y|\boldsymbol{X}=\mathbf{x}}(\mathrm{acc}(h(\boldsymbol{X}), Y))dx \\ &= \mathrm{E}_{\boldsymbol{X}}(\mathrm{E}_{Y|\boldsymbol{X}=\mathbf{x}}(\mathrm{acc}(h(\boldsymbol{X}), Y))) . \end{aligned} \tag{30}$$

Thus, maximizing $\mathrm{E}_{Y|\boldsymbol{X}=\mathbf{x}}(\mathrm{acc}(h(\boldsymbol{X}), Y))$ for each sample $\mathbf{x}$ would be optimal. By noticing that $\mathrm{acc}(h(\mathbf{x}), y) = 1$ if $y = h(\mathbf{x})$ and $\mathrm{acc}(h(\mathbf{x}), y) = 0$ otherwise, we have:

$$\begin{aligned} \mathrm{E}_{Y|\boldsymbol{X}=\mathbf{x}}(\mathrm{acc}(h(\boldsymbol{X}), Y)) &= \sum_y p_{Y|\boldsymbol{X}}(y|\mathbf{x})\mathrm{acc}(h(\mathbf{x}), y) \\ &= p_{Y|\boldsymbol{X}}(h(\mathbf{x})|\mathbf{x}) . \end{aligned} \tag{31}$$

So, the optimal classifier $h^\star$ is the one attributing to each sample $\mathbf{x}$ the class

$$h^\star(\mathbf{x}) = \underset{c \in [\![1,C]\!]}{\mathrm{argmax}}(p_{Y|\boldsymbol{X}}(c|\mathbf{x})) . \tag{32}$$

□

Definition 33

The ***discriminative function*** associated with the class c is denoted g_c. A class c is assigned to a sample $\mathbf{x}$ if $\forall c' \neq c,\ g_c(\mathbf{x}) \geq g_{c'}(\mathbf{x})$.

Property 2

Within our model, the LDA is an optimal classifier. It is characterized by C discriminative functions:

$$g_c(\mathbf{x}) = \mathbf{w}_c^\intercal \mathbf{x} + w_{c0} \text{ with } \begin{cases} \mathbf{w}_c = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c, \\ w_{c0} = -\frac{1}{2}\boldsymbol{\mu}_c^\intercal \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c. \end{cases} \tag{33}$$

Proof of property 2. We propose to define the optimal classifier through discriminative functions. An optimal classifier attributes to a sample $\mathbf{x}$ the class c maximizing $p_{Y|\boldsymbol{X}}(c|\mathbf{x})$. According to the Bayes'theorem,

$$p_{Y|\boldsymbol{X}}(c|\mathbf{x}) = \frac{p_{\boldsymbol{X}|Y}(\mathbf{x}|c)p_Y(c)}{p_{\boldsymbol{X}}(\mathbf{x})} \ . \tag{34}$$

As $\log$ is an increasing function, we can also choose the class c as the one maximizing

$$\log(p_{Y|\boldsymbol{X}}(c|\mathbf{x})) = \log(p_{\boldsymbol{X}|Y}(\mathbf{x}|c)) + \text{cst}_1 \ . \tag{35}$$

Here, we call $\text{cst}_1 = \log(p_Y(c)) - \log p_{\boldsymbol{X}}(\mathbf{x})$ because it has the same value for all c (assuming that the classes are balanced). As the samples belonging to class c follow a multivariate normal distribution with $\boldsymbol{\Sigma}$ being invertible (real symmetric), we can write:

$$p_{\boldsymbol{X}|Y}(\mathbf{x}|c) = \frac{1}{(2\pi)^{\frac{|\mathcal{V}|}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_c)^\intercal \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_c)\right] , \tag{36}$$

where $|\mathcal{V}|$ is the number of vertices in $\mathcal{V}$. Thus,

$$\log(p_{\boldsymbol{X}|Y}(\mathbf{x}|c)) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_c)^\intercal \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_c) + \text{cst}_2 \ , \tag{37}$$

where cst_2 also has the same value for all c. By expanding this term and removing the constant values for all classes, we obtain $g_c(\mathbf{x}) = \mathbf{w}_c^\intercal \mathbf{x} + w_{c0}$ with $\mathbf{w}_c = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c$ and $w_{c0} = -\frac{1}{2}\boldsymbol{\mu}_c^\intercal \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c$. □

Property 3

As classes are balanced, LDA amounts to whitening the data samples before applying a NCM. The data samples can be easily whitened by being projected onto the eigenvectors of the GSO $\mathbf{S}$.

Proof of property 3. Recall that $\boldsymbol{X}_c \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma})$. Given a matrix $\mathbf{P} \in \mathbb{R}^{\mathcal{V}\times\mathcal{V}}$, $\mathbf{P}\boldsymbol{X}_c \sim \mathcal{N}(\mathbf{P}\boldsymbol{\mu}_c, \mathbf{P}\boldsymbol{\Sigma}\mathbf{P}^\intercal)$. Given the diagonal matrix $\boldsymbol{\Delta}$ such that $\boldsymbol{\Delta}_{i,i} = \lambda_i^2 + \left(\frac{\beta}{\alpha}\right)^2$ and $\mathbf{P} = \boldsymbol{\Sigma}^{-\frac{1}{2}} = \alpha^{-1}\boldsymbol{\Delta}^{-\frac{1}{2}}\mathbf{U}^\intercal$, $\mathbf{P}\boldsymbol{X}_c \sim \mathcal{N}(\mathbf{P}\boldsymbol{\mu}_c, \mathbf{I})$.

Let us denote the projected samples $\breve{\mathbf{x}} = \boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{x}$ and the projected class means $\breve{\boldsymbol{\mu}}_c = \boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\mu}_c$. The discriminative functions become:

$$\begin{aligned} g_c(\mathbf{x}) &= \mathbf{w}_c^\intercal \mathbf{x} + w_{c0} \\ &= (\boldsymbol{\mu}_c^\intercal \boldsymbol{\Sigma}^{-\frac{1}{2}})(\boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{x}) - \frac{1}{2}(\boldsymbol{\mu}_c^\intercal \boldsymbol{\Sigma}^{-\frac{1}{2}})(\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\mu}_c) \\ &= \breve{\boldsymbol{\mu}}_c^\intercal \breve{\mathbf{x}} - \frac{1}{2}\breve{\boldsymbol{\mu}}_c^\intercal \breve{\boldsymbol{\mu}}_c \ . \end{aligned} \tag{38}$$

By noticing that the square distance between $\breve{\boldsymbol{\mu}}_c$ and $\breve{\mathbf{x}}$ is:

$$\begin{aligned} \|\breve{\mathbf{x}} - \breve{\boldsymbol{\mu}}_c\|_2^2 &= \breve{\boldsymbol{\mu}}_c^\intercal \breve{\boldsymbol{\mu}}_c - 2\breve{\boldsymbol{\mu}}_c^\intercal \breve{\mathbf{x}} + \breve{\mathbf{x}}^\intercal \breve{\mathbf{x}} \\ &= -2g_c(\mathbf{x}) + \text{cst}_3 \ , \end{aligned} \tag{39}$$

with cst_3 having the same value for all classes, we obtain the following conclusion. The class c maximizing $g_c(\mathbf{x})$ is the class whose the square distance between $\breve{\mathbf{x}}$ and its centroid $\breve{\boldsymbol{\mu}}_c$ is minimized. This characterizes a NCM classifier (samples $\breve{\mathbf{x}}$ mapped to the nearest class mean $\breve{\boldsymbol{\mu}}_c$). □

In practice, as the NCM is scale-invariant, we only need to infer the ratio $\frac{\beta}{\alpha}$ to apply graph-LDA. In the next section, this ratio is denoted σ.

3.3.3 Results

In this last subsection, we compare the results obtained on our few-shot learning benchmark for brain activation maps. We first present the results obtained from the deep learning approach exploiting additional data before presenting the results obtained from the signal processing approach by applying graph-LDA.

As in the previous section, the hyperparameters of the DNN architecture are selected on validation classes (see Fig. 56). The results, reported in Table 4 on novel classes, can be compared with the previous results of the previous subsection. The comparison shows that using the graph structure through a diffusion step (SGC) does not improve the classification accuracy. Several reasons can explain this result. First, SGC may not be adapted to our problem. When the features are diffused along an anatomical graph, they are smoothed. Important information may be lost in the process. This hypothesis is likely since several papers highlight the fact that cognitive processes encoding various mental functions have very different structures. For instance, in [CD16], brain functional networks (i.e. collections of widespread brain regions showing functional connectivity) are shown to be more integrated during a memory task and more segregated during a motor task). A simple diffusion filter is very unlikely to represent this variety. The averaged anatomical graph may also be unrepresentative of individual subject brains due to inter-subject variability.

The results obtained with graph-LDA are presented in Fig. 57. In the experiment, several classification methods are compared (NCM, LR, LDA) with or without preprocessing (steps 1 and 2 of graph-LDA). Graph-LDA is represented by the green dashed line (preprocessing + NCM). To perform the preprocessing, several ratio σ have been tested and the one leading to the best performance ($\sigma = 0.5$) have been kept. Each point is computed on 15 contrast maps not used for training. The implementations of the classifiers come from scikit-learn [Ped+11]. For the LDA, the 'lsqr' solver with the oracle approximating shrinkage estimator [Che+10] (for estimating the covariance matrix) is used. It is recommended for classification problems with few training examples and normally-distributed classes. Note that as the LDA requires estimating the covariance matrix, it cannot be evaluated in the 1-shot setting. We observe that the preprocessing boosts the performance of the NCM although it is still less efficient than a LR or a LDA. As before, this is either because the graph does not optimally represent the relationships between brain regions or because our model is not rich enough to represent the complexity of the data samples.

The results in Table 4 and in Fig. 57 are reported on the same set of novel classes. Notice that the results of the NCM are not the same in Table 4 and in Fig. 57. This is

because the NCM used in the table is based on the cosine similarity between a sample and a class centroid whereas the NCM used in the figure depends on the Euclidean distance. We observe that the data driven solution is more efficient than the considered model. Indeed with 5-shot, the accuracy obtained with a simple LDA and a LR with our preprocessing steps are around 80% whereas it is around 86.00% with few-shot learning. Similarly in the 1-shot case, the best score obtained with a LR and our preprocessing is around 57% compared to 72% with few-shot learning. This is not surprising as the few-shot learning method uses far more data than the model.

