Beyond the training task in classification : looking at extension of the notion of generalization

Raphaël Baena

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

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

ECOLE DOCTORALE N°648
Sciences pour l'ingénieur et le numérique
Spécialité : Signal, Image et Vision

Par
Raphaël BAENA

Beyond the training task in classification: looking at extension of the notion of generalization

Thèse présentée et soutenue à IMT Atlantique, Brest, le 01/12/23
Unité de recherche : Lab-STICC, CNRS UMR 6285
Thèse N° : 2023IMTA0372

Rapporteurs avant soutenance :

Nicolas Courty Professeur des universités, Université Bretagne Sud
David Picard Directeur de recherche, École des Ponts ParisTech

Composition du Jury :

Président :	Pierre Borgnat	Directeur de recherche, ENS Lyon
Examinateurs :	Nicolas Courty	Professeur des universités, Université Bretagne Sud
	David Picard	Directeur de recherche, École des Ponts ParisTech
	Claire Boyer	Maître de Conférence, (LPSM) Sorbonne Universit é
Dir. de thèse :	Vincent Gripon	Directeur de recherche, IMT Atlantique
Co-encadrant :	Lucas Drumetz	Maître de conférence, IMT Atlantique

Contents

Abstract

The thesis focuses on the concept of generalization, particularly in the context of supervised machine learning classification. This approach involves learning to solve a task (classification) based on labeled training data. Generalization is defined as the ability to make accurate predictions on unseen data during training. Traditionally, generalization is limited to data that belongs to the same domain as the training task.

However, recent literature highlights the capability of deep learning architectures to generalize beyond their training task. Thus, a model trained on a specific task can be partially reused for other tasks.

The thesis explores various possible extensions of generalization, including learning on a set of classes and the ability to generalize to a larger set of classes, learning on coarse labels to predict more complex labels, learning on artificially complex tasks to improve generalization, and learning invariant operators for a specific task.

French

La thèse s'intéresse à la notion de généralisation, en particulier dans le cadre de la classification en apprentissage automatique de manière supervisée. Cette approche consiste à apprendre à résoudre une tâche (classification) à partir de données d'entrainement étiquetées. La généralisation est définie comme la capacité à réaliser des prédictions correctes sur des données non observées pendant l'entraînement. Cette notion est généralement restreinte à des données qui correspondent au même domaine que celui de la tâche d'entrainement.

Cependant, une littérature récente met en exergue la capacité des architectures d'apprentissage profond à généraliser en dehors de leur tâche d'entrainement. Ainsi, un modèle entrainé sur une tâche particulière peut être réutilisée en partie sur d'autres tâches.

Ainsi, nous explorons différentes extensions possibles de la généralisation : apprentissage sur un ensemble de classe et l'habilité à généraliser sur un ensemble de classes plus grands, l'apprentissage sur des labels grossiers pour prédire des labels plus complexes, l'apprentissage sur une tâche artificiellement complexe pour améliorer la généralisation ou encore l'apprentissage d'opérateurs invariants pour résoudre une tâche spécifique.

Résumé Long *(french summary)*

Introduction

De la compréhension du langage naturel à la reconnaissance d'images, l'intelligence artificielle (IA) est parvenue à résoudre des tâches très complexes qui étaient jusqu'à récemment considérées comme impossibles pour des machines (DEVLIN et al., 2018 ; RADFORD et al., 2021 ; RADFORD et al., 2018). L'une des capacités impressionnantes de l'IA est sa capacité à apprendre et à reconnaître des motifs par elle-même à partir de larges quantités de données, puis à réaliser des prédictions précises sur de nouvelles données. En apprentissage automatique, cette capacité est classiquement appelée généralisation (GOODFELLOW et al., 2016).

À l'origine, la notion de généralisation se limitait aux prédictions effectuées sur des données similaires à celles utilisées pour entraîner le modèle d'IA (CARUANA, 1997 ; ZHUANG et al., 2020a). Cependant, on peut constater dans la littérature récente en rapport à l'apprentissage par transfert que la notion de généralisation a été étendue largement au-delà de son contexte d'origine (CARON et al., 2020 ; FEI-FEI et al., 2004 ; KRIZHEVSKY et al., 2017 ; PAN & YANG, 2010 ; RADFORD et al., 2015). L'apprentissage par transfert permet en effet aux systèmes d'IA de transférer les connaissances et représentations apprises vers de nouveaux domaines et des tâches différentes. La généralisation englobe alors la capacité à prendre des décisions, non plus uniquement sur des données non observées, mais également sur de nouvelles tâches.

Cependant, malgré ces grandes avancées, le domaine de l'apprentissage par transfert présente encore certaines limites et plusieurs questions fondamentales restent sans réponse, typiquement en ce qui concerne les procédures d'apprentissage qui conduisent à une généralisation optimale dans un contexte de transfert et les mécanismes sous-jacents (HUH et al., 2016 ; JANOCHA & CZARNECKI, 2017a ; TIAN et al., 2020a ; YOSINSKI et al., 2014). Dans ce manuscrit, nous envisageons plusieurs extensions de la notion de généralisation et imaginons plusieurs scénarios pour aborder ces questions.

Intelligence Artificielle et Apprentissage Profond

La démocratisation de l'apprentissage automatique a été rendue possible par l'accès à de grandes quantités de données (appelées "big data") et à des ressources informatiques puissantes et moins coûteuses, en particulier les unités de traitement graphique (GPU), au début du XXIe siècle (BOMMASANI et al., 2021). Dans ce contexte particulier, l'apprentissage profond a émergé comme une approche dominante. L'une des percées les plus significatives dans l'apprentissage profond a été le développement des réseaux neuronaux convolutifs (CNN)(KRIZHEVSKY et al., 2017 ; LECUN et al., 1998), qui ont révolutionné la vision par ordinateur. Les CNN sont capables d'extraire des caractéristiques d'images selon plusieurs échelles et emplacements, les rendant très efficaces pour des tâches telles que la détection d'objets(REDMON et al., 2016), la classification (S.

REN et al., 2015), la segmentation d'images (MINAEE et al., 2021). Le succès des CNN a très vite conduit au développement de modèles d'apprentissage profond très performants pour des problèmes de vision par ordinateur (HE et al., 2016 ; WIGHTMAN et al., 2021).

Apprentissage Supervisé et par Transfert

L'apprentissage supervisé est une approche couramment employée pour résoudre de nombreux problèmes (GOODFELLOW et al., 2016). Cependant, l'une des principales limites de cette approche est qu'elle a tendance à construire souvent un modèle trop spécifique à la tâche d'entraînement (RADFORD et al., 2015). En général, cela implique que si la tâche d'application est modifiée, la performance du modèle peut être gravement impactée, nécessitant parfois de devoir ré-entraîner complètement le modèle. Récemment, des chercheurs ont développé des approches d'apprentissage dites par transfert qui visent à exploiter les connaissances acquises sur une tâche d'entraînement pour les appliquer sur de nouvelles tâches (PAN & YANG, 2010 ; ZHUANG et al., 2020a). Ces approches peuvent être particulièrement utiles dans des situations où, pour une nouvelle tâche, il y a peu de données étiquetées disponibles.

Initialement, l'apprentissage par transfert se réduisait principalement à l'adaptation de domaine (CARUANA, 1997 ; M. WANG & DENG, 2018), où la tâche cible restait inchangée, mais le domaine des données (ajouts de bruits, luminosité différente) était modifié. Avec le temps, le domaine de l'apprentissage par transfert a fortement évolué, ouvrant de nouvelles possibilités permettant la réutilisation de modèles pré-entraînés sur des tâches autrefois considérées comme hautement spécialisées et difficiles (JING & TIAN, 2020 ; PAN & YANG, 2010 ; ZHUANG et al., 2020a).

Cette avancée a été particulièrement marquante dans plusieurs domaines : imagerie médicale (MAGOULAS & PRENTZA, 1999), compréhension du langage naturel (DEVLIN et al., 2018 ; RADFORD et al., 2018), industrie automobile (BOJARSKI et al., 2016), etc. Cependant, ces approches présentent également leurs propres défis. En effet, il existe de nombreux paramètres qui peuvent influencer les représentations apprises et leur capacité à être réutilisées sur différentes tâches : les critères utilisés pour l'apprentissage, les régularisations, le nombre de paramètres à optimiser, l'architecture et la finesse des informations disponibles pour s'entraîner (par exemple, étiquetage hiérarchiques, sémantiques, multi-classes)(HUH et al., 2016 ; TIAN et al., 2020a ; WIGHTMAN et al., 2021 ; WU et al., 2017). Il n'est donc pas aisé de déterminer quelles procédures d'apprentissage conduisent à la plus grande transférabilité et quelles pourraient être leurs éventuelles biais et limitations(D'ASCOLI et al., 2021).

Ainsi, dans cette thèse, nous nous sommes concentrés sur plusieurs extensions de la notion de généralisation qui illustrent la capacité des architectures d'apprentissage profond à généraliser au-delà de leurs tâches d'entraînement, notamment dans le cadre de la classification. Nous avons envisagé plusieurs scénarios : de la classification d'étiquettes grossières à raffinées, d'un ensemble de classes à un autre, de tâches artificiellement complexifiées à des tâches plus simples.

Résumé des Contributions

Apprentissage d'Opérateurs Invariants sur des Domaines Irréguliers pour des Tâches de Classification

Les couches et les opérations utilisées pour construire un modèle d'apprentissage profond impactent profondément les performances ainsi que les propriétés des caractéristiques apprises par le modèle, en particulier en termes d'invariance et d'équivariances aux transformations géométriques. Dans le cadre de la vision par ordinateur, l'utilisation de réseaux de neurones convolutifs (CNN) a permis d'obtenir des gains significatifs de précision grâce à des représentations invariantes aux translations et aux faibles perturbations (LeCun et al., 1998 ; Mallat, 2016).