To conclude, we did not manage to improve the few-shot learning results using the anatomical graph contrary to other studies such as [Zha+21b]. Compared to the method in [Zha+21b], three differences emerge: they learned a more *complex graph filter* on a *functional graph* applied to *time-varying fMRI signals* (instead of contrast maps).

Learning method	5-shot		1-shot	
	MLP	SGC	MLP	SGC
NCM	70.56 ± 0.21		57.26 ± 0.20	
SimpleShot [Wan+19]	$\mathbf{86.00 \pm 0.16}$	85.14 ± 0.16	$\mathbf{72.54 \pm 0.20}$	71.96 ± 0.21
MAML++ [AES19]	82.31 ± 0.18	80.87 ± 0.18	67.64 ± 0.22	67.71 ± 0.22

Table 4 – Average accuracy and 95% confidence interval obtained with a NCM, MLP or SGC architectures trained separately on 10000 5-way problems generated from the novel dataset. The results with MLPs are discussed in subsection 3.2.4 whereas the results with SGC are discussed in subsection 3.3.3. Table adapted from [Bon+20b].

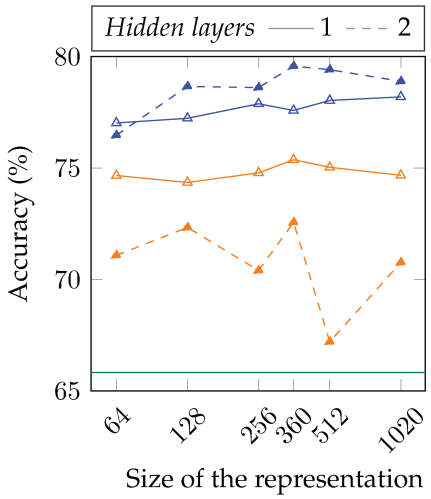


Figure 55 – Average accuracy obtained with a NCM or MLPs on 500 5-way 5-shot problems generated from the validation classes. Adapted from [Bon+20b].

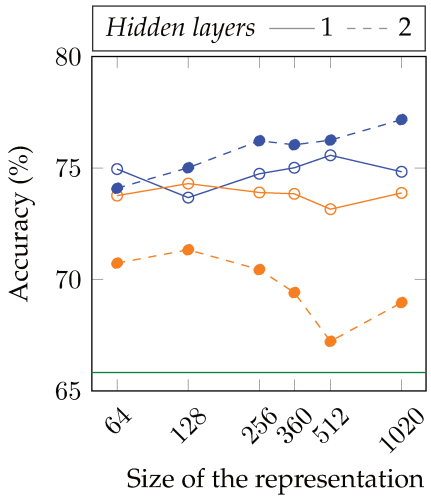


Figure 56 – Average accuracy obtained with a NCM or SGC on 500 5-way 5-shot problems generated from the validation classes. Adapted from [Bon+20b].

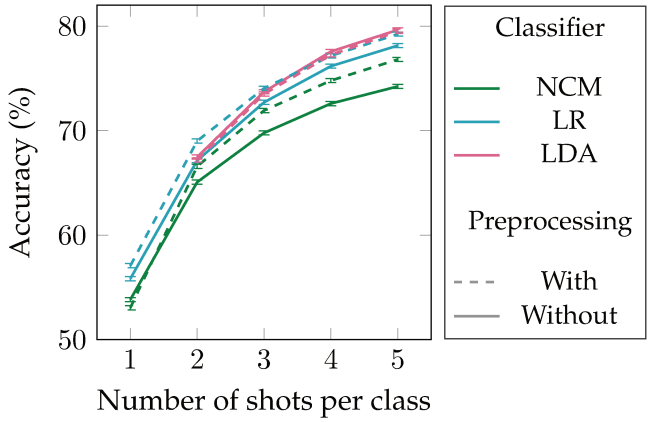


Figure 57 – Average accuracy and 95% confidence interval obtained on 10000 5-way classification problems generated from the novel dataset. Extracted from [BFG21].

Summary

This chapter is centered around the decoding of brain activity, application for which training examples are scarce. Indeed, we try to decode contrast maps generated from signals from MRI scanners. fMRI allows non-invasive recording of brain activity by measuring blood flow related to brain activity. Its spatial resolution is high (in the order of a millimeter) but its temporal resolution is low (in the order of a second). The activity measured during a session in which an individual repeats the same task can be processed to generate contrast maps. A contrast map is a 3D image of the brain composed of small cubes called voxels. The value associated with a small cube represents how much the brain region was activated by the studied task. In many studies, the number of available maps per studied condition is very low (less than 100 samples).

Through this manuscript, we studied few-shot classification problems. The *watchword* to solve these problems is *using more information*. Three tendencies can be distinguished in machine learning:

- the first one is *transferring knowledge from other datasets gathered for somehow similar problems*, as shown in chapter 2;
- the second one is *modeling data distribution* using a priori knowledge on the problem at hand. The goal is to use the model to find an optimal way to distinguish the classes.
- a third tendency can also be observed when merging the first two.

In this chapter, we have shown that we can take advantage of transfer learning through the few-shot learning methods currently used in computer vision [13]. The experiments presented come from the paper:

Myriam Bontonou et al. "Few-shot Learning for Decoding Brain Signals". In: *arXiv preprint arXiv:2010.12500* (2020)

As anatomical graph modeling the connections between brain regions can be inferred from MRI, we tried to exploit this knowledge in two different ways [13]: either by inserting a graph filter in the architecture used for few-shot learning, or by developing a classification method from a model of the data. The classification method, called graph-LDA is presented in the paper:

Myriam Bontonou, Nicolas Farrugia, and Vincent Gripon. "Graph-LDA: Graph Structure Priors to Improve the Accuracy in Few-Shot Classification". In: *arXiv preprint arXiv:2108.10427* (2021)

13. Codes to reproduce the experiments are available at `https://github.com/mbonto/graph-LDA` and at `https://github.com/mbonto/fewshot_neuroimaging_classification`.

Conclusion

Contents

THIS thesis is organized around the following question: can we learn from a few examples by transferring knowledge and/or exploiting the structure of the data? We mainly focus on natural image and neuroimaging classification problems.

Summary of Contributions

Deep learning turns out to be a fantastic tool. It has an enormous potential to assist human beings in many aspects of their lives. Researchers are currently working on AI-powered tools such as automatic translators or decision-making assistants able to quickly analyze diverse sources of data. However, to develop such applications, the deep learning field faces several challenges, among which is the need for large amounts of training data. The exact amount of data required to solve a given problem is unknown. That is why the motto of deep learning is the more data the better, although beyond a certain threshold, adding more data often results in a minimal performance gain as shown in Fig. 58.

The need for huge amounts of data is a problem in areas where data acquisition is difficult. For instance, in the medical field, designing a specific study to collect medical images is expensive. Moreover, the images taken daily on individuals around the world are rarely accessible for ethical reasons, and even if they were, some classes representing specific diseases would contain few representative examples. It is thus really important to find methods to optimize AI algorithms from a few examples.

Evaluating the generalization performance in few-shot learning

We have worked on an issue closely related to the problem of learning with few examples, which is: how to evaluate the generalization performance of models trained with few examples? Indeed for a given application, it is interesting to successfully train a model from a few examples but it is even more crucial to be able to evaluate if the trained model will perform correctly. Our methodology is quite simple. We seek a complexity measure whose value is related to the generalization capacity of a network. In other words, a measure informative of the error rate of the network on examples that it did not use during training. Such a measure can only be computed from the available information, i.e. the few training examples and the pre-trained network.

Finding such a measure is not obvious. Evaluating its relevance is not either. As an evaluation metric, we look at the correlation between the values of the measure and the generalization performance on several few-shot problems. Specifically, we consider a dataset containing several classes and a sufficiently large number of images per class. From this set, we define several few-shot problems with a fixed number of classes, training examples per class and test examples. However, correlation measures are not entirely relevant. They do not measure the causality between two variables. A complexity measure and the generalization performance can be highly correlated because of the effect of hidden variables. For example, one could imagine a situation in which the correlation is computed on a set of pre-trained networks having a varying number of layers. Assuming that the performance of the deeper DNNs is better, a measure counting the number of layers would probably be correlated with the performance while it is not very relevant in more general cases. To counter this problem, we test the predictive ability of the most correlated measures. With a first group of few-shot problems, we define a threshold value for each measure indicating whether the problem would be easy or hard to solve. Then, on a second group of few-shot problems, we re-

port the accuracy with which these problems are assigned to the easy or hard categories according to the value of the complexity measure.

In practice, we study three types of measures in two few-shot problem settings. The two settings are inductive few-shot learning and transductive few-shot learning. A few-shot problem contains N classes, K labeled training examples per class and Q test examples per class. In the transductive framework, the test examples are available during training without their labels. In the inductive setting, only the labeled training examples are available. The measures are either defined from the available data without their labels, the labeled data only, or the labeled data and the unlabeled examples (only possible in the transductive framework). The conclusion of the study is quite mixed. Only one measure reasonably predicts generalization performance, and only in the transductive framework. This measure estimates the confidence that a logistic regression has in its predictions. In summary, we have highlighted the problem of estimating generalization ability without a test set, but we have not found a convincing complexity measure for the inductive setting. As we have mentioned several times, the search for complexity measures is a very active area in deep learning. Developing such a measure would allow us to better understand the functioning of the networks and why not to optimize them.

Applicability of few-shot learning to practical problems: from natural images to neuroimaging

In this thesis, we have also worked on another facet of few-shot learning: its applicability to practical problems. We study here the decoding of the brain activity of a person performing various cognitive tasks. This problem is similar to a classical image classification problem except that the nature of the signals is different. The brain activity of a person is summarized by a 3D contrast map where each voxel indicates how much the corresponding brain area was activated during the task compared to a baseline state. Since decoding is similar to a classical classification problem and since the state-of-the-art methods for image classification are CNNs, deep learning is more and more adapted to decode brain activity. One limitation remains that DNNs are effective when they are trained on a lot of data. They successfully learn to identify relevant features for each class. Unfortunately, in the context of brain decoding, the databases are not as extensive as those for natural images. In this manuscript, the classes we consider often contain less than a hundred examples. Therefore, few-shot learning techniques seem to provide us a good opportunity to solve the problem of brain decoding.

For image classification, most few-shot learning methods consist in training a DNN, called backbone, on a generic dataset. The data samples of the few-shot problem are then processed by the backbone. Representations of the data samples can be retrieved within the backbone, and used as a proxy of the original sample to train a simple classifier.

In this manuscript, we propose to face the lack of neuroimaging data by adapting few-shot learning methods validated on natural images. To that end, we define a generic dataset gathering contrast maps from the IBC project [Pin+18; Pin+20]. Although each class is represented by a small number of contrast maps (between 21 and 78), many classes are studied at the same time. Our choice is based on the assumption that a backbone pre-trained on numerous classes covering diverse cognitive domains would learn rich representations of the data samples. In practice, we compress the values of the 3D contrast map over 360 brain regions defined by the classic Glasser at-

las [Gla+16]. We thus reduce the size of each example from several hundred thousand voxels to 360 values.

We divide the classes of IBC into two subsets. A base set containing a large number of classes to pre-train the networks. A novel set containing a smaller number of classes to generate few-shot problems. On one hand, we try an embedding-based method. We pre-train a small MLP to distinguish all base classes at once. In parallel, we try a method inspired from meta learning. The initial parameters of a small MLP are learned so that it solves a large number of few-shot problems with only a few training iterations. During pre-training, these few-shot problems are generated from the base classes. Both methods are evaluated on numerous few-shot problems generated from the novel classes. In our study, the performance after knowledge transfer is much better than that obtained with a simple classifier trained directly on the few-shot problems. As for the natural images, the meta learning method performed worse than the embedding-based method learning to extract features relevant to distinguish all classes.

In parallel, we also seek to take advantage of the intrinsic structure of the contrast maps. CNNs obtain very good classification performances on images thanks to the convolution layers learning to detect combinations of small patterns on the data. However, contrast maps do not have the same regular structure as objects on natural images. In contrast maps, activity-related signals can be quite diffuse. As the information propagates through the axons of the neurons, which can themselves be imaged, we try to take advantage of this a priori knowledge of the data structure. After averaging the contrast maps by brain region, we assume that the dependencies between signals from different regions are partially represented by an anatomical graph. Each node of this graph is associated with one of the 360 regions previously defined. The weights of the edge between two nodes are proportional to the number of fibers connecting the two associated regions. In this manuscript, we report two different ways of exploiting prior knowledge on the data structure: the first one is closer to deep learning and the second one is more related to statistical signal processing.

In a first attempt, we insert a graph filter at the beginning of our MLP. This filtering process is really simple. For each contrast map, it simply consists in diffusing the values associated with the brain regions on the graph, so that the signal associated with a region is averaged with the values taken by regions connected to it. However, this filter does not improve the transfer performance. This may be due to the fact that the division of the 3D image into brain regions is very coarse (there are only 360 regions). It is possible that the same filter is more efficient on a finer parcellation.

In a second attempt, we model the data distribution with a relatively simple model. Each class is represented by a reference vector. The contrast maps are assumed to be the result of the sum of this reference vector with two types of noise: an isotropic noise and a graph-dependent noise. The isotropic noise models external sources of noise related for example to data acquisition. The graph-dependent noise accounts for intrinsic relationships between brain regions. With our model, each class follows a multivariate Gaussian distribution. This type of classification problem is optimally solved by a Linear Discriminant Analysis (LDA). LDA amounts to whitening the data before applying a Nearest Class Mean (NCM). To whiten the data, one would have to estimate the covariance matrix which is difficult with few examples. However, thanks to the model, the covariance matrix is estimated from the graph with only a single parameter. We apply the whitening steps and compare the results of a NCM, a LR and a LDA for which the covariance matrix is estimated with a particular technique designed for few-

shot settings. We show that the preprocessing help to improve the performance of both NCM and the LR on the input data but it does not work better than the LDA. We conclude that the model used is maybe not rich enough to represent the data complexity. It is also possible that the average anatomical graph we use is not sufficiently representative of the brain structure of different individuals. Once again, the parcellation of the brain into regions is maybe too coarse.

Other contributions

Aside from these two main topics, we had the opportunity to work on other problems. Notably, we published a paper [Bon+19c] introducing a unified formalism to describe several Graph Neural Networks (GNNs), which are an extension of DNNs to graph signals. The comparative reading induced by the formalism helps choosing the GNN architecture that most suits a particular problem. In another line of work, we leverage data structure with graphs to improve deep learning. We propose a new loss function taking into account the relative distances between the examples within a DNN to make the classification more robust. We also propose a new distillation technique that enables to compress a teacher network into a smaller network while keeping the same performance. These works enabled us to familiarize with the fields of graph signal processing and deep learning, and to face some of the challenges that exist in these fields.

Future Works and Opening

The work developed in this manuscript leads us to several interesting reflections. Some of them invite us to compare the methodology we have proposed with other methods. Others more generally question neural networks and the philosophy behind deep learning.

Directions for future work

First, when we tried to improve the performance of a classifier trained to recognize the cognitive tasks performed by a person based on his or her brain activity, we assumed the dependencies between the signals measured on different brain regions are partly represented by the anatomical graph. We first thought that learning directly 3D convolutional filters on the brain activation maps would not be optimal as brain signals are not really localized and as patterns of activation do not have the same meaning when they are translated over the brain. Nevertheless, it would be interesting to compare the performance of DNN trained on 3D activation maps with the ones of DNNs trained on the brain average regions. The averaged brain contains less information and the 3D CNN might succeed to detect combinations of activated brain regions meaningful with respect to a particular task.

Second, the decoding application also opens up questions about the interpretation of the results. We have shown that transfer helps to improve the classification score but we have not studied in detail why. We do not know which part of the data is important for the network to recognize which class. With current interpretation techniques, it would be possible to generate saliency maps highlighting regions whose values matter for choosing a class among other classes. However, the decisions are not as interpretable as those obtained with other algorithms. For example, in this paper [Men+21], a linear model is trained to classify brain activation maps from data from many studies. It would be interesting to compare the performance of this linear model and a trained

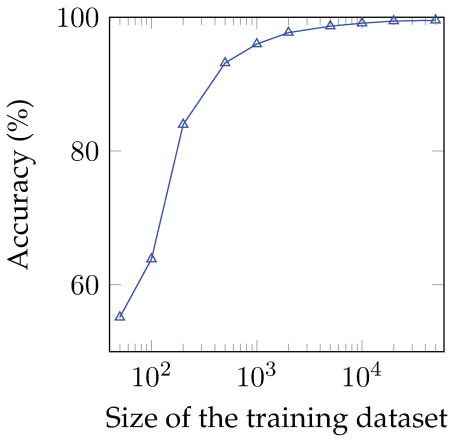


Figure 58 – Accuracy obtained on the test set of MNIST with a ResNet-18 trained on a variable number of training examples from MNIST. In each training set, the classes are balanced.

neural network on the same data. We would see if the non-linearities that make decisions less interpretable increase the performance in neuroimaging as it is the case on natural images.

Finally, we can also think about the generic dataset we gathered in our few-shot neuroimaging benchmark. All data samples come from the same neuroimaging study. It would also have been possible to gather data from several other research studies. This opens the question of how much data is needed to better train a model. The issue of this fundamental question goes beyond few-shot learning. For a given problem, what data do we need to improve our performance? How much will performance increase? Can performance be decreased when we add data?

Open questions

In this final part, we mention some general reflections beyond the few-shot learning framework.

The first reflection concerns the impact of the training dataset on the performance of the models. Training a model on a too small data set reduces its performance. The data does not represent the variability of the future data on which it will be applied. One could therefore say that the more data available, the better. However, the fact is that at some point, adding data leads to a very small gain in performance. Here is an example on MNIST in Fig. 58. 10000 examples are needed to get 99.13% accuracy. Another 10000 examples are needed to gain 0.32% and 30000 additional examples enable to gain 0.12%.

Can adding more data harm the generalization performance? In other words, is it better to use a small carefully curated dataset or a huge collection of noisier data samples? The question arises for neuroimaging, where many studies contain a few examples per class. Does adding these studies to our generic dataset improve the performance, decrease it or even modify it?

In the context of few-shot learning, the issue is particularly relevant. Let us consider that we have a limited budget to generate and annotate new data for our generic dataset. Is it better to add new classes to our generic dataset in order to diversify it or should we increase the number of examples per class? This paper [SCA20] investigates

the issue on natural images and concludes that one should prioritize a large number of different classes. Other questions are also raised on the influence of the similarity between base and test classes.

Last but not least, we would like to point out a philosophical question about deep learning. As a professor, we explain to students that in deep learning, we are replacing hand-written functions with programs that can learn by themselves from a dataset. Access to a wide variety of data replaces the need for human intervention. We then introduce CNNs by explaining that the networks learn better by exploiting an a priori on the structure of the images. In theory, an MLP could learn to perform the same task but it would have to learn many more parameters and for that many more data are often needed. Finally, in the context of few-shot learning, in which few data are available, we try to exploit all the available a priori information on the problem. Again, if a good backbone was available to extract adequate representations of the data samples, these a priori would not be necessary. For example, in automatic language generation, a model has been trained on millions of texts in an unsupervised way by predicting the word following a text beginning. In this article [Bro+20], the authors show that text representations from this network allow to solve new tasks by training only on a few examples. We conclude that today many tasks are facilitated by the introduction of a priori in networks but that in the future, these a priori may be less and less necessary.