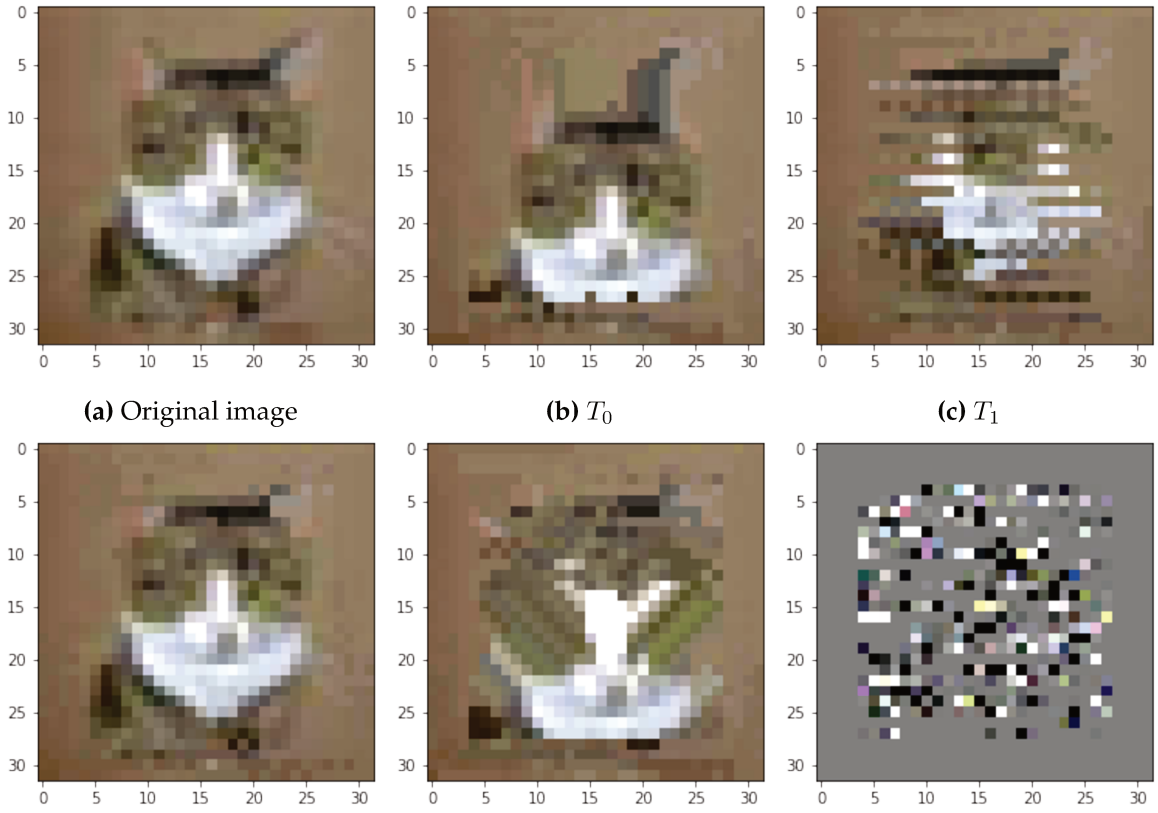
Cependant, s'il est aisé d'opérer des convolutions sur des domaines réguliers comme des images, définir de telles opérations sur des domaines abstraits, tels que les graphes, devient considérablement plus complexe. En effet, par définition, la notion de direction n'est pas explicite sur des graphes, alors que cette notion est centrale pour parvenir à définir des translations, puis des convolutions.

En traitement du signal sur graphe, plusieurs chercheurs ont proposé des définitions analogues (graphe anneau) à celles du traitement du signal, pour la transformée de Fourier et les translations sur graphe (Ortega, Frossard, Kovačević, Moura et al., 2018). L'approche classique pour obtenir des translations consiste à partir de la structure du graphe, par exemple en utilisant son Laplacien (Ortega, Frossard, Kovačević, Moura et al., 2018). Cependant, le problème de ces approches est qu'elles sont complètement agnostiques du signal porté par le graphe et conduisent à des opérateurs isotropes (Monti et al., 2018 ; Pasdeloup et al., 2018)

Dans le chapitre II, nous proposons donc de renverser le paradigme des réseaux de neurones convolutifs (CNN), qui combinent des couches de convolutions et de pooling pour obtenir des représentations invariantes aux translations, en définissant les translations comme des opérateurs invariants pour les tâches de classification. Pour cela, nous définissons des pseudo-couches de convolutions à l'aide d'opérateurs apprenables via un mécanisme de partage de poids (weight-sharing). Notre approche produit des architectures qui peuvent être entraînées de bout en bout pour résoudre des tâches de classification. Les opérateurs employés dans les couches de convolutions sont contraints par la structure du graphe et appris en résolvant une tâche prétexte de classification. En fin de compte, nous interprétons les opérateurs appris comme des translations sur graphe.

Cette méthode est d'abord employée sur des images représentées sur des graphes réguliers (grilles). Dans ce cadre, nous avons réussi à retrouver les translations d'images usuelles verticales. De manière plus surprenante, nous avons également appris des opérations de compression et de dilatation horizontales, qui correspondent effectivement à des opérations invariantes dans le cadre de la classification d'images. Nous avons ensuite testé cette approche en remplaçant les grilles par un graphe plus irrégulier construit à partir de la matrice de covariances de ces images. Les résultats sont illustrés sur la Figure 12. Dans ce cadre, nous retrouvons une nouvelle fois les opérations précédentes. Enfin, nous expérimentons sur des réseaux d'hyperliens où nous avons démontré la capacité à atteindre des résultats similaires à l'état de l'art, tout en apprenant des translations.

Ce travail illustre comment les architectures profondes peuvent être délibérément apprises pour incorporer des propriétés d'invariances dans leurs représentations en fonction de la tâche d'intérêt. Cette approche ouvre une voie prometteuse pour la définition



(a) Original image **(b)** T_0 **(c)** T_1

(d) T_2 **(e)** T_3 **(f)** GSP translation

FIGURE 1 – Translations inférées $T_0, T_1, \ldots, T_3$ et comparaison avec la translation définie dans (HAMMOND et al., 2011) sur un graphe quasi-régulier.

de nouvelles opérations et de couches plus adaptées à des domaines géométriques irréguliers.

Entraînement sur des Tâches Virtuelles complexes afin d'Obtenir une Meilleure Généralisation

Dans le chapitre II, nous avons vu que la conception d'un réseau de neurones joue un rôle crucial dans sa généralisation. Néanmoins, les techniques de régularisation jouent également une part importante lorsqu'il s'agit d'améliorer les performances. À cet égard, il est courant d'avoir recours à des méthodes d'augmentations de données, qui sont une forme de régularisations dépendantes des données. De manière intéressante, certaines de ces techniques peuvent être considérées comme des méthodes augmentant artificiellement la complexité de la tâche d'entraînement.

Par exemple, Mixup (H. ZHANG et al., 2017) réalise une augmentation de données en générant des exemples "virtuels" via des interpolations linéaires et aléatoires de paires d'images. Cette méthode a démontré sa capacité à améliorer efficacement la généralisation des modèles d'apprentissage profond, conduisant à une plus grande précision. Cependant, il est important de noter que ces exemples virtuels ne respectent pas toujours la géométrie des données. Typiquement, l'interpolation de deux images, pixels par pixels, peut donner lieu à des exemples en dehors de la variété des images naturelles.

Dans le chapitre III, nous essayons de comprendre l'intérêt de Mixup en termes de

régularisation et nous proposons d'améliorer cette méthode en contraignant les interpolations à respecter une certaine localité. Notre amélioration, nommée Local Mixup, implique l'utilisation d'une matrice de poids qui vient pondérer les exemples virtuels en fonction de la distance des points impliqués dans les interpolations. Cette matrice de poids peut s'interpréter comme la matrice d'adjacence d'un graphe. Dans ces travaux, nous explorons diverses techniques pour construire ce graphe : via des graphes à seuils, des graphes de K plus proches voisins (KNN) et des graphes avec des noyaux exponentiels décroissants. Nous illustrons cette méthode dans la Figure 14.

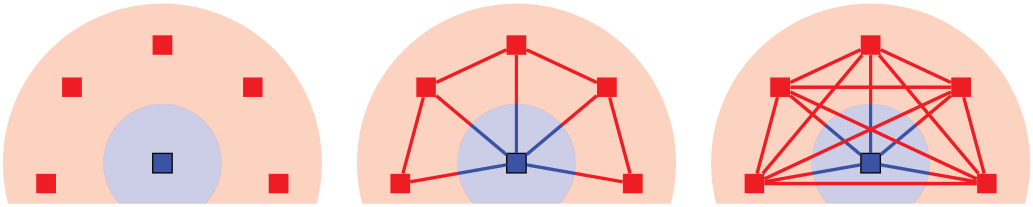


FIGURE 2 – Illustration de *Local Mixup*. À gauche, seuls les échantillons sont considérés, sans augmentation de données. Les vérités terrain sont représentées par les zones colorées. Au milieu, nous avons *Local Mixup* où seuls les échantillons suffisamment proches sont interpolés, évitant ainsi toute contradiction avec la vérité terrain. À droite, nous représentons *Mixup* dans lequel nous interpolons tous les échantillons, conduisant à des échantillons virtuels dits contradictoires. Pour une meilleure lisibilité, nous colorons les segments avec l'argmax de l'interpolation des sorties au lieu d'un dégradé de couleur.

Dans un premier temps, nous fournissons une analyse théorique pour comprendre les effets de Mixup sur la régularisation. En faible dimension, nous montrons son impact sur le biais et la variance du modèle. En plus grande dimension, nous étudions son impact sur la constante de Lipschitz du modèle en jouant sur une borne inférieure. Nous représentons expérimentalement l'évolution de cette borne sur le jeu de données CIFAR-10, comme illustré sur la Figure 18. L'avantage de notre méthode est qu'elle permet de relâcher les effets de régularisation. En pratique, Local Mixup conduit à de meilleures performances de généralisation sur plusieurs jeux de données de vision tels que SVHN, CIFAR-10 et FASHIONMNIST (voir Table 2).

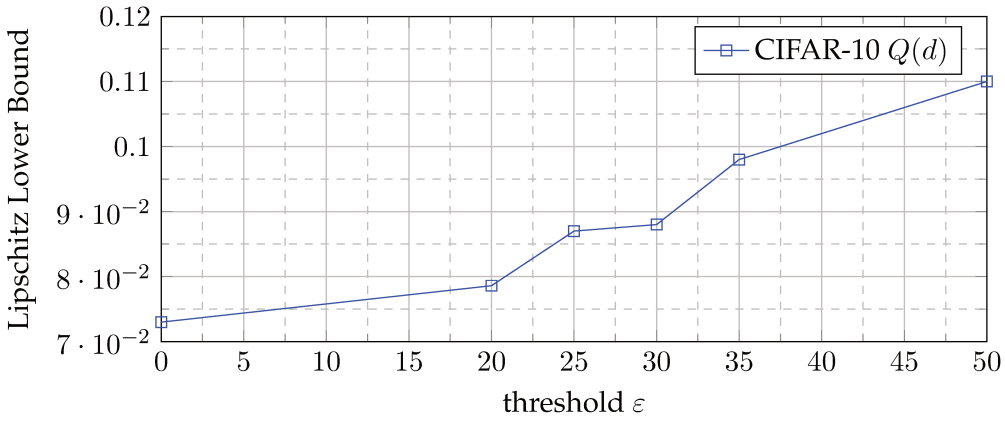


FIGURE 3 – Evolution of $Q(D)$ on the dataset cifar10. Note that $\varepsilon = 0$ corresponds to Vanilla. $\varepsilon = 50$ corresponds to classical *Mixup*.