Appendix A

Methods and Algorithms

A.1 Few-shot learning methods

The notations used in this section are the following. We consider a N-way K-shot Q-query few-shot problem. The i^{th} example (shot or query) is $\mathbf{x}^i$. Its representation obtained from a pretrained backbone is $\mathbf{f}^i$. The one-hot vector representing its class is $\mathbf{y}^i$. The output of a LR is $\hat{\mathbf{y}}^i$. In the methods presented below, several DNN architectures can be used for the backbone.

MAML++

MAML++ [AES19] is a few-shot learning method following the meta-learning paradigm. It is an optimized version of a previous method called Model Agnostic Meta-Learning (MAML) [FAL17]. The purpose of MAML is to learn initialization parameters for a backbone such that after a few epochs of training on a few-shot task, it achieves a strong generalization performance. The training set of the few-shot task is often called a support set and the generalization is evaluated on a target set (also called query set). The training set and the target set contain samples from the same classes. As in all few-shot learning methods, the backbone is trained on several tasks generated from a set of base classes. Its hyperparameters are selected on validation classes after training. The results reported in articles are computed on novel classes.

Here are the equations describing the training of a backbone with MAML. A backbone with parameters θ is denoted f_θ. Its initial parameters are denoted θ_0. The support set associated with a task b is denoted S_b whereas the target set is denoted T_b. The initial parameters of the backbone are updated through two steps, with an inner-loop update process and an outer-loop update process.

First, the inner loop computes the performance obtained on various few-shot tasks after having updated the parameters of the same initial backbone a small number of time for each task independently. To that end, B few-shot tasks are generated from the base classes. For each task b, the parameters of the backbone are updated L times on the support set S_b. The updated parameters on data from the task b after i steps can be expressed as:

$$\theta_i^b = \theta_{i-1}^b - \alpha \nabla_\theta \mathcal{L}_{S_b}(f_{\theta_{i-1}^b}) \,, \tag{40}$$

where α is the learning rate, θ_i^b represents the parameters of the backbone after i updates on the support set of the task b, ∇ is the symbol for gradient and $\mathcal{L}_{S_b}(f_{\theta_{i-1}^b})$ is the loss

computed on the support set of the task b after $i-1$ updates. The total loss computed from the B tasks is:

$$\mathcal{L}_{\text{meta}}(\theta_0) = \sum_{b=1}^{B} \mathcal{L}_{T_b}(f_{\theta_L^b}(\theta_0)) \ . \tag{41}$$

During the outer-loop update process, the initial parameters are updated to minimize this loss:

$$\theta_0 = \theta_0 - \beta \nabla_\theta \sum_{b=1}^{B} \mathcal{L}_{T_b}(f_{\theta_L^b}(\theta_0)) \ , \tag{42}$$

where β is the learning rate. In the original paper, $\mathcal{L}$ is the cross-entropy loss.

As said previously, MAML++ is an optimized version of MAML. The authors of MAML++ highlighted several issues leading to training instability with MAML and solved them with MAML++. More details can be found in the original paper [AES19].

SimpleShot

In SimpleShot [Wan+19], the backbone is trained from scratch to minimize the cross-entropy loss on all classes at once, as in traditional deep learning. To solve a few-shot problem, the features of the data samples are the ones obtained after the penultimate layer of the backbone. Then, each feature vector $\mathbf{f}^i$ is divided by its l^2 norm $\left\|\mathbf{f}^i\right\|_2$. Finally, a NCM is used on top of the normalized extracted features.

S2M2$_R$

In S2M2$_R$ [Man+20], the backbone is trained from scratch on all classes at once to minimize 3 losses: a classification loss $\mathcal{L}_{\text{class}}$, an auxiliary loss $\mathcal{L}_{\text{ss}}$ inspired from self-supervised learning and a Manifold Mixup loss $\mathcal{L}_{\text{mm}}$. More precisely, it is first trained to minimize the $\mathcal{L}_{\text{class}} + \mathcal{L}_{\text{ss}}$. Then, it is fine-tuned with $\mathcal{L}_{\text{mm}}$ for a few more epochs.

The auxiliary loss is based on an auxiliary task: the images are rotated by a certain degree ($0°, 90°, 180°$ or $270°$) and the task consists in predicting the angle of rotation applied to each image. A LR is trained to recognize these 4 classes from the features of the training examples (obtained in the penultimate layer of the backbone, just before the other LR trained to recognize the classes of the images). Formally, given an example i, the self-supervised loss is computed by:

$$\mathcal{L}_{\text{ss}} = \frac{1}{4} \sum_{\mathbf{r}^i} \mathcal{L}_{\text{CE}}(\mathbf{r}^i, \hat{\mathbf{r}}^i) \, , \tag{43}$$

where $\mathbf{r}^i$ is the one-hot vector associated with the rotation of the example i, $\hat{\mathbf{r}}^i$ is the vector outputted by the LR on the rotated input example i and $\mathcal{L}_{\text{CE}}$ is the cross-entropy loss.

The classification loss is the standard cross-entropy loss on the rotated training examples computed at the output of the second LR (last layer of the backbone). For each example i, it is defined as:

$$\mathcal{L}_{\text{class}} = \frac{1}{4} \sum_{\mathbf{r}^i} \mathcal{L}_{\text{CE}}(\mathbf{y}^i, \hat{\mathbf{y}}^i(\mathbf{r}^i)) \, , \tag{44}$$

where $\mathbf{y}^i$ is the one-hot vector associated with the class of the example and $\hat{\mathbf{y}}^i(\mathbf{r}^i)$ is the output of the LR on the example rotated by the degree associated with $\mathbf{r}^i$.

Finally, to compute the Manifold Mixup loss, pairs of training examples are mixed. Their labels are also mixed. This loss is formulated to help the backbone to associate the relevant mixed label with the mixed features. Formally, given two training examples i, j, their feature vectors after one of the first layers of the backbone are mixed $\text{Mix}_\lambda(\mathbf{f}^i, \mathbf{f}^j)$, their labels are also mixed $\text{Mix}_\lambda(\mathbf{y}^i, \mathbf{y}^j)$. We denote the class chosen by the LR after the examples have been mixed on a certain layer by $\hat{\mathbf{y}}(\text{Mix}_\lambda(\mathbf{f}^i, \mathbf{f}^j))$. The Manifold Mixup loss is computed by:

$$\mathcal{L}_{\text{mm}} = \mathcal{L}_{\text{CE}}(\text{Mix}_\lambda(\mathbf{y}^i, \mathbf{y}^j), \hat{\mathbf{y}}(\text{Mix}_\lambda(\mathbf{f}^i, \mathbf{f}^j)))\,, \tag{45}$$

$$\text{where } \text{Mix}_\lambda(a, b) = \lambda a + (1 - \lambda)b\,. \tag{46}$$

The coefficient λ is randomly generated from a $\beta(2, 2)$ distribution.

To solve a few-shot problem, the features of the data samples are extracted from the penultimate layer of the pretrained backbone. Each feature vector is divided by its l^2 norm $\left\|\mathbf{f}^i\right\|_2$. Finally, a LR is trained from scratch on top of the normalized feature vectors. Note that the same classifier (normalization + LR) is used during training.

Transfer+Graph Interpolation

The few-shot method called Transfer+Graph Interpolation [HGP21] is a method designed to extend inductive few-shot methods to a transductive setting, in which additional unlabeled examples of the new classes are available during training. Transfer+Graph Interpolation has been designed to be easily used on top of any pretrained backbone. After extracting the representations of the examples from the pretrained backbone, the representations are diffused along a similarity graph as in SGC [Wu+19]. Then, a LR is trained on top of the diffused representations.

Formally, given a pretrained backbone, the representations of the examples are gathered in $\mathbf{F} \in \mathbb{R}^{N(K+Q)\times d}$. A cosine similarity graph is built from the representations of the examples. Each vertex of the graph represents an example. Two vertices are connected by an edge weighted by the cosine similarity of their representations. The weights are stored in the adjacency matrix $\mathbf{A}$. Given two vertices i, j and their feature vectors $\mathbf{f}^i$, $\mathbf{f}^j$, the adjacency matrix coefficient $\mathbf{A}_{i,j}$ is defined as:

$$\mathbf{A}_{i,j} = \cos(\mathbf{f}^i, \mathbf{f}^j) \text{ where } \cos(\mathbf{f}^i, \mathbf{f}^j) = \frac{\mathbf{f}^{i\intercal}\mathbf{f}^j}{\|\mathbf{f}^i\|_2 \|\mathbf{f}^j\|_2}\,. \tag{47}$$

In the original study and in this manuscript, the representations do not contain negative values (they are extracted after a ReLU function). So, the output of the cosine similarity function ranges from 0 (orthogonal vectors) to 1 (aligned vectors). The idea is to smooth the representations by averaging them with their most similar neighbors. Thus, the graph is adapted to only connect vertices with the most similar representations. Self-loops are removed. Then, $\mathbf{A}_{i,j}$ is kept only if it is one of the k largest values per row or per column of $\mathbf{A}$. Finally, given the diagonal degree matrix $\mathbf{D}$, the resulting matrix is normalized as follows:

$$\mathbf{E} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\,,\ \text{where } \mathbf{D}_{i,i} = \sum_j \mathbf{A}_{i,j}\,. \tag{48}$$

At the end, given the identity matrix $\mathbf{I}$, and two constant α, κ, the new features are obtained by propagating the extracted features as follows:

$$\mathbf{F}' = (\alpha\mathbf{I} + \mathbf{E})^\kappa\mathbf{F}\,. \tag{49}$$

Algorithm 2: Impact of the choice of shots within classes on generalization.

```
standard_deviations ← [ ];
for trial = 1 → 100 do
    randomly select 5 classes;
    accuracies ← [ ];
    for run = 1 → 100 do
        randomly select 5 shots and 50 queries
        per class;
        train a LR;
        append the test accuracy to accuracies;
    end for
    append the standard deviation of
    accuracies to standard_deviations;
end for
return average of standard_deviations
```

Algorithm 3: Impact of the choice of classes on generalization.

```
standard_deviations ← [ ];
for trial = 1 → 100 do
    select 5 random shots within each class;
    accuracies ← [ ];
    for run = 1 → 100 do
        randomly select 5 classes and 50
        queries per class;
        train a LR;
        append the test accuracy to accuracies;
    end for
    append the standard deviation of
    accuracies to standard_deviations;
end for
return average of standard_deviations
```

In this manuscript, the hyperparameters used are the one advised by the original paper: $\alpha = 0.75$, $k = 15$ and $\kappa = 1$.

A.2 Case study on mini-ImageNet

Standard deviations on few-shot problems on fixed classes or on fixed shots within classes

The standard deviation of the accuracy on randomly generated 5-way 5-shot 50-query few-shot problems is computed in two different ways. With Algorithm 2, the impact of the choice of shots within classes on the generalization gap is evaluated. With Algorithm 3, the impact of the choice of classes is evaluated.

Overlap between classes

The overlap between the representations of the 600 examples of a class in mini-ImageNet with the 600 examples of another class in mini-ImageNet is computed in three steps.

- The cosine similarities are computed between all representations of the examples belonging to one of the two classes.
- A cosine similarity graph where each node is an example is built. The most similar examples are linked with edges weighted by their cosine similarity.
- *Louvain's community detection algorithm* is applied to detect communities inside the graph. Our measure of overlap is the average entropy of the classes within each community. When the classes are well separated, each community should only contain one class and therefore, the entropy should be zero.

In a few-shot learning context, our measure of overlap is particularly relevant. Indeed, in deep learning, we often look for a classifier minimizing the cross-entropy CE on each example $\mathbf{x}$. In order to formally define the cross-entropy, we need to introduce several notations. Let us call X the random variable representing the examples $\mathbf{x}$. The set of values taken by X is denoted $\mathcal{X}$. Similarly, we call Y the random variable representing the labels $\mathbf{y}$ and $\mathcal{Y}$ the set of values taken by Y. The true distribution of the class of an example $\mathbf{x}$ is denoted $P(Y|X = \mathbf{x})$. The distribution estimated by a classifier

is denoted $P'(Y|X = \mathbf{x})$. For each example $\mathbf{x}$, the cross-entropy CE is defined as:

$$\text{CE}(P(Y|X = \mathbf{x}), P'(Y|X = \mathbf{x})) = -\sum_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y}|\mathbf{x}) \log_2 P'(\mathbf{y}|\mathbf{x}) \ . \tag{50}$$

We try to minimize the expected cross-entropy $E_{\mathbf{x}\in\mathcal{X}}(\text{CE}(P(Y|X = \mathbf{x}), P'(Y|X = \mathbf{x})))$.

Here, we consider an additional variable: the community to which the examples belong described by the random variable C. In few-shot learning, as the small number of training data and their high dimensionality prevent to learn complex distributions, it is often assumed that examples with similar representations belong to the same class. So, we make the hypothesis that the distribution $P'(Y|X = \mathbf{x})$ of the class of $\mathbf{x}$ estimated by a classifier is the same as the distribution $P''(Y|C = c)$ of the class of $\mathbf{x}$ knowing its community c. Thus, we can compute the cross-entropy of a classifier by considering the communities c instead of the examples $\mathbf{x}$. A few-shot classifier is assumed to minimize $E_{c\in\mathcal{C}}(\text{CE}(P(Y|C = c), P''(Y|C = c))$. Given the entropy H, we know that:

$$\text{CE}(P(Y|C = c), P''(Y|C = c) \geq \text{H}(Y|C = c) = -\sum_{y\in\mathcal{Y}} P(\mathbf{y}|c) \log_2 P(\mathbf{y}|c) \ . \tag{51}$$

Thus, a lower bound on the expected cross-entropy of a few-shot classifier is:

$$\text{E}_{c\in\mathcal{C}}(\text{H}(Y|C = c)) \ . \tag{52}$$

The measure of overlap between two classes $\mathbf{y}_1$ and $\mathbf{y}_2$ reported in this manuscript is an empirical estimation of this lower bound on the cross-entropy of a few-shot classifier. Formally,

$$\begin{aligned}\text{E}_{c\in\mathcal{C}}(\text{H}(P(Y|C = c))) \approx &-\sum_{c\in\mathcal{C}} P(c)(P(\mathbf{y}_1|c) \log_2 P(\mathbf{y}_1|c) + P(\mathbf{y}_2|c) \log_2 P(\mathbf{y}_2|c)) \\ &\text{where } P(\mathbf{y}_1|c)) = \frac{\text{number of examples of class } \mathbf{y}_1 \text{ in } c}{\text{number of examples in } c} = 1 - P(\mathbf{y}_2|c)) \\ &\text{and } P(c) = \frac{\text{number of examples in c}}{\text{total number of examples}} \ .\end{aligned} \tag{53}$$

Finally, we take into account the randomness of the communities created by Louvain's algorithm by reporting a measure of overlap averaged over 5 runs:

$$\text{overlap}(\mathbf{y}_1, \mathbf{y}_2) = \frac{1}{5}\sum_{r=1}^{5} \text{E}_{c\in\mathcal{C}}(\text{H}(P(Y|C = c))) \ . \tag{54}$$

Bibliography

[Abr+14] Alexandre Abraham, Fabian Pedregosa, Michael Eickenberg, Philippe Gervais, Andreas Mueller, Jean Kossaifi, Alexandre Gramfort, Bertrand Thirion, and Gael Varoquaux. "Machine learning for neuroimaging with scikit-learn". In: *Frontiers in Neuroinformatics* 8 (2014), p. 14. ISSN: 1662-5196. DOI: 10.3389/fninf.2014.00014. URL: https://www.frontiersin.org/article/10.3389/fninf.2014.00014.

[Abr+20] Anees Abrol, Zening Fu, Mustafa Salman, Rogers Silva, Yuhui Du, Sergey Plis, and Vince Calhoun. "Hype versus hope: Deep learning encodes more predictive and robust brain imaging representations than standard machine learning". In: *bioRxiv* (2020).

[AES19] Antreas Antoniou, Harrison Edwards, and Amos J. Storkey. "How to train your MAML". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=HJGven05Y7.

[AVT16] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. "Soundnet: Learning sound representations from unlabeled video". In: *Advances in neural information processing systems* 29 (2016), pp. 892–900.

[Bar+19] Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. "Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks". In: *The Journal of Machine Learning Research* 20.1 (2019), pp. 2285–2301.

[BBG21] Myriam Bontonou, Louis Béthune, and Vincent Gripon. "Predicting the Generalization Ability of a Few-Shot Classifier". In: *Information* 12.1 (2021), p. 29.

[Ben+07] Michael Bensch, Ahmed A. Karim, Jürgen Mellinger, Thilo Hinterberger, Michael Tangermann, Martin Bogdan, Wolfgang Rosenstiel, and Niels Birbaumer. "Nessi: An EEG-Controlled Web Browser for Severely Paralyzed Patients". In: *Comput. Intell. Neurosci.* 2007 (2007). DOI: 10.1155/2007/71863. URL: https://doi.org/10.1155/2007/71863.

[Ber+21] Liane Bernstein, Alexander Sludds, Ryan Hamerly, Vivienne Sze, Joel Emer, and Dirk Englund. "Freely scalable and reconfigurable optical hardware for deep learning". In: *Scientific reports* 11.1 (2021), pp. 1–12.

[Bey+19] Lucas Beyer, Xiaohua Zhai, Avital Oliver, and Alexander Kolesnikov. "S4L: Self-Supervised Semi-Supervised Learning". In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 1476–1485. DOI: 10.1109/ICCV.2019.00156. URL: https://doi.org/10.1109/ICCV.2019.00156.

[BFG21] Myriam Bontonou, Nicolas Farrugia, and Vincent Gripon. "Graph-LDA: Graph Structure Priors to Improve the Accuracy in Few-Shot Classification". In: *arXiv preprint arXiv:2108.10427* (2021).

[Bis06] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[Bla+10] Benjamin Blankertz, Michael Tangermann, Carmen Vidaurre, Siamac Fazli, Claudia Sannelli, Stefan Haufe, Cecilia Maeder, Lenny Ramsey, Irene Sturm, Gabriel Curio, and Klaus Mueller. "The Berlin Brain–Computer Interface: Non-Medical Uses of BCI Technology". In: *Frontiers in Neuroscience* 4 (2010), p. 198. ISSN: 1662-453X. DOI: 10.3389/fnins.2010.00198. URL: https://www.frontiersin.org/article/10.3389/fnins.2010.00198.

[Bo+21] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. "Beyond Low-frequency Information in Graph Convolutional Networks". In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 3950–3957. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16514.

[Bon+19a] Myriam Bontonou, Carlos Lassance, Vincent Gripon, and Nicolas Farrugia. "Comparing linear structure-based and data-driven latent spatial representations for sequence prediction". In: *Wavelets and Sparsity XVIII*. Vol. 11138. International Society for Optics and Photonics. 2019, 111380Z.

[Bon+19b] Myriam Bontonou, Carlos Lassance, Ghouthi Boukli Hacene, Vincent Gripon, Jian Tang, and Antonio Ortega. "Introducing graph smoothness loss for training deep learning architectures". In: *2019 IEEE Data Science Workshop (DSW)*. IEEE. 2019, pp. 160–164.

[Bon+19c] Myriam Bontonou, Carlos Lassance, Jean-Charles Vialatte, and Vincent Gripon. "A unified deep learning formalism for processing graph signals". In: *arXiv preprint arXiv:1905.00496* (2019).

[Bon+20a] Myriam Bontonou, Carlos Lassance, Jean-Charles Vialatte, and Vincent Gripon. "Un modèle unifié pour la classification de signaux sur graphe avec de l'apprentissage profond". In: *GRETSI*. 2020.

[Bon+20b] Myriam Bontonou, Giulia Lioi, Nicolas Farrugia, and Vincent Gripon. "Few-shot Learning for Decoding Brain Signals". In: *arXiv preprint arXiv:2010.12500* (2020).

[Bro09] Korbinian Brodmann. *Vergleichende Lokalisationslehre der Grosshirnrinde in ihren Prinzipien dargestellt auf Grund des Zellenbaues*. Barth, 1909.