De futures directions de recherche impliquent l'exploration de métriques plus adaptées pour mesurer la distance entre les interpolations et l'extension de nos résultats théoriques à des scénarios plus généraux.

METHOD	CIFAR-10	CIFAR-100	Fashion-MNIST	SVHN
Baseline	4.98 ± 0.03	30.6 ± 0.27	6.20 ± 0.2	10.01 ± 0.15
Mixup	4.13 ± 0.03	29.23 ± 0.4	6.36 ± 0.16	8.31 ± 0.14
Local Mixup	4.03 ± 0.03	29.08 ± 0.34	5.97 ± 0.2	8.20 ± 0.13

TABLE 1 – Taux d'erreur (%) sur CIFAR-10, CIFAR-100 (Resnet18), Fashion-MNIST (DenseNet) et SVHN (LeNet). Les valeurs sont moyennées sur 100 exécutions pour CIFAR-10 et 10 exécutions pour CIFAR-100, Fashion-MNIST et SVHN. Les erreurs moyennes avec leurs intervalles de confiance sont représentées.

Régularisation de l'entropie des caractéristiques pour améliorer la transférabilité des architectures d'apprentissage profond

Le chapitre précédent explore la possibilité d'améliorer la généralisation en s'entraînant sur une tâche artificiellement plus complexe. Dans ce chapitre, nous examinons cette fois-ci la possibilité pour un modèle entraîné sur des tâches simplifiées de prendre des décisions sur des tâches plus complexes, liées à la tâche d'entraînement. Par cette étude, nous montrons la capacité d'un modèle à généraliser au-delà de sa tâche d'entraînement. Nous observons qu'il peut effectivement extraire des informations à propos de tâches plus fines grâce à l'emploi de régularisations appropriées.

Dans ces travaux, nous introduisons une nouvelle méthode de régularisation entropique de l'extracteur de caractéristiques dénommée FIERCE. Notre méthode vise à augmenter l'entropie de l'espace des caractéristiques des modèles d'apprentissage profond, tout en préservant les informations essentielles pour des tâches ultérieures. Nos expériences démontrent en effet le rôle crucial de l'entropie de l'espace des caractéristiques dans des approches de transfert d'apprentissage.

Tout d'abord, nous examinons si une architecture d'apprentissage profond peut étendre sa généralisation au-delà de sa tâche d'entraînement avec les critères usuels d'entraînement (Entropie Croisée). Dans ce cadre, nous transformons une tâche de régression (étiquettes raffinées) en une tâche de classification (étiquettes grossières, approximées). Ensuite, nous entraînons un réseau sur la tâche de classification et essayons d'extraire des informations correspondantes aux étiquettes raffinées, ignorées durant l'apprentissage.

En utilisant l'entropie croisée et une descente de gradient stochastique, nous observons deux phases durant l'apprentissage, similairement à d'autres auteurs (SHWARTZ-ZIV & TISHBY, 2017). Dans un premier temps, l'espace des caractéristiques est capable de fournir des informations sur les étiquettes affinées, mais dans une deuxième phase d'apprentissage plus longue, ces informations sont progressivement effacées au fur et à mesure que le modèle privilégie sa performance sur la tâche d'entraînement. De manière analogue, l'entropie de l'espace de caractéristiques suit ces deux phases. Nous présentons ces résultats sur la Figure 20.

D'un point de vue théorique, nous montrons que l'entropie croisée et les méthodes de régularisations entropiques sur la sortie du réseau encouragent la sélection des caractéristiques les plus discriminantes au détriment de leur diversité. C'est pourquoi nous appliquons notre méthode de régularisation FIERCE pour encourager le modèle à utiliser davantage de caractéristiques, qui bien que moins discriminantes, peuvent porter de l'information sur la tâche raffinée. Nos expériences montrent que notre méthode permet effectivement d'augmenter l'entropie de l'espace des caractéristiques, résultant ainsi sur de meilleures performances pour plusieurs scénarios de transfert : de

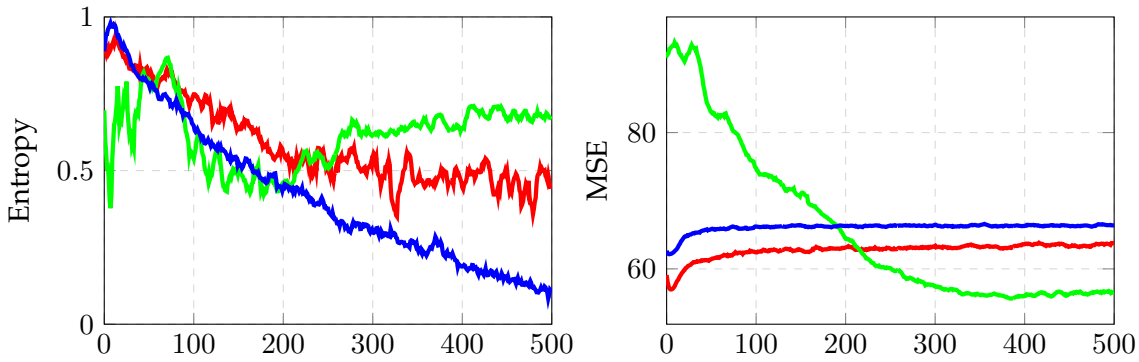


FIGURE 4 – Évolution de l'entropie de l'espace des caractéristiques (mise à l'échelle entre 0 et 1) et de l'erreur quadratique moyenne sur l'ensemble de données de régression d'âge (ROTHE et al., 2018) pour différents critères : Cross Entropy (rouge), FIERCE (proposé) avec $\lambda = 0.3$ (vert), Label Smoothing (bleu). Nous observons que les modèles entraînés avec l'entropie croisée standard ou le lissage d'étiquettes affichent brièvement une MSE minimale avant de se stabiliser à une valeur plus importante. Au contraire, la méthode FIERCE proposée atteint une MSE globalement plus basse qui est maintenue au fil de l'apprentissage. Cette capacité à atteindre et maintenir une MSE plus faible est négativement corrélée avec l'entropie de l'espace des caractéristiques, comme le montre la figure de gauche.

la régression à la classification, de l'apprentissage avec peu d'exemples (Fewshot) et de l'apprentissage "coarse to fine grains".

En résumé, nos résultats plaident en faveur du potentiel de notre méthode FIERCE dans des scénarios de classification grossière à fine et de transfert d'apprentissage. Il serait intéressant d'appliquer ces travaux sur des jeux de données hiérarchiques tels que CIFAR-100 ou iNaturalist. Notamment, étudier les métaclasses existantes et évaluer l'émergence de structures hiérarchiques et de sous-classes dans l'espace des caractéristiques qui peuvent résulter avec notre méthode.

En outre, il serait pertinent de réaliser une étude sur les propriétés géométriques de l'espace des caractéristiques et de l'évolution de ces dernières pendant l'apprentissage. Notre étude a démontré que l'entropie peut révéler des phases distinctes pendant l'apprentissage qui ne sont pas apparentes lorsque l'on surveille uniquement la précision. Il est possible que d'autres propriétés, comme la courbure de l'espace des caractéristiques ou bien sa dimension intrinsèque, évoluent pendant l'apprentissage. Pour analyser de manière exhaustive la géométrie de l'espace des caractéristiques, on pourrait envisager d'utiliser des outils tels que les graphes et métriques dérivées. Cette approche pourrait conduire au développement de nouveaux critères d'entraînement reposant sur des propriétés géométriques.

Modèle théorique sur le Transfert en Classification : Dans quelle mesure les Sous-ensembles de Classes peuvent-ils généraliser ?

Dans ce dernier chapitre, notre attention se porte sur le transfert dans le contexte de la classification. Le transfert de connaissance est devenu une approche importante dans le domaine de l'apprentissage profond, permettant aux modèles de tirer profit de connaissances acquises sur une tâche d'entraînement pour les appliquer efficacement sur de nouvelles tâches. Néanmoins, malgré son application étendue, les mécanismes sous-jacents à cette habilité restent énigmatiques.

Notre intérêt se porte en particulier sur la capacité des neurones profonds à apprendre à partir d'un sous-ensemble de classes, puis à généraliser ensuite à de nouveaux sous-ensembles de classes. Notre objectif est de caractériser ce phénomène à l'aide de fondements théoriques définis dans un cadre rigoureux. En pratique, ces définitions nous permettent de mieux comprendre la notion de transférabilité dans le cadre de la classification transfert et d'étudier des questions fondamentales dans ce domaine.

Dans notre modèle théorique, nous considérons qu'un problème de classification à n classes est soluble si chaque paire de classes peut être séparée par un modèle. Dans le cadre général, nous définissons un modèle comme une bipartition composée de sous-espaces, celle-ci séparant deux classes si les supports de classe sont inclus de manière disjointe dans les sous-espaces. Ensuite, nous définissons une relation d'ordre entre les modèles, ce qui nous permet de caractériser les modèles les plus expressifs, ceux qui séparent un grand nombre de paires de classes. Cette relation d'ordre peut s'illustrer aisément à l'aide d'un Diagramme de Hasse (voir Figure 27). Dans un cadre plus appliqué, nous proposons de considérer des modèles construits à partir d'hyperplans et adaptons les définitions générales à ce cas.