[Bro+17] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. "Geometric Deep Learning: Going beyond Euclidean data". In: *IEEE Signal Process. Mag.* 34.4 (2017), pp. 18–42. DOI: 10.1109/MSP.2017.2693418. URL: https://doi.org/10.1109/MSP.2017.2693418.

[Bro+20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[Cat21] Grégoire Cattan. "The Use of Brain–Computer Interfaces in Games Is Not Ready for the General Public". In: *Frontiers in Computer Science* 3 (2021), p. 20. ISSN: 2624-9898. DOI: 10.3389/fcomp.2021.628773. URL: https://www.frontiersin.org/article/10.3389/fcomp.2021.628773.

[CD16] Jessica R Cohen and Mark D'Esposito. "The segregation and integration of distinct brain networks and their relationship to cognition". In: *Journal of Neuroscience* 36.48 (2016), pp. 12083–12094.

[Cha+21] Clément Chadebec, Elina Thibeau-Sutre, Ninon Burgos, and Stéphanie Allassonnière. "Data Augmentation in High Dimensional Low Sample Size Setting Using a Geometry-Based Variational Autoencoder". In: *arXiv preprint arXiv:2105.00026* (2021).

[Che+10] Yilun Chen, Ami Wiesel, Yonina C Eldar, and Alfred O Hero. "Shrinkage algorithms for MMSE covariance estimation". In: *IEEE Transactions on Signal Processing* 58.10 (2010), pp. 5016–5029.

[Che+19] Zitian Chen, Yanwei Fu, Yu-Xiong Wang, Lin Ma, Wei Liu, and Martial Hebert. "Image deformation meta-networks for one-shot learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8680–8689.

[Che+20] Mark Cheung, John Shi, Oren Wright, Lavendar Y. Jiang, Xujin Liu, and José M. F. Moura. "Graph Signal Processing and Deep Learning: Convolution, Pooling, and Topology". In: *IEEE Signal Process. Mag.* 37.6 (2020), pp. 139–149. DOI: 10.1109/MSP.2020.3014594. URL: https://doi.org/10.1109/MSP.2020.3014594.

[Col21] Florian François Colombo. *Learning music composition with recurrent neural networks*. Tech. rep. EPFL, 2021.

[DB79] David L Davies and Donald W Bouldin. "A cluster separation measure". In: *IEEE transactions on pattern analysis and machine intelligence* PAMI-1.2 (1979), pp. 224–227.

[DBV16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016,*

Barcelona, Spain. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett. 2016, pp. 3837–3845.

[DBY19] Nima Dehmamy, Albert-László Barabási, and Rose Yu. "Understanding the Representation Power of Graph Neural Networks in Learning Graph Topology". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 15387–15397.

[Def+17] Michaël Defferrard, Lionel Martin, Rodrigo Pena, and Nathanaël Perraudin. *PyGSP: Graph Signal Processing in Python*. Version v0.5.0. Oct. 2017. DOI: 10.5281/zenodo.1003158. URL: https://doi.org/10.5281/zenodo.1003158.

[DH+06] Richard O Duda, Peter E Hart, et al. *Pattern classification*. John Wiley & Sons, 2006.

[Dha20] Payal Dhar. "The carbon impact of artificial intelligence". In: *Nature Machine Intelligence* 2.8 (2020), pp. 423–425.

[DM19] Zsolt Domozi and Andras Molnar. "Surveying private pools in suburban areas with neural network based on drone photos". In: *IEEE EUROCON 2019-18th International Conference on Smart Technologies*. IEEE. 2019, pp. 1–6.

[Dos+14] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin A. Riedmiller, and Thomas Brox. "Discriminative Unsupervised Feature Learning with Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. Ed. by Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger. 2014, pp. 766–774. URL: https://proceedings.neurips.cc/paper/2014/hash/07563a3fe3bbe7e3ba84431ad9d055af-Abstract.html.

[DR17] Gintare Karolina Dziugaite and Daniel M. Roy. "Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data". In: *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. Ed. by Gal Elidan, Kristian Kersting, and Alexander T. Ihler. AUAI Press, 2017. URL: http://auai.org/uai2017/proceedings/papers/173.pdf.

[Est+19] Oscar Esteban, Christopher J Markiewicz, Ross W Blair, Craig A Moodie, A Ilkay Isik, Asier Erramuzpe, James D Kent, Mathias Goncalves, Elizabeth DuPre, Madeleine Snyder, et al. "fMRIPrep: a robust preprocessing pipeline for functional MRI". In: *Nature methods* 16.1 (2019), pp. 111–116.

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1126–1135.

[Fed21] Evelina Fedorenko. "The early origins and the growing popularity of the individual-subject analytic approach in human neuroscience". In: *Current Opinion in Behavioral Sciences* 40 (2021), pp. 105–112.

[FHT+01] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York, 2001.

[Fon+09] VS Fonov, AC Evans, RC McKinstry, CR Almli, and DL Collins. "Unbiased nonlinear average age-appropriate brain templates from birth to adulthood". In: *NeuroImage* 47 (2009). Organization for Human Brain Mapping 2009 Annual Meeting, S102. ISSN: 1053-8119. DOI: https://doi.org/10.1016/S1053-8119(09)70884-5.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[GEB15] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. "A neural algorithm of artistic style". In: *arXiv preprint arXiv:1508.06576* (2015).

[GL18] Henry Gray and WH Lewis. "Anatomy of the human body. Lea & Febiger". In: *Philadephia, PA* (1918).

[Gla+16] Matthew F Glasser, Timothy S Coalson, Emma C Robinson, Carl D Hacker, John Harwell, Essa Yacoub, Kamil Ugurbil, Jesper Andersson, Christian F Beckmann, Mark Jenkinson, et al. "A multi-modal parcellation of human cerebral cortex". In: *Nature* 536.7615 (2016), pp. 171–178.

[Gor+15] Krzysztof J Gorgolewski, Gael Varoquaux, Gabriel Rivera, Yannick Schwarz, Satrajit S Ghosh, Camille Maumet, Vanessa V Sochat, Thomas E Nichols, Russell A Poldrack, Jean-Baptiste Poline, et al. "NeuroVault. org: a web-based repository for collecting and sharing unthresholded statistical maps of the human brain". In: *Frontiers in neuroinformatics* 9 (2015), p. 8.

[Gor+17a] Evan M Gordon, Timothy O Laumann, Adrian W Gilmore, Dillan J Newbold, Deanna J Greene, Jeffrey J Berg, Mario Ortega, Catherine Hoyt-Drazen, Caterina Gratton, Haoxin Sun, et al. "Precision functional mapping of individual human brains". In: *Neuron* 95.4 (2017), pp. 791–807.

[Gor+17b] Krzysztof Gorgolewski, Oscar Esteban, Gunnar Schaefer, Brian Wandell, and Russell Poldrack. "OpenNeuro—a free online platform for sharing and analysis of neuroimaging data". In: *Organization for human brain mapping. Vancouver, Canada* 1677.2 (2017).

[GSK18] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. "Unsupervised Representation Learning by Predicting Image Rotations". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=S1v4N2l0-.

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6572.

[Gue+21] Nitzan Guetta, Asaf Shabtai, Inderjeet Singh, Satoru Momiyama, and Yuval Elovici. "Dodging Attack Using Carefully Crafted Natural Makeup". In: *arXiv preprint arXiv:2109.06467* (2021).

[Hag+08] Patric Hagmann, Leila Cammoun, Xavier Gigandet, Reto Meuli, Christopher J Honey, Van J Wedeen, and Olaf Sporns. "Mapping the structural core of human cerebral cortex". In: *PLoS biology* 6.7 (2008), e159.

[Hax+01] James V Haxby, M Ida Gobbini, Maura L Furey, Alumit Ishai, Jennifer L Schouten, and Pietro Pietrini. "Distributed and overlapping representations of faces and objects in ventral temporal cortex". In: *Science* 293.5539 (2001), pp. 2425–2430.

[Hax12] James V Haxby. "Multivariate pattern analysis of fMRI: the early beginnings". In: *Neuroimage* 62.2 (2012), pp. 852–855.

[HB18] Martin N Hebart and Chris I Baker. "Deconstructing multivariate decoding for the study of brain function". In: *Neuroimage* 180 (2018), pp. 4–18.

[HBL15] Mikael Henaff, Joan Bruna, and Yann LeCun. "Deep Convolutional Networks on Graph-Structured Data". In: *CoRR* abs/1506.05163 (2015). arXiv: 1506.05163. URL: http://arxiv.org/abs/1506.05163.

[He+15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[Hen+18] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. "Deep reinforcement learning that matters". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[HGP21] Yuqing Hu, Vincent Gripon, and Stéphane Pateux. "Graph-based interpolation of feature vectors for accurate few-shot classification". In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 8164–8171.

[HH09] Suzana Herculano-Houzel. "The human brain in numbers: a linearly scaled-up primate brain". In: *Frontiers in Human Neuroscience* 3 (2009), p. 31. ISSN: 1662-5161. DOI: 10.3389/neuro.09.031.2009. URL: https://www.frontiersin.org/article/10.3389/neuro.09.031.2009.

[HM16] Ruining He and Julian McAuley. "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering". In: *proceedings of the 25th international conference on world wide web*. 2016, pp. 507–517.

[Hoc+12] Leigh R Hochberg, Daniel Bacher, Beata Jarosiewicz, Nicolas Y Masse, John D Simeral, Joern Vogel, Sami Haddadin, Jie Liu, Sydney S Cash, Patrick Van Der Smagt, et al. "Reach and grasp by people with tetraplegia using a neurally controlled robotic arm". In: *Nature* 485.7398 (2012), pp. 372–375.

[Jia+19] Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. "Predicting the Generalization Gap in Deep Networks with Margin Distributions". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=HJlQfnCqKX.

[Jia+20a] Yiding Jiang, Pierre Foret, Scott Yak, Daniel M Roy, Hossein Mobahi, Gintare Karolina Dziugaite, Samy Bengio, Suriya Gunasekar, Isabelle Guyon, and Behnam Neyshabur. "Neurips 2020 competition: Predicting generalization in deep learning". In: *arXiv preprint arXiv:2012.07976* (2020).

[Jia+20b] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. "Fantastic Generalization Measures and Where to Find Them". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=SJgIPJBFvH.

[Jum+21] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (2021), pp. 583–589.

[Kan+00] Eric R Kandel, James H Schwartz, Thomas M Jessell, Steven Siegelbaum, A James Hudspeth, and Sarah Mack. *Principles of neural science*. Vol. 4. McGraw-hill New York, 2000.

[KB15] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[Ken38] Maurice G Kendall. "A new measure of rank correlation". In: *Biometrika* 30.1/2 (1938), pp. 81–93.

[Kes+17] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[KMC97] Nancy Kanwisher, Josh McDermott, and Marvin M Chun. "The fusiform face area: a module in human extrastriate cortex specialized for face perception". In: *Journal of neuroscience* 17.11 (1997), pp. 4302–4311.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[KW17] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=SJU4ayYgl.

[Las+20a] Carlos Lassance, Louis Béthune, Myriam Bontonou, Mounia Hamidouche, and Vincent Gripon. "Ranking Deep Learning Generalization using Label Variation in Latent Geometry Graphs". In: *arXiv preprint arXiv:2011.12737* (2020).

[Las+20b] Carlos Lassance, Myriam Bontonou, Ghouthi Boukli Hacene, Vincent Gripon, Jian Tang, and Antonio Ortega. "Deep geometric knowledge distillation with graphs". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 8484–8488.

[LB03] Denis Le Bihan. "Looking into the functional architecture of the brain with diffusion MRI". In: *Nature reviews neuroscience* 4.6 (2003), pp. 469–480.

[LeC+98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[LGO21] Carlos Lassance, Vincent Gripon, and Antonio Ortega. "Representing deep neural networks latent space geometries with graphs". In: *Algorithms* 14.2 (2021), p. 39.

[Li+19] Xiaoxiao Li, Nicha C Dvornek, Yuan Zhou, Juntang Zhuang, Pamela Ventola, and James S Duncan. "Graph neural network for interpreting task-fmri biomarkers". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2019, pp. 485–493.

[Li+20] Xiaoxiao Li, Sheng Bi, Yongxing Wang, Min Dong, and Zhangshao Chen. "A Few-shot Dynamic Obstacle Avoidance Strategy in Unknown Environments". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–7.

[Liu+20] Yuan Liu, Ayush Jain, Clara Eng, David H Way, Kang Lee, Peggy Bui, Kimberly Kanada, Guilherme de Oliveira Marinho, Jessica Gallegos, Sara Gabriele, et al. "A deep learning system for differential diagnosis of skin diseases". In: *Nature medicine* 26.6 (2020), pp. 900–908.

[LW04] Olivier Ledoit and Michael Wolf. "Honey, I shrunk the sample covariance matrix". In: *The Journal of Portfolio Management* 30.4 (2004), pp. 110–119.

[Man+20] Puneet Mangla, Nupur Kumari, Abhishek Sinha, Mayank Singh, Balaji Krishnamurthy, and Vineeth N Balasubramanian. "Charting the right manifold: Manifold mixup for few-shot learning". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, pp. 2218–2227.

[McA99] David A McAllester. "PAC-Bayesian model averaging". In: *Proceedings of the twelfth annual conference on Computational learning theory*. 1999, pp. 164–170.

[McR+17] Donald W McRobbie, Elizabeth A Moore, Martin J Graves, and Martin R Prince. *MRI from Picture to Proton*. Cambridge university press, 2017.

[Men+21] Arthur Mensch, Julien Mairal, Bertrand Thirion, and Gaël Varoquaux. "Extracting representations of cognition across neuroimaging studies improves brain decoding". In: *PLoS computational biology* 17.5 (2021), e1008795.

[Mil95] George A Miller. "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[Mon11] Martin M Monti. "Statistical analysis of fMRI time-series: a critical review of the GLM approach". In: *Frontiers in human neuroscience* 5 (2011), p. 28.

[Nie15] Michael A Nielsen. *Neural networks and deep learning*. Vol. 25. Determination press San Francisco, CA, 2015.

[OEJ01] AM Owen, R Epstein, and IS Johnsrude. "Functional Magnetic Resonance Imaging. An Introduction to Methods". In: *fMRI: Applications to Cognitive Neuroscience*. Oxford Univ. Press, 2001.

[Pap+17] Dimitri Papadopoulos Orfanos, Vincent Michel, Yannick Schwartz, Philippe Pinel, Antonio Moreno, Denis Le Bihan, and Vincent Frouin. "The Brainomics/Localizer database". In: *NeuroImage* 144 (2017). Data Sharing Part II, pp. 309–314. ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2015.09.052. URL: https://www.sciencedirect.com/science/article/pii/S1053811915008745.

[Pas+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 8024–8035. URL: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

[PCZ+98] Victor Y Pan, Z Chen, Ailong Zheng, et al. "The complexity of the algebraic eigenproblem". In: *Mathematical Sciences Research Institute, Berkeley* (1998), pp. 1998–71.

[Ped+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[PGB15] Julien Perret, Maurizio Gribaudi, and Marc Barthelemy. "Roads and cities of 18th century France". In: *Scientific Data* 2 (2015), p. 150048. DOI: 10.1038/sdata.2015.48. URL: https://doi.org/10.1038/sdata.2015.48.

[Pin+07] Philippe Pinel, Bertrand Thirion, Sébastien Meriaux, Antoinette Jobert, Julien Serres, Denis Le Bihan, Jean-Baptiste Poline, and Stanislas Dehaene. "Fast reproducible identification and large-scale databasing of individual functional cognitive networks". In: *BMC neuroscience* 8.1 (2007), pp. 1–18.

[Pin+18] Ana Luísa Pinho, Alexis Amadon, Torsten Ruest, Murielle Fabre, Elvis Dohmatob, Isabelle Denghien, Chantal Ginisty, Séverine Becuwe-Desmidt, Séverine Roger, Laurence Laurier, et al. "Individual Brain Charting, a high-resolution fMRI dataset for cognitive mapping". In: *Scientific data* 5.1 (2018), pp. 1–15.

[Pin+20] Ana Luísa Pinho, Alexis Amadon, Baptiste Gauthier, Nicolas Clairis, André Knops, Sarah Genon, Elvis Dohmatob, Juan Jesús Torre, Chantal Ginisty, Séverine Becuwe-Desmidt, et al. "Individual Brain Charting dataset extension, second release of high-resolution fMRI data for cognitive mapping". In: *Scientific Data* 7.1 (2020), pp. 1–16.

[Ple19] Petr Plechác. "Relative contributions of Shakespeare and Fletcher in Henry VIII: An Analysis Based on Most Frequent Words and Most Frequent Rhythmic Patterns". In: *CoRR* abs/1911.05652 (2019). arXiv: 1911.05652. URL: http://arxiv.org/abs/1911.05652.

[Pol06] Russell A. Poldrack. "Can cognitive processes be inferred from neuroimaging data?" In: *Trends in Cognitive Sciences* 10.2 (2006), pp. 59–63. ISSN: 1364-6613. DOI: https://doi.org/10.1016/j.tics.2005.12.004. URL: https://www.sciencedirect.com/science/article/pii/S1364661305003360.

[PVDV19] Maria Giulia Preti and Dimitri Van De Ville. "Decoupling of brain function from structure reveals regional behavioral specialization in humans". In: *Nature communications* 10.1 (2019), pp. 1–7.

[Qia99] Ning Qian. "On the momentum term in gradient descent learning algorithms". In: *Neural networks* 12.1 (1999), pp. 145–151.

[RDB18] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. "Artistic Style Transfer for Videos and Spherical Images". In: *Int. J. Comput. Vis.* 126.11 (2018), pp. 1199–1219. DOI: 10.1007/s11263-018-1089-z. URL: https://doi.org/10.1007/s11263-018-1089-z.

[Reb+10] Brice Rebsamen, Cuntai Guan, Haihong Zhang, Chuanchu Wang, Cheeleong Teo, Marcelo H Ang, and Etienne Burdet. "A brain controlled wheelchair to navigate in familiar environments". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 18.6 (2010), pp. 590–598.

[Ren+18] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. "Meta-Learning for Semi-Supervised Few-Shot Classification". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=HJcSzz-CZ.

[RJS17] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. "Learning to generate reviews and discovering sentiment". In: *arXiv preprint arXiv:1704.01444* (2017).

[Rus+08] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. "LabelMe: a database and web-based tool for image annotation". In: *International journal of computer vision* 77.1-3 (2008), pp. 157–173.

[Rus+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. "ImageNet Large Scale Visual Recognition Challenge". In: *Int. J. Comput. Vis.* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y. URL: https://doi.org/10.1007/s11263-015-0816-y.

[SCA20] Othman Sbai, Camille Couprie, and Mathieu Aubry. "Impact of Base Dataset Design on Few-Shot Image Classification". In: *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XVI*. Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. Vol. 12361. Lecture Notes in Computer Science. Springer, 2020, pp. 597–613. DOI: 10.1007/978-3-030-58517-4_35. URL: https://doi.org/10.1007/978-3-030-58517-4_35.

[Sch+19] Victor Schmidt, Alexandra Luccioni, S. Karthik Mukkavilli, Narmada Balasooriya, Kris Sankaran, Jennifer T. Chayes, and Yoshua Bengio. "Visualizing the Consequences of Climate Change Using Cycle-Consistent Adversarial Networks". In: *CoRR* abs/1905.03709 (2019). arXiv: 1905.03709. URL: http://arxiv.org/abs/1905.03709.

[Sch+20] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. "Mastering atari, go, chess and shogi by planning with a learned model". In: *Nature* 588.7839 (2020), pp. 604–609.

[SCM14] Terrence J Sejnowski, Patricia S Churchland, and J Anthony Movshon. "Putting big data to good use in neuroscience". In: *Nature neuroscience* 17.11 (2014), pp. 1440–1441.

[Seg+16] Santiago Segarra, Gonzalo Mateos, Antonio G Marques, and Alejandro Ribeiro. "Blind identification of graph filters". In: *IEEE Transactions on Signal Processing* 65.5 (2016), pp. 1146–1159.