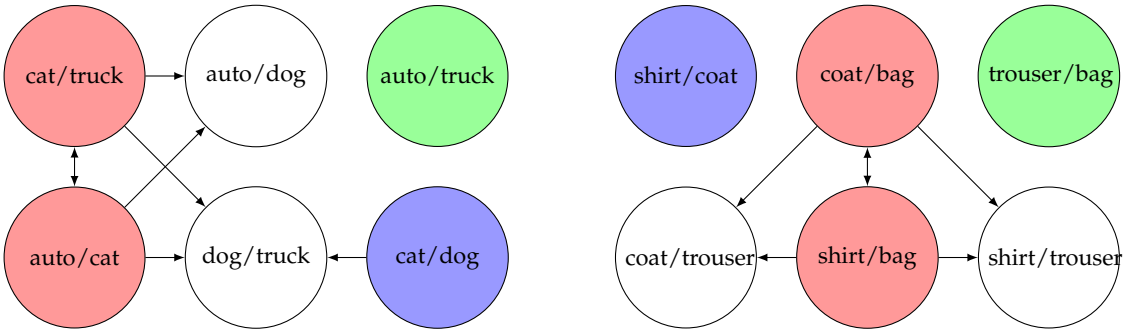
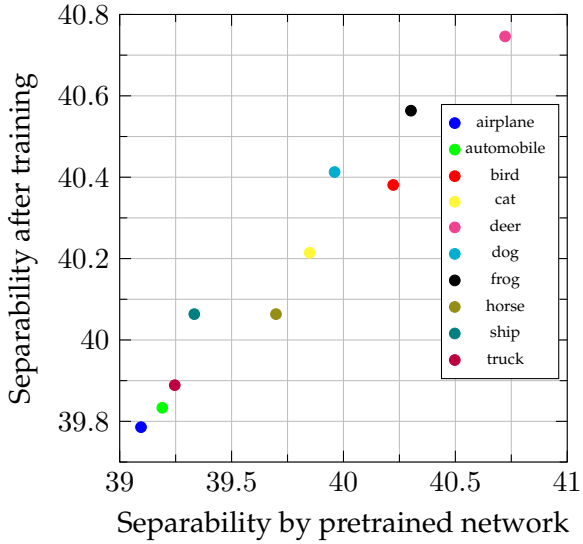


FIGURE 5 – Diagrammes de Hasse illustrant la relation d'ordre entre des modèles appris sur des paires de classes. Chaque modèle est caractérisé par les paires de classes qu'il sépare. Les flèches qui pointent d'un modèle A à un autre B signifient que le modèle A présente une meilleur expressivité, car il sépare efficacement toutes les paires que le modèle B sépare. Les modèles mis en couleur représentent les plus expressifs et sont appelés modèles 'fondamentaux'. Les modèles partageant les mêmes couleurs sont considérés comme équivalents, car ils séparent des paires identiques de classes. En revanche, les modèles non colorés sont considérés comme redondants, car les paires qu'ils séparent peuvent être séparées par d'autres modèles. À gauche, nous examinons un scénario avec 4 classes de CIFAR-10 : auto, chat, chien, camion. Le diagramme illustre que chat/camion et auto/chat sont non seulement équivalents, mais aussi plus expressifs que 'auto/chien' et 'chien/camion'. En termes plus simples, les modèles 'auto/-chien' et 'chien/camion' peuvent être omis car ils n'offrent aucune séparation supplémentaire par rapport à chat/camion et auto/chat. À droite, en considérant 4 classes de FASHION-MNIST : sac, manteau, chemise, pantalon. Le diagramme révèle que manteau/sac et chemise/sac sont équivalents et possèdent une plus grande expressivité que manteau/pantalon et chemise/pantalon.

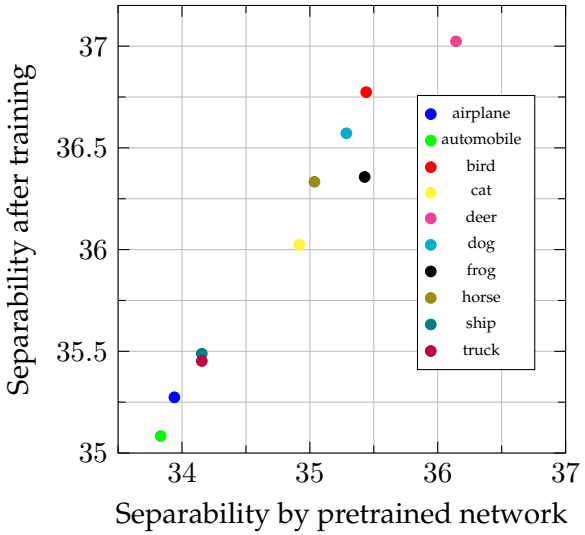
D'un point de vue expérimental, nous considérons des jeux de données composés d'un ensemble de classes C, et nous cherchons à nous entraîner sur des sous-ensembles $C' \subset C$. Dans ce cadre, nous nous demandons s'il est possible d'identifier les sous-ensembles de classes conduisant à la meilleure généralisation. Cela nous amène à exploiter notre modèle théorique et à introduire la notion de séparabilité, qui caractérise le potentiel

de transfert d'un modèle entraîné sur un ensemble de classes C'.

Nous montrons que la séparabilité est un bon indicateur de performances en transfert dans plusieurs scénarios : fine-tuning où un réseau pré-entraîné est fine-tuné sur un sous-ensemble de classes, entraînement de zéro où une architecture est entraînée sur un sous-ensemble de classe, et Fewshot. Une approche plus qualitative nous amène à définir la séparabilité par classe que nous représentons sur la Figure. Cette figure illustre l'impact de chaque classe sur la généralisation. En axe x, la séparabilité est donnée par un réseau pré-entraîné, en axe y, la séparabilité d'un réseau entraîné uniquement sur le sous-ensemble de classe. Sur la Figure 34, nous pouvons observer que pour le jeu de données CIFAR-10, les classes les plus importantes sont le cerf, l'oiseau, le chien et l'oiseau.



(a) Entraîné à partir de zéro sur 6 classes de CIFAR10. L'analyse met en évidence les classes les plus prometteuses, telles que le cerf, l'oiseau, le chien et la grenouille.



(b) Entraîné à partir de zéro sur 4 classes de CIFAR10. L'analyse met en évidence les classes les plus prometteuses, telles que le cerf, la grenouille, le cheval et l'oiseau.

FIGURE 6 – Entraînement à partir de zéro sur des sous-ensembles de classes de CIFAR10. L'axe des x représente les résultats du **VIT-8 préentraîné**, tandis que l'axe des y correspond au Resnet18 entraîné à partir de zéro sur des sous-ensembles de classes de CIFAR10.

Des recherches supplémentaires pourraient exploiter le cadre présenté dans ce chapitre afin d'explorer plusieurs questions d'importance de l'apprentissage par transfert. Dans WIGHTMAN et al., 2021, les auteurs soulignent qu'il est crucial d'adapter la routine d'entraînement à chaque architecture, mais trouver la routine d'entraînement optimale (critères, augmentations de données, régularisations, etc.) reste une tâche complexe. Notre modèle pourrait être exploité pour examiner l'impact des différentes procédures d'entraînement sur les représentations apprises et leur transférabilité. De plus, le formalisme introduit dans ce travail ouvre des possibilités pour l'élaboration de jeux de données difficiles pour des scénarios de transfert. En choisissant soigneusement des ensembles de classes avec des caractéristiques spécifiques, les chercheurs peuvent créer des benchmarks servant à t

Conclusion générale

En résumé, nous nous sommes plongés dans plusieurs scénarios où les architectures d'apprentissage profond démontrent leur capacité à généraliser au-delà de leur tâche d'entraînement. Cette habileté a été largement exploitée dans le domaine de l'apprentissage par transfert. La quintessence serait d'atteindre une intelligence artificielle dite générale capable de généraliser, d'apprendre de manière indépendante, tout en pouvant résoudre des tâches nouvelles. Comme l'a déclaré Demis Hassabis, ***l'apprentissage par transfert*** *est la clé de l'intelligence générale. [...] La clé d'un apprentissage par transfert réussi réside dans l'acquisition de* ***connaissances conceptuelles abstraites*** *des détails perceptuels de sa source*.

Par conséquent, au cours des dernières années, le domaine de l'apprentissage profond a connu une prolifération de nouvelles architectures, de nouveaux critères et de processus d'entraînement, amenés par la communauté afin de fournir des représentations de données de plus en plus abstraites et générales. Notamment, il y a eu un changement de paradigme, avec de plus en plus de chercheurs et d'industriels qui tendent à exploiter de larges modèles pré-entraînés sur d'immenses jeux de données, pour les affiner sur leurs tâches d'intérêts. Dans le domaine de la vision par ordinateur, l'aspiration ultime est le développement d'un extracteur de caractéristiques universel, capable de représenter des images naturelles, qui pourrait résoudre n'importe quel problème de classification.

Pourtant, il reste encore de nombreuses problématiques fondamentales à explorer dans le domaine de l'apprentissage par transfert. Bien que les méthodes d'apprentissage auto-supervé et non supervisé soient devenues des approches dominantes dans ce domaine, la détermination des processus d'entraînement optimaux pour du transfert demeure ambiguë. Cette ambiguïté est exacerbée par le fait que chaque architecture tend à s'accompagner de son propre processus d'entraînement (WIGHTMAN et al., 2021). En conséquence, des questions fondamentales persistent, notamment :

- Quels composants de l'architecture devraient être affinés ou figés ?
- Quel type d'information est extrait à chaque couche du modèle ?
- Quelles propriétés géométriques (invariance, équivariance) devraient être ciblées par différentes parties du modèle ?
- Pouvons-nous concevoir une architecture avec une généralité suffisante pour éviter le surajustement à la tâche d'entraînement ? Que signifie être trop spécialisé pour une tâche, et dans quels scénarios cela pose-t-il problème ?

Dans cette thèse, notre exploration s'est principalement centrée sur des scénarios dans le cadre de la classification. Nous avons examiné l'apprentissage par transfert d'une tâche de classification à une autre, appris des opérateurs invariants par rapport à la classification, généré des tâches artificielles complexes pour améliorer la généralisation sur des ensembles de données de classification. Nous nous sommes concentrés sur la **reconnaissance de motifs** (pattern recognition), et avons mis en lumière comment la capacité des réseaux de neurones profond à apprendre des motifs généraux applicables à d'autres tâches.

Cependant, comme le souligne (LAKE et al., 2017), les modèles d'apprentissage profond, malgré leur habileté en terme de reconnaissance de motifs et leurs performances impressionnantes sur de nombreuses tâches, manquent toujours de certaines caractéristiques essentielles similaire à celle des humains. Une véritable intelligence, dite générale, englobe davantage que la simple reconnaissance de motifs ; elle englobe la capacité

à raisonner, à apprendre à partir de peu d'exemples, et à comprendre les relations causales entre les objets et les événements du monde.

Les modèles d'apprentissage profond ont obtenu un succès remarquable dans des tâches telles que la reconnaissance d'images, le traitement du langage naturel. Cependant, ils ne possèdent pas le même niveau d'abstraction et de bon sens que les humains. Bien que l'apprentissage profond ait ses forces, de nouvelles techniques et des paradigmes supplémentaires seront nécessaires pour atteindre une intelligence polyvalente.

Acknowledgments

I would like to express my deepest appreciation for the invaluable guidance and support provided by my thesis director, Professor Vincent Gripon, and my supervisor, Professor Lucas Drumetz. Professor Vincent Gripon provided me with the opportunity to work alongside him during my undergraduate engineering studies on a research project. This initial project played a pivotal role in shaping my career path and inspiring me to pursue a research-focused journey in deep learning. Professor Lucas Drumetz, who also served as my professor during my engineering studies, has been an exceptional mentor throughout my thesis.

In addition to their exceptional mentorship in the academic realm, I would like to emphasize and express my gratitude for their personal and human qualities, as they have always been very supportive during this thesis.

I express my sincere appreciation to Professor Antonio Ortega, who granted me the privilege of visiting his laboratory and collaborating with his team at the University of Southern California (USC) for several months. I extend my thanks to him for his warm hospitality and generous consideration.

Thanks should also go to my incredible team at Brain, including permanent staff, postdoctoral researchers, and Ph.D. students, for fostering a collaborative and supportive atmosphere within our research lab. I wish to express my thanks to the lab secretaries of the department MEE for their great support in coordinating and managing various logistical aspects of my academic journeys,

I would like to acknowledge Michel Jezequel, who supervised the research training program at IMT Atlantique during my student years. This program played a pivotal role in shaping my research career and provided me with invaluable opportunities. Furthermore, I extend my thanks to the faculty members of IMT Atlantique, particularly Professor Dominique Pastor and Professor Abdeldjalil Aissa El Bey, who oversaw the "Mathematical and Computational Engineering" specialty during the final year of my engineering degree. I deeply appreciated the courses offered by this specialty.

Many thanks to the exceptional educators who have guided me throughout my academic journey, from middle school to high school and my preparatory classes. Their dedication to teaching and their ability to ignite a passion for science within me have been instrumental in my academic development. Without their influence, I would not have reached my current position. A special thank you is due to Mrs. Eveline Elaldi, my physics and chemistry teacher during middle school, who also supervised the astronomical club at College Jean Rostand in Draguignan. Mrs. Elaldi was unquestionably one of the first individuals who sparked my interest in science.

Lastly, none of my achievements would have been possible without the unwavering support of my loving family, friends, and my partner. I want to express my heartfelt gratitude to my parents for their enduring love and care throughout my life. Words can not convey the depth of my affection and appreciation for them. Bertrand, you have been a constant source of support and encouragement throughout my journey in completing this thesis.

Chapter 1

Introduction

Contents

1.1 Scientific context

From natural language processing to image recognition, Artificial Intelligence (AI) has managed to solve very complex tasks that were until relatively recently thought impossible for machines (Devlin et al., 2018; Radford et al., 2021; Radford et al., 2018). One of AI's impressive abilities is its capacity to learn and recognize patterns by itself in large amounts of data, and achieve accurate predictions on unseen data. This capability is called generalization (Goodfellow et al., 2016).

Originally, generalization in AI was limited to predictions on data similar to that used to train the AI (Caruana, 1997; Zhuang et al., 2020a). However, recent literature in the field of transfer learning have extended the notion of generalization beyond its original context (Caron et al., 2020; Fei-Fei et al., 2004; Krizhevsky et al., 2017; Pan & Yang, 2010; Radford et al., 2015). Transfer learning allows AI systems to transfer knowledge and skills to distinct domains and distinct tasks.

Despite these advancements, the field of Transfer Learning still presents some limitations and unanswered questions, such as which training procedures lead to the best results on the final task and what mechanisms underlie them (Huh et al., 2016; Janocha & Czarnecki, 2017a; Tian et al., 2020a; Yosinski et al., 2014). In this manuscript, we consider an extension of the usual scope of generalization and envision multiple scenarios to address these questions.

1.1.1 Artificial Intelligence

Originally, artificial intelligence faced a significant hurdle when it came to tackling tasks that humans find easy to execute but difficult to articulate formally (Goodfellow et al., 2016). These are the kind of problems that we solve naturally such as identifying spoken language or facial recognition in images.

Nevertheless, researchers have managed to design computer programs capable of solving not only some of theses tasks (Chan et al., 2015) but also tasks considered as very complex by humans such as driving cars (Bojarski et al., 2016), competing at the highest level in strategy games (Bakhtin et al., 2022; Silver et al., 2016), generating audiovisual/text content (Radford et al., 2018; Rombach et al., 2022).

Such breakthroughs have been achieved by the use of Machine Learning (ML), a subfield of artificial intelligence that enables computers to acquire knowledge by extracting patterns from data without being explicitly programmed (Bishop & Nasrabadi, 2006). The democratization of Machine Learning was brought by the access to large amounts of data (referred to as "big data"), and powerful and cheaper computational resources, especially graphical processing unit (GPU), at the beginning of the XXI century (Bommasani et al., 2021).

1.1.2 Deep learning

Deep learning, in particular, has emerged as a dominant approach within ML, driving many of the most significant advances in the field. Deep learning architectures are designed with a series of simple building blocks, often called layers, that extract increasingly complex features from the input. This allows a deep learning architecture to learn representations of the data, which can capture subtle patterns and relationships that may not be apparent or clearly defined by human experts.

One of the most significant breakthroughs in deep learning was the development of convolutional neural networks (CNNs) (Krizhevsky et al., 2017; LeCun et al., 1998),

which have revolutionized computer vision. CNNs are able to extract features from images at multiple scales and locations, making them highly effective for tasks such as object detection (Redmon et al., 2016) and image segmentation (Minaee et al., 2021).

The success of CNNs has led to the development of deep learning models for a wide range range of problems in artificial intelligence, including classification or regression in computer vision.

1.1.3 Supervised learning: Generalization on a specific task

Supervised learning is the classical approach used to solve the aforementioned problems (Goodfellow et al., 2016). In this approach, each example is labeled with a corresponding target value – e.g. an element of a finite dictionary in the case of classification– , and the architecture is trained by comparing its predicted output with the correct label through a predefined criterion. This process enables the model to learn the relationship between the input data and the output labels, allowing it to make accurate predictions on new, unseen data. However, one of the main limitations of supervised learning is that it builds a model that is specific to the given task (Radford et al., 2015). This means that if the task is modified, such as by adding new classes to a classification problem, the model's performance can be severely impacted, and it may need to be fully re-trained. This makes it challenging to adapt the model to new, related tasks, which is a common problem in real-world applications.

1.1.4 Transfer Learning

Recently, researchers have developed transfer learning approaches that aim to leverage knowledge learned from one task to improve the performance on related tasks (Pan & Yang, 2010; Zhuang et al., 2020a). These approaches can be particularly useful in situations where there is limited labeled data available for a new task, as they can enable the model to learn from a related task with more labeled data. Moreover, they prove invaluable when training is too costly or resource-intensive to collect extensive labeled data for each specific task. One common approach is Self Supervised Learning where a neural network is trained on a large labeled dataset to learn a general feature extractor that can then be leveraged on smaller, possibly distinct, datasets (T. Chen et al., 2020; He et al., 2020; Jing & Tian, 2020; Oord et al., 2018).

However, these approaches present their own challenges. There are multiple parameters that can influence the representations learned and their ability to be reused on different tasks, including the criteria used for training, regularizations, number of parameters, architecture, and subtlety of the labels (e.g., hierarchical, semantic, multi-class) (Huh et al., 2016; Tian et al., 2020a; Wightman et al., 2021; Wu et al., 2017).

Therefore, it is not clear which training procedures would necessarily lead to the highest transferability and what possible limitations there may be (d'Ascoli et al., 2021). Future research will need to explore these questions further in order to develop more effective transfer learning approaches for AI.

1.2 Classification and Vision: An Overview of Supervised Learning with Deep Neural Networks

Supervised learning has been a key area of research in machine learning and has shown significant progress in a wide range of applications, particularly in classification tasks (LeCun, 2015a) in computer vision. In this chapter, we focus on classification in the context

of supervised learning. Specifically, we examine the use of deep learning architectures known to achieve state of the art in classification tasks (Voulodimos et al., 2018).

Vision datasets have become increasingly popular in recent years as a benchmark for evaluating the performance of deep learning models (L. Deng, 2012; Goodfellow et al., 2016; Krizhevsky, Hinton, et al., 2009; Krizhevsky et al., 2017). Although our focus is not exclusively on vision tasks, we leverage the availability of solutions for this domain to explore the notion of generalization later on in the manuscript.

1.2.1 Deep Learning Architecture for classification

The field of deep learning has rapidly evolved with increasingly complex architectures being designed for various applications, particularly on the task of classification in vision (Dosovitskiy et al., 2020; He et al., 2016; LeCun et al., 1998). However, the incorporation of more and more parameters has led to difficulties in training these architectures (d'Ascoli et al., 2021; Krizhevsky et al., 2017; Wightman et al., 2021). These challenges include for instance vanishing gradients (Glorot & Bengio, 2010), where gradients become too small, hindering information propagation across deep layers; dimensional collapse (Jing et al., 2021), which leads to the loss of crucial features in network representations; overfitting, causing models to become overly specialized to training data; and the risk of exploding gradients (Pascanu et al., 2013), which disrupts training stability. These collectively hinder the effective training of complex deep architectures.

As a result, novel techniques and architectures have been proposed to address this issue, such as Residual Neural Networks (He et al., 2016). In this section we present the architectures that we widely used in the experiments of this thesis.

Deep Learning Architecture

The goal of a Deep Learning Architecture is to approximate some function f^*. In the context of classification, an example is a classifier $\mathbf{y} = f^*(\mathbf{x})$ that maps an input $\mathbf{x}$ to a class $\mathbf{y}$. One of the key to the success of Deep Learning Architectures lies in their large number of parameters and complexity to approximate such functions (He et al., 2016; LeCun, 2015a; Simonyan & Zisserman, 2014).

Definition 1: Deep Learning Architecture

A ***Deep Learning Architecture*** defines a mapping $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$ and learns the parameters $\boldsymbol{\theta}$ that result in a approximation of f^*. The mapping is represented by a composition of simpler functions called ***layers***.

The basic architecture is a chain structure, also referred to as a feedforward neural network, where l layers are connected in a sequence $f = f^{(l)} \circ f^{(l-1)} \circ f^{(1)}$. The length of the chain, l, determines the ***depth*** of the architecture, and can be very large, resulting in a complex and powerful models. Hence the name *deep* learning.

Definition 2: Layers of a Deep Learning Architecture

A ***layer*** is a function that takes an input vector $\mathbf{x}$ and produces an output vector (Goodfellow et al., 2016). It consists of a linear transformation followed by a non-linear function σ, also known as the ***activation function***: real-valued functions applied component-wise to real-valued variables.

The transformation is defined as follows:

$$\mathbf{x} \mapsto \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \text{ where } \mathbf{W} \text{ is a matrix and } \mathbf{b} \text{ a vector.}$$

The parameters $\{\mathbf{W}, \mathbf{b}\}$ are learned during training over a set of data. When there is no constraint on the form of $\mathbf{W}$, the layer is said to be ***fully connected***

The non-linearity is essential and enable to capture complex patterns in the data. Common activation functions include sigmoid, hyperbolic tangent, and Rectified Linear Unit (ReLU).

The last layer is usually fully connected and commonly referred to as the ***output***. Its activation function is typically chosen to produce a specific range of values, such as $[0, 1]$ (useful e.g. to represent probabilities) in the case of the softmax function or real-valued outputs in the case of the hyperbolic tangent or the Rectified Linear Unit (ReLU).

The layers in between are called *hidden* layers. Of particular interest is the output of the penultimate layer, often referred to as the ***feature space***. This is because it represents a learned abstraction of the input data that is used by the final layer to produce the output. Hence, the feature space captures the most important aspects of the input data that are relevant for the task.

Therefore, in the context of classification, we will typically view the architecture in two parts: a ***feature extractor*** and a classifier. The goal of the feature extractor is to deliver a representation of the data on which the classifier can separate the classes. Typically, the classifier is linear, i.e. a logistic regression, as it is defined by a simple linear layer followed by the softmax function.

Convolutional Neural Network

The breakthrough in computer vision can be largely attributed to the success of Convolutional Neural Networks (CNNs) (LeCun et al., 1998). These networks utilize ***Convolutional layers*** to learn and represent visual features in a hierarchical manner. By doing so, they capture the relevant patterns and structures present in the input data, enabling state-of-the-art performance on tasks such as object detection, image classification, and semantic segmentation.

Definition 3: Convolutional Layer

Convolutional layers have a matrix $\mathbf{W}$ constrained to define convolution operations. This design ensures that the hidden layers combine only local information, as research has shown the benefits of extracting local features and combining them into high-order representations called ***feature maps***.

Convolutional layers can be followed by a ***pooling layer*** to reduce the dimension of the output. Moreover, in the case of images, the association of a convolutional and pooling layer provide a transformation that its invariant to translation and small pertubations (LeCun et al., 1998; Mallat, 2016).

In the following definition, we will assume that the input data is in the form of images with shape (N, C, H, W), where N represents the number of samples, C denotes the number of channels (e.g., black and white, RGB, hyperspectral), H indicates the height, and W the width of the image. To represent a specific input in vector notation, we will use the notation $\mathbf{X}[i, j, k, l]$, and $\mathbf{x}$ to refers to a single image of shape (C, H, W).

Definition 4: Pooling Layer

A ***pooling layer*** is a type of layer, typically applied after a convolutional layer, that operates on its input by partitioning it into non-overlapping regions and computing a fixed-dimensional output for each region. As a result, the dimension size of the input is reduced.

Two common types of pooling layers are ***average pooling*** and ***max pooling***.

Definition 5: Average Pooling

Given an input tensor $\mathbf{X}$ of shape (N, C, H, W), an ***average pooling*** operation computes the output tensor $\mathbf{Y}$ of shape (N, C, H', W') by partitioning each channel of $\mathbf{X}$ into non-overlapping regions and computing the average of the values in each region.

Definition 6: Max Pooling

Given an input tensor $\mathbf{X}$ of shape (N, C, H, W), a ***max pooling*** operation computes the output tensor $\mathbf{Y}$ of shape (N, C, H', W') by partitioning each channel of $\mathbf{X}$ into non-overlapping regions and computing the maximum value in each region.

Residual Neural Networks

Multiple papers have highlighted the crucial importance of the depth of the architecture in achieving high accuracy on challenging vision datasets (Simonyan & Zisserman, 2014). However, one issue with such architectures is the degradation problem, which limits the depth of the network. Indeed, as the depth increases, accuracy saturation occurs, leading to reduced performance (He et al., 2016).

To address the degradation problem, ***Residual layers*** have been introduced as an effective solution. One of the most popular and widely used architectures that uses residual networks is ResNet, as described in (He et al., 2016). ResNet has been widely used as a benchmark in computer vision (Tan & Le, 2019; Yun et al., 2019; H. Zhang et al., 2017). Although the emergence of attention layers and self-supervised learning have questioned the use of ResNets (Dosovitskiy et al., 2020), recent studies have shown that ResNets can still compete and achieve state-of-the-art performance with improved training procedures, as demonstrated in (Wightman et al., 2021).

Definition 7: Residual Layers

A residual layer is defined as a building block $\mathcal{H}(\mathrm{x}) = \mathcal{F}(\mathrm{x}) + \mathrm{x}$, where $\mathcal{F}$ is a stack of several layers, such as the composition of multiple convolutional layers:

$$\mathcal{H}(\mathrm{x}) = \mathcal{F}(\mathrm{x}) + \mathrm{x}$$

If the optimal mapping is the identity $\mathcal{F}$ should converge to 0.

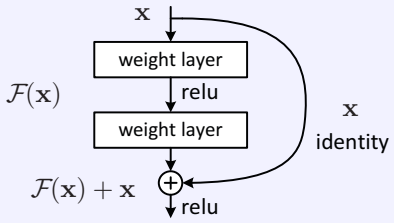


Figure 7 – An illustration of a residual layer with Relu as activation function. Here $\mathcal{F}$ is the composition of two layers. Figure is from (He et al., 2016).

As we conclude this section, our attention naturally turns to the presentation of the standard datasets used in supervised learning, building upon the foundation we have laid by introducing standard architectures in classification.

1.2.2 Standard Vision Datasets for Supervised Learning

MNIST

The MNIST dataset (L. Deng, 2012) is a popular benchmark in computer vision and machine learning. It is composed of 70,000 grayscale images of handwritten digits, each of which is 28x28 pixels in size. The images are split into a training set of 60,000 images and a test set of 10,000 images. Each image in the dataset is labeled with the corresponding digit it represents, ranging from 0 to 9.

The MNIST dataset has been widely used as a benchmark for evaluating the performance of machine learning models, particularly those for image classification tasks. It allowed to demonstrate the ability of convolutional neural networks (LeCun et al., 1998). Since then, it has also been used in numerous research studies and publications to test various method in computer vision (LeCun, 2015a).

Compared to others dataset in vision, MNIST is relatively small which allow to run numerous experiments and test various hypothesis. One disadvantage is that the task is relatively easily and current state-of-the art networks reach a nearly perfect accuracy.

CIFAR10 and CIFAR100

The CIFAR-10 dataset (Krizhevsky, Hinton, et al., 2009) is another widely used as a benchmark dataset in the field of computer vision. It consists of 60,000 color images in 10 classes, with 6,000 images per class. Each image in the dataset is 32x32 pixels in size. The classes include: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

The CIFAR-10 dataset has been used for a variety of computer vision tasks, including image classification, object detection, and segmentation. It has been a popular choice for evaluating the performance of deep learning models, particularly convolutional neural networks (CNNs).

One advantage of CIFAR10 is the high quality and diversity of the data. The classification is more challenging than MNIST and it still used to test recent and advanced architecture such as Resnet (He et al., 2016). Moreover, it has been used in numerous research studies and publications, including papers on image classification, object detection, and transfer learning. For example, (Krizhevsky, Hinton, et al., 2009) used the

CIFAR-10 dataset to train a deep convolutional neural network that achieved state-of-the-art performance on image classification, while (Girshick et al., 2014) used it to study the transferability of features learned from pre-trained CNNs.

Overall, the CIFAR-10 dataset remains a valuable resource for the development and testing of new computer vision algorithms and techniques.

Similarly, the CIFAR-100 dataset (Krizhevsky, Hinton, et al., 2009), an extension of CIFAR-10, serves as another essential benchmark in the field of computer vision. CIFAR-100 comprises 60,000 color images, divided into 100 classes, each containing 600 images. Like its predecessor, each image in the CIFAR-100 dataset maintains a resolution of 32x32 pixels. The diversity of classes in CIFAR-100 ranges from common objects to more fine-grained categories, making it a valuable resource for assessing the robustness and versatility of computer vision models.

CIFAR-100 presents unique challenges due to its increased class complexity, demanding deeper and more discriminative models to achieve high classification accuracy. Researchers often leverage CIFAR-100 to explore a wider range of computer vision tasks, including fine-grained image classification (Touvron et al., 2021), object recognition (Redmon et al., 2016), and transfer learning. It has been instrumental in the development and evaluation of state-of-the-art models, such as ResNet (He et al., 2016), which have consistently demonstrated their effectiveness in handling this rich and varied dataset. Additionally, CIFAR-100 has been a subject of investigation in numerous studies focusing on feature transferability, model generalization, and the impact of various architectural and training strategies (Krizhevsky et al., 2017; Radford et al., 2015).

Imagenet

The ImageNet dataset (J. Deng et al., 2009) is a widely used benchmark dataset in the field of computer vision and machine learning. It is one of the largest publicly available datasets for visual object recognition, containing over 14 million images categorized into more than 20,000 object classes. The images in the dataset have been annotated with bounding boxes and image-level labels, which makes it suitable for training and evaluating object detection and classification models.

The large size of the dataset makes it more difficult than MNIST and CIFAR10 (Russakovsky et al., 2015). It has been widely used in research studies and competitions to benchmark and evaluate the performance of computer vision models, particularly deep learning models. For example, the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC) has used a subset of the ImageNet dataset for its image classification and object detection challenges since 2010, which has resulted in many breakthroughs in the field of computer vision. In addition to classification and detection, the dataset has also been used for other computer vision tasks such as segmentation or object detection.

Many state-of-the-art deep learning models for computer vision have been developed and tested on the ImageNet dataset. For instance, the VGG-16 (Simonyan & Zisserman, 2014) and ResNet models(He et al., 2016) were developed by training on subsets of the dataset, achieving significantly higher accuracy rates than previous models.

The availability and high quality of the ImageNet dataset have made it a critical resource for both supervised and self-supervised learning in computer vision. For transfer learning, numerous models has been pre-trained on ImageNet and fine-tuned on other datasets for specific tasks (Oquab et al., 2014; Yosinski et al., 2014).

Futhermore, self-supervised learning has recently emerged as a promising approach for pre-training deep neural networks on large datasets such as ImageNet without requiring manual annotations. Several works have shown that models pre-trained with contrastive learning on ImageNet achieve state-of-the-art results on a variety of downstream computer vision tasks (T. Chen et al., 2020; He et al., 2020; Tian et al., 2020b).

1.2.3 Training Procedure

Deep learning architectures are trained using an optimization algorithm that minimizes a loss function over a large dataset. Depending on the task various loss functions can be used. In classification, the cross entropy is the most popular choice.

Definition 8: Cross-Entropy Loss

The ***cross-entropy loss*** is a fundamental criterion in machine learning used to measure the dissimilarity between predicted and true probability distributions for a given dataset. Let's formally introduce the key terms involved:

- N: The number of instances in the dataset.
- K: The number of distinct categories or classes in the classification problem.
- $\mathbf{p}_i$: The true probability distribution over the K categories for the i-th instance, where $\mathbf{p}_i = (p_{i1}, p_{i2}, \ldots, p_{iK})$, with p_{ij} denoting the probability of category j for instance i.
- $\mathbf{q}_i$: The predicted probability distribution over the same K categories for the i-th instance, where $\mathbf{q}_i = (q_{i1}, q_{i2}, \ldots, q_{iK})$, with q_{ij} denoting the predicted probability of category j for instance i.

The cross-entropy loss for the entire dataset is defined as:

$$H(\mathbf{p}, \mathbf{q}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} p_{ij} \log(q_{ij}) \tag{1}$$

The cross-entropy loss quantifies how well the predicted probabilities $\mathbf{q}$ match the true probabilities $\mathbf{p}$ across the entire dataset.

One of the key advantages of deep learning architectures is that they allow for automatic differentiation, which makes it possible to compute gradients of the loss function with respect to the model parameters efficiently using the chain rule and backpropagation. Among the most widely adopted techniques for optimizing these gradients is Stochastic Gradient Descent (SGD) and its variants (Amari, 1993; Sutskever et al., 2013)

Stochastic Gradient Descent

Stochastic gradient descent (SGD) is a widely used optimization algorithm for training deep Learning Architecture (Bottou & Cun, 2003; Robbins & Monro, 1951). It works by randomly selecting a small subset of the training data (known as a batch) and computing an estimate of gradient of the loss function with respect to the model parameters using only the examples in the batch.

The gradients are computed using the chain rule method and backpropagation (Le-

Cun, 2015a), which involves computing the gradient of each layer of the network with respect to its inputs and passing the gradients backward through the network. The model parameters are then updated in the direction of the negative gradient, scaled by a learning rate hyperparameter. This process is repeated for many iterations, called ***epochs***, over the entire dataset.

Stochastic gradient descent presents many advantages:

- Efficiency: By splitting the training data into smaller subsets SGD is computationally efficient and can be used to train large deep neural networks on massive datasets.
- Generalization: Because SGD updates the model parameters incrementally using small batches of training data, it can help prevent overfitting and improve generalization. This is because it introduces randomness into the optimization process, which can help prevent the model from becoming overly specialized to the training data and instead encourages it to learn more generalizable features.
- Flexibility: SGD is a flexible optimization algorithm that can be adapted to different learning scenarios and loss functions. For example, different learning rates can be used for different model parameters, and the optimization process can be modified using techniques such as momentum or adaptive learning rate schedules (Smith, 2017; Sutskever et al., 2013)
- Robustness: SGD is a robust optimization algorithm that can handle noisy and non-convex loss functions. This is because it updates the model parameters incrementally and can navigate complex loss landscapes more effectively than batch optimization algorithms.
- Scalability: SGD is a scalable optimization algorithm that can be parallelized and distributed across multiple machines, enabling faster training of large-scale deep neural networks.

While stochastic gradient descent (SGD) remains a popular choice for training neural networks, the Adam optimizer (Kingma & Ba, 2014) presents a compelling alternative since it overcomes some of the limitations of traditional SGD (Keskar & Socher, 2017). It incorporates adaptive learning rates, ensuring that each parameter receives an appropriate update size, which can significantly accelerate convergence. Additionally, Adam employs momentum and bias correction, enhancing the stability and robustness of the optimization process. These features collectively make Adam a preferred choice for optimizing neural networks, as it often leads to faster convergence and improved performance compared to SGD.

Regularization

Due to their large number of parameters Deep Neural Network are prone to overfitting (He et al., 2016; Krizhevsky et al., 2017). ***Overfitting*** occurs when the model is able to fit the training data very well, but performs poorly on new data. To prevent overfitting, regularization techniques are used to encourage the model to learn simpler and more generalizable representations.

One common regularization technique is weight decay (Loshchilov & Hutter, 2017), which adds a penalty term to the loss function proportional to the L2 norm of the model parameters. This encourages the model to use smaller weights, which tends to result in smoother decision boundaries and less overfitting. Another commonly used technique

is dropout, which randomly drops out a fraction of the activations in the network during training (Srivastava et al., 2014). This makes the network more robust to noise and encourages the model to learn redundant representations.

The choice of optimization algorithm, loss function, and regularization techniques can have a significant impact on the performance of a Deep Neural Architecture. In particular, regularization affects the geometry of the feature space (Janocha & Czarnecki, 2017a; Shen et al., 2021). Therefore, the choices of regularizations depend on the specifics of the problem being solved (Tian et al., 2020a).

1.3 Problem Statement

In classification, the ultimate goal is to learn a function f that maps input data to output labels, and achieve high performances on unseen data. Such ability is called generalization. We will first define this concept and then discuss the limit of this definition.

1.3.1 Criterion (Loss Function)

The criterion, or loss function, is a metric that tells us how well the model's predictions match the ground truth labels. In other words, it quantifies the error between the predicted labels and the actual labels. The goal of training a deep learning model is to minimize this error, so that the model can make accurate predictions on new data.

In the context of classification, the criterion is typically used to quantify the error between the predicted labels $\hat{y} = f(\mathbf{x})$ and the ground truth labels y in a training set $D_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$. The criterion is a non-negative function that is minimized during training, typically by adjusting the model's parameters through optimization algorithms such as stochastic gradient descent. There are various types of criteria, such as mean squared error, cross-entropy, and binary cross-entropy, each suited to different types of problems (Bishop & Nasrabadi, 2006).

1.3.2 Risk Minimization

In machine learning, the optimal solution is the one which minimizes the loss function over all possible inputs and outputs. This is known as risk minimization, and it ensures that the model generalizes well to unseen data.

Definition 9: Expected Risk

Let us assume that we are given a joint distribution $P(\mathbf{x}, y)$ over the inputs and labels, a model f and a criterion $\mathcal{L}$, the expect risk is defined as:

$$\mathcal{R}(f) = \mathrm{E}[\mathcal{L}(f(\mathbf{x}, y))] = \int \mathcal{L}(f(\mathbf{x}, y)) dP(\mathbf{x}, y).$$

However, in practice, we do not have access to the joint distribution, and it is impossible to directly minimize the expected risk. Instead, we rely on ***empirical risk*** minimization, which minimizes the average loss over the training set D_{train}. The empirical risk if then defined as:

$$\hat{\mathcal{R}}(f) = \mathrm{E}_{D_{train}}[\mathcal{L}(f(\mathbf{x}, y)] = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}_i, y_i)).$$

By minimizing the empirical risk, we hope to find a function f that performs well on unseen data. However, this approach can lead to overfitting if the model becomes too complex and fits the noise in the training set rather than the underlying patterns.

1.3.3 Generalization

Generalization usually refers to the ability of a model to perform well on new, unseen data. To measure generalization performance, we evaluate the model on a test set, which is separate from the training set. In classificaiton, the performance of the model is measured using a metric such as accuracy, which is the fraction of correctly classified samples. When the model performs well on the training data (small empirical risk), but poorly on test data this is called ***overfitting***.

In addition to generalizing to new data, recent literature shows that deep learning architecture can also generalize beyond their training task. For example, a model trained on a classification task can be fine-tuned on a related task with only minor modifications. The feature space can be either *frozen*, i.e only the output layer is retrained, or the whole architecture can be retrained with a very small learning rate. In other words, given task, a model can learn features general enough to be reused for others tasks.

1.3.4 Beyond the training task in classification: looking at extensions of the notion of generalization

The ability of deep learning architectures to generalize beyond their training tasks is a remarkable achievement which was not obvious, particularly given their numerous layers and millions of parameters. Given their complexity, one might naturally assume that these architectures would learn highly specific parameters tailored to their training task. Nevertheless, the significant advancements in the field of transfer learning have unequivocally demonstrated the capacity to repurpose such architectures for tasks different from the ones they were originally trained on (Zhuang et al., 2020a).

Initially, transfer learning primarily revolved around domain adaptation (Caruana, 1997; M. Wang & Deng, 2018), where the target task remained unchanged, but the target domain is changed. For instance, consider the transition from recognizing handwritten letters of one alphabet to the same but with distinct handwriting styles. Over time, the landscape of transfer learning has evolved, ushering in a new era of possibilities that enable the reuse of pre-trained models for tasks once considered highly specialized and challenging (Jing & Tian, 2020; Pan & Yang, 2010; Zhuang et al., 2020a). This transformative advancement has been particularly prominent in several critical domains, illuminating the remarkable adaptability and versatility of deep learning architectures.

In the realm of Medical Image Analysis, pre-trained models, initially trained on vast datasets of general images, can be fine-tuned for specific medical imaging tasks. This includes the detection of diseases in X-rays, the identification of abnormalities in MRI scans, and the precise segmentation of medical images to aid in accurate diagnoses (Magoulas & Prentza, 1999). In the domain of Natural Language Understanding, models like BERT (Devlin et al., 2018) and GPT (Radford et al., 2018) have undergone pre-training on extensive text corpora, endowing them with the capability to excel in a broad spectrum of NLP tasks. These tasks encompass sentiment analysis, language translation, question-answering, and summarization, often requiring minimal task-specific fine-tuning for outstanding performance(Sun et al., 2019). In the Automotive industry, deep learning models trained on diverse driving scenarios have demonstrated their prowess in the realm of autonomous driving (Bojarski et al., 2016). These models exhibit the

capacity to comprehend intricate traffic patterns, recognize pedestrians, identify lane boundaries, and make real-time decisions, thus significantly advancing the development of self-driving vehicles and enhancing road safety.

Therefore, in this thesis we have centered on extending this notion of generalization and investigating the ability of deep learning architectures to generalize beyond their considered tasks, especially in the case of classification. We envisioned multiple scenarios of generalization, including coarse to fine-grained classification, from one set of inputs and classes to other inputs and classes, or from artificially complexified tasks to simpler ones.

1.4 Related Work

1.4.1 Extension of the notion of generalization

Deep learning architectures have been shown to be highly effective in creating meaningful representations of data for a diverse range of applications, such as image classification (Krizhevsky et al., 2017), regression (Rothe et al., 2018), object detection (Redmon et al., 2016), reconstruction or generation (Goodfellow et al., 2020). Although these representations are inferred on specific tasks and datasets, recent research has shown that they can be utilized for other tasks and data (Bengio, 2012; Caron et al., 2020; Pratt, 1992), illustrating the ability of deep learning architectures to generalize beyond their training task.

A task is typically defined on a specific domain (Atanov et al., 2022; Zhuang et al., 2020b). A natural extension of the definition of generalization is to evaluate the performance of a trained architecture on the same task but with different domains, such as images captured under different lighting conditions or colors (Pratt, 1992). This process is known as domain adaptation, which is extensively covered in the literature (M. Wang & Deng, 2018). In addition to domain adaptation, the robustness literature also covers another type of domain adaptation that involves noisy data.

Another extension is to consider a new type of tasks with or without altering the domain. For example, in (S. Ren et al., 2015) the authors employed an architecture trained on ImageNet with the classification criteria as a feature extractor to perform object detection. In the context of classification, another example is to changes the classes. For instance in Few Shot Learning, a feature extractor is trained on large dataset of classes and then reused to classify unseen classes (Vinyals et al., 2016).

The extensions discussed previously suggest that deep learning architectures are capable of capturing more subtle information during training on specific tasks. In a study by (Gonzalez-Garcia et al., 2018), the authors found that convolutional neural networks (CNNs) can learn to represent objects as a set of semantically meaningful parts, without explicit part annotations during training. Similarly, in (Bau et al., 2017), the authors examined deep architectures used for semantic segmentation and noted the emergence of disentangled representations. These representations signify that the network encodes information in a way that separates and isolates individual attributes or features of different classes, enhancing the model's interpretability.

This ability of architectures to generalize beyond their training task can be leveraged for various purposes (Fei-Fei et al., 2004; Girshick et al., 2014; Radford et al., 2015; Redmon et al., 2016; Zhuang et al., 2020a). For example, it has been demonstrated that very deep architectures with a large number of parameters can achieve higher accuracy in classification tasks (Nakkiran et al., 2021; Simonyan & Zisserman, 2014). However, such

architectures require a substantial amount of data for successful training. Therefore, one can take advantage of large datasets to pretrain an architecture on them and reuse the inferred parameters to work on smaller datasets (Donahue et al., 2014; Girshick et al., 2014; Zeiler & Fergus, 2014). This approach, known as transfer learning, has been successfully applied in various domains, including computer vision, natural language processing, and speech recognition (Devlin et al., 2018; Jia et al., 2018; Simonyan & Zisserman, 2014).

Using an architecture trained on a different task is also a promising approach as it has the potential to lead to better accuracy compared to training directly on the final task. For example, in (Stewart et al., 2023), the authors discuss how deep learning architectures exhibit improved performance on regression tasks when trained using a classification loss (Cross Entropy) as opposed to a regression loss (Mean Square Error). Additionally, training on a very difficult task that includes solving the final task can also lead to better generalization. One example is Mixup (H. Zhang et al., 2017), which generates virtual samples that may not necessarily adhere to the input geometry, resulting in more complex tasks. Another example, is the use of an additional criterion to predict image rotations while training the network, resulting in improved classification accuracy (Gidaris et al., 2018; Mangla et al., 2020).

1.4.2 The difficulty of generalization

While the ability of generalizing beyond the training tasks is a fundamental question that has been always considered in the community of Artificial Intelligence (Atanov et al., 2022; Bengio et al., 2013; Caruana, 1997; Pratt, 1992; Ying et al., 2018), but such capability is not always guaranteed.

In (Tian et al., 2020a), the authors demonstrate that the optimal representation for a given task depends of the task itself. Furthermore, (Shwartz-Ziv & Tishby, 2017) identified two distinct phases that may occur during the training of a deep learning architecture using stochastic gradient descent. In the first phase, the mutual information between the feature space and the output increases. Subsequently, in the second and usually much longer phase, the mutual information between the input and feature space decreases. In other words, the neural network initially discovers the features that can be used to predict the labels before compressing them to retain only the most relevant information for the classification task.

The ability of deep learning architectures to specialize on specific tasks can lead to overfitting on the training task, as demonstrated in (C. Zhang et al., 2021), where the authors showed that a deep learning architecture can perfectly fit random labels. As a result, (Rosenstein et al., 2005) pointed out that transfer learning may hinder the performances when the tasks are too dissimilar. Recently, researchers try to characterize which tasks are compatible with the training task (Atanov et al., 2022).

Moreover, in transfer learning, selecting which part of the architecture to reuse for new tasks is not always obvious (Bordes et al., 2022; Yosinski et al., 2014). Some approaches freeze the feature extractor and train a classifier on it (B. Zhou et al., 2016), while others fine-tune the whole architecture (Krizhevsky et al., 2017; Simonyan & Zisserman, 2014). To identify which layers contain the most general features, (Yosinski et al., 2014) investigated this question and found that early layers capture general patterns while later layers become more specialized to the training task. In (Bordes et al., 2022), the authors cut the last layers of an architecture at different levels to perform transfer learning and demonstrated that this approach can increase performance in transfer.

1.4.3 Representation Learning

Representation learning is a fundamental concept in computer vision, essential for tasks such as image classification, object detection, and segmentation. It involves extracting meaningful and informative features from raw data, allowing machines to understand and interpret visual information effectively. The training process involves various parameters that affect the inferred representation (d'Ascoli et al., 2021; Wightman et al., 2021). These parameters included (not only) the training data (input, labels) (Bagherinezhad et al., 2018; Eshratifar et al., 2021; Touvron et al., 2021; Wu et al., 2017), the criterion (T. Chen et al., 2020; He et al., 2020; Janocha & Czarnecki, 2017b; Stewart et al., 2023), the regularization (Krogh & Hertz, 1991; Loshchilov & Hutter, 2017; Srivastava et al., 2014), the architecture (Caron et al., 2021; He et al., 2016; LeCun et al., 1998; Tian et al., 2020a), the optimizer (Amari, 1993), number of epochs (Goodfellow et al., 2016).

A significant paradigm shift in representation learning for computer vision has been the adoption of Transformers as a viable alternative to CNNs (Caron et al., 2021). While CNNs (LeCun et al., 1998) have excelled at tasks involving spatial hierarchies, Transformers have demonstrated their ability to capture long-range dependencies and global context, making them suitable for a broader range of vision tasks (Caron et al., 2021; Dosovitskiy et al., 2020). Vision Transformers (ViTs) have gained popularity for their impressive performance in image classification and object detection (Caron et al., 2021).

Traditionally, supervised learning criteria, such as cross-entropy loss, have been the norm in computer vision tasks (Goodfellow et al., 2016; Krizhevsky et al., 2017). However, self-supervised learning has gained momentum as an alternative approach (T. Chen et al., 2020; He et al., 2020). Self-supervised learning tasks involve creating a pretext task where labels are generated from the data itself, eliminating the need for manual annotation (Radford et al., 2015). Techniques like contrastive learning and momentum contrast have shown that self-supervised criteria can lead to representations that are as effective as those learned through traditional supervised methods (T. Chen et al., 2020; He et al., 2020; Oord et al., 2018). This shift has reduced the reliance on large datasets, making representation learning more scalable (Bommasani et al., 2021).

Data augmentation, a form of regularization, has become a cornerstone in modern representation learning (Shorten & Khoshgoftaar, 2019; Simard et al., 2001). Data augmentation techniques involve applying random transformations to the training data, such as rotation (Gidaris et al., 2018), scaling (J. Deng et al., 2009), and cropping (Takahashi et al., 2019), to create a more diverse training set. This helps the network generalize better and reduces overfitting. Recent advancements in data augmentation strategies, such as RandAugment, have shown that well-designed augmentations can significantly improve the quality of learned representations (Cubuk et al., 2020).

1.5 Summary of Contributions

My work has centered on extending this notion of generalization and investigating the ability of deep learning architectures to generalize beyond their considered tasks, especially in the case of classification. I have envisioned multiple scenarios of generalization, including coarse to fine-grained classification, from one set of inputs and classes to other inputs and classes, or from artificially complexified tasks to simpler ones.

1.5.1 Learnable Operators for Translation-Invariant Representations on Irregular Domains

The layers and operations employed in the construction of a deep learning architecture play a pivotal role in shaping the characteristics of its representation, particularly in terms of invariance and equivalence to geometric transformations. However, defining these operations becomes considerably more challenging when dealing with irregular or abstract domains, such as graphs. By definition, graphs inherently lack directional information. One notable complication arises from the traditional definitions of graph translations, which rely on the graph's Laplacian matrix, resulting in isotropic operators (Monti et al., 2018; Pasdeloup et al., 2018).

Our research focuses on the development of invariant operators for classification tasks on graphs, which we interpret as graph translations. Drawing inspiration from Convolutional Neural Networks (CNNs), which leverage input label invariance through weight-sharing mechanisms, we propose the formulation of adaptable operators that align with the graph's inherent structure. These operators serve as the foundation for pseudo-convolutions achieved by learning specialized weight-sharing schemes. Consequently, our approach yields architectures that can be trained end-to-end for solving classification tasks, effectively functioning as pseudo-translations or classification-invariant operations.

Utilizing this methodology, we have successfully inferred conventional 2D image translations, including vertical and horizontal translations, as well as compression and dilation operators for grid-graphs. Our experiments involving abstract hyperlink networks further demonstrate the effectiveness of our proposed methodology, highlighting its capacity for generalization and interpretability in the context of 2D images.

This work exemplifies how architectures can be purposefully designed to incorporate invariance properties into their representations, which can be learned based on the specific task at hand. This approach presents a promising avenue for the definition of novel operations and layers that are better suited for handling irregular domains.

1.5.2 Using virtual tasks to obtain a better generalization

While the architectural design plays a pivotal role in achieving generalization, regularization techniques have also been widely employed to enhance generalization performance. Interestingly, some of these techniques can be seen as methods for artificially increasing the complexity of the training task. For example, Mixup (H. Zhang et al., 2017) achieves this augmentation by generating synthetic samples through linear interpolation. Mixup has demonstrated its ability to enhance the generalization performance of deep learning models, leading to improved accuracy. However, it is worth noting that in certain cases, these synthetic samples may not accurately align with the underlying data geometry, leaving room for improvement.

In this study, we introduce an enhanced data augmentation technique called Local Mixup, which builds upon the foundation of Mixup by incorporating locality information. The implementation of these constraints involves the utilization of a weight matrix that assigns weights to interpolations based on the distances between the data points involved in the interpolation. Remarkably, this weight matrix directly corresponds to the adjacency matrix of a graph. We explore various techniques for constructing this graph, including threshold graphs, K-nearest neighbor (KNN) graphs