[Seg+17] Santiago Segarra, Antonio G Marques, Gonzalo Mateos, and Alejandro Ribeiro. "Network topology inference from spectral templates". In: *IEEE Transactions on Signal and Information Processing over Networks* 3.3 (2017), pp. 467–483.

[Shu+13] David I. Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. "The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains". In: *IEEE Signal Process. Mag.* 30.3 (2013), pp. 83–98. DOI: 10.1109/MSP.2012.2235192. URL: https://doi.org/10.1109/MSP.2012.2235192.

[Spr+15] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. "Striving for Simplicity: The All Convolutional Net". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6806.

[Sri+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[STK05] Olaf Sporns, Giulio Tononi, and Rolf Kötter. "The human connectome: a structural description of the human brain". In: *PLoS computational biology* 1.4 (2005), e42.

[TGB18a] Nicolas Tremblay, Paulo Gonçalves, and Pierre Borgnat. "Design of graph filters and filterbanks". In: *Cooperative and Graph Signal Processing*. Elsevier, 2018, pp. 299–324.

[TGB18b] Nicolas Tremblay, Paulo Gonçalves, and Pierre Borgnat. "Chapter 11 - Design of Graph Filters and Filterbanks". In: *Cooperative and Graph Signal Processing*. Ed. by Petar M. Djurić and Cédric Richard. Academic Press, 2018, pp. 299–324. ISBN: 978-0-12-813677-5. DOI: https://doi.org/10.1016/B978-0-12-813677-5.00011-0.

[Thr98] Sebastian Thrun. "Lifelong learning algorithms". In: *Learning to learn*. Springer, 1998, pp. 181–209.

[TL94] Greg Turk and Marc Levoy. "Zippered Polygon Meshes from Range Images". In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '94. New York, NY, USA: Association for Computing Machinery, 1994, 311–318. ISBN: 0897916670. DOI: 10.1145/192161.192241. URL: https://doi.org/10.1145/192161.192241.

[Tse+21] Gabriel Tseng, Hannah Kerner, Catherine Nakalembe, and Inbal Becker-Reshef. "Learning To Predict Crop Type From Heterogeneous Sparse Labels Using Meta-Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 1111–1120.

[VC15] Vladimir N Vapnik and A Ya Chervonenkis. "On the uniform convergence of relative frequencies of events to their probabilities". In: *Measures of complexity*. Springer, 2015, pp. 11–30.

[VE+12] David C Van Essen, Kamil Ugurbil, Edward Auerbach, Deanna Barch, Timothy EJ Behrens, Richard Bucholz, Acer Chang, Liyong Chen, Maurizio Corbetta, Sandra W Curtiss, et al. "The Human Connectome Project: a data acquisition perspective". In: *Neuroimage* 62.4 (2012), pp. 2222–2231.

[Ver+19] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. "Manifold Mixup: Better Representations by Interpolating Hidden States". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 6438–6447. URL: http://proceedings.mlr.press/v97/verma19a.html.

[Vin+16] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. "Matching Networks for One Shot Learning". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett. 2016, pp. 3630–3638.

[VOF20] Clément Vignac, Guillermo Ortiz-Jiménez, and Pascal Frossard. "On The Choice of Graph Neural Network Architectures". In: *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*. IEEE, 2020, pp. 8489–8493. DOI: 10.1109/ICASSP40776.2020.9054357. URL: https://doi.org/10.1109/ICASSP40776.2020.9054357.

[VP19] Gaël Varoquaux and Russell A Poldrack. "Predictive models avoid excessive reductionism in cognitive neuroimaging". In: *Current Opinion in Neurobiology* 55 (2019). Machine Learning, Big Data, and Neuroscience, pp. 1–6. ISSN: 0959-4388. DOI: https://doi.org/10.1016/j.conb.2018.11.002. URL: https://www.sciencedirect.com/science/article/pii/S0959438818301089.

[Wan+19] Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. "Simpleshot: Revisiting nearest-neighbor classification for few-shot learning". In: *arXiv preprint arXiv:1911.04623* (2019).

[Wan+20] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. "Generalizing from a Few Examples: A Survey on Few-shot Learning". In: *ACM Comput. Surv.* 53.3 (2020), 63:1–63:34. DOI: 10.1145/3386252. URL: https://doi.org/10.1145/3386252.

[Wu+19] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. "Simplifying Graph Convolutional Networks". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 6861–6871. URL: http://proceedings.mlr.press/v97/wu19e.html.

[Xu+19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How Powerful are Graph Neural Networks?" In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[Yan+20] Qiang Yang, Yu Zhang, Wenyuan Dai, and Sinno Jialin Pan. *Transfer learning*. Cambridge University Press, 2020.

[Yar+11] Tal Yarkoni, Russell A Poldrack, Thomas E Nichols, David C Van Essen, and Tor D Wager. "Large-scale automated synthesis of human functional neuroimaging data". In: *Nature methods* 8.8 (2011), pp. 665–670.

[ZB20] Yu Zhang and Pierre Bellec. "Transferability of brain decoding using graph convolutional networks". In: *bioRxiv* (2020).

[Zha+21a] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. "Understanding deep learning (still) requires rethinking generalization". In: *Communications of the ACM* 64.3 (2021), pp. 107–115.

[Zha+21b] Yu Zhang, Loïc Tetrel, Bertrand Thirion, and Pierre Bellec. "Functional annotation of human cognitive states using deep graph convolution". In: *NeuroImage* 231 (2021), p. 117847.

[Zha+21c] Yu Zhang, Loïc Tetrel, Bertrand Thirion, and Pierre Bellec. "Functional annotation of human cognitive states using deep graph convolution". In: *NeuroImage* 231 (2021), p. 117847. DOI: 10.1016/j.neuroimage.2021.117847. URL: https://doi.org/10.1016/j.neuroimage.2021.117847.

[ZZK19] Hongguang Zhang, Jing Zhang, and Piotr Koniusz. "Few-Shot Learning via Saliency-Guided Hallucination of Samples". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 2770–2779. DOI: 10.1109/CVPR.2019.00288. URL: http://openaccess.thecvf.com/content_CVPR_2019/html/Zhang_Few-Shot_Learning_via_Saliency-Guided_Hallucination_of_Samples_CVPR_2019_paper.html.

Index

List of Abbreviations

k-NN k Nearest Neighbors. 46

AI Artificial Intelligence. 20–27, 29, 34, 41, 42, 45, 88, 114

BCI Brain-Computer Interface. 88, 96, 104

CNN Convolutional Neural Network. 30, 32, 33, 53, 54, 99, 103, 115, 116, 119

DNN Deep Neural Network. 21–23, 25–33, 35, 37, 40, 49–54, 65, 66, 68, 70, 72–74, 78, 86, 98–101, 103, 109, 114, 115, 117, 121

EEG Electroencephalography. 88, 91

fMRI functional Magnetic Resonance Imaging. 29, 31–35, 88, 91–101, 104, 105, 110, 112

GCN Graph Convolutional Network. 66

GFT Graph Fourier Transform. 62, 64, 65

GLM General Linear Model. 95, 101

GNN Graph Neural Network. 54, 65, 66, 99, 105, 117

GSO Graph-Shift Operator. 62, 64, 65, 106–108

GSP Graph Signal Processing. 28, 33, 37, 61, 62

IBC Individual Brain Charting. 36, 101, 102, 104, 115, 116

LDA Linear Discriminant Analysis. 46, 47, 106–110, 116, 117

LR Logistic Regression. 47, 49, 66, 70, 71, 78, 109, 110, 116, 117, 121–123

MEG Magnetoencephalography. 88, 91

MLP Multilayer Perceptron. 21, 22, 25, 51, 52, 65, 103, 105, 111, 116, 119

MRI Magnetic Resonance Imaging. 35, 92, 93, 99, 104, 112

NCM Nearest Class Mean. 46, 47, 71, 103, 107–111, 116, 117

ROI Region Of Interest. 93, 102

SGC Simplifying Graph Convolutional Networks. 66, 105, 109, 111, 123

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Exploiter la Structure des Données pour Apprendre à partir de Quelques Exemples : Applications à la Vision Assistée par Ordinateur et à la Neuro-Imagerie

Mots clés : Apprentissage profond, Apprentissage automatique, Traitement des signaux sur graphe, Neurosciences

Résumé : L'objet de cette thèse est d'investiguer certains verrous liés au développement des méthodes d'apprentissage profond, notamment 1) l'apprentissage à partir de peu d'exemples, qui permet de s'adapter à des situations inédites en extrapolant les informations acquises dans un contexte limité et 2) l'apprentissage sur les domaines complexes et abstraits se prêtant mal au formalisme mathématique, en s'intéressant tout particulièrement à l'irrégularité de l'activité cérébrale.

Tout d'abord, nous présentons des méthodes de classification permettant d'apprendre à partir de peu d'exemples en s'inspirant d'un contexte plus riche. Nous mettons en avant la question cruciale de l'évaluation de la qualité de l'apprentissage et nous proposons plusieurs pistes de recherche pour y répondre. Ensuite, nous adaptons ces méthodes à des données de neuroimagerie dont le nombre est limité par le coût des techniques d'acquisition. Le problème est d'autant plus complexe que l'activité cérébrale est structurée par des réseaux de neurones adaptatifs et redondants.

Title : Leveraging Data Structure to Learn from Few Examples: Applications to Computer Vision and Neuroimaging

Keywords : Deep learning, Machine learning, Graph signal processing, Neuroscience

Abstract : The purpose of this thesis is to investigate some of the challenges related to the development of deep learning methods, in particular 1) learning from few examples, which allows to adapt to new situations by extrapolating the information acquired in a limited context, and 2) learning from complex and abstract domains that do not lend themselves well to mathematical formalism, with a particular interest in the irregularity of the brain activity.

First, we present classification methods that allow learning from few examples by drawing their inspiration from a richer context. We highlight the crucial question of how to assess the quality of learning and propose several avenues to address it. Then, we adapt these methods to neuroimaging data whose number is limited by the cost of acquisition techniques. The problem is all the more complex as brain activity is structured by adaptive and redundant neural networks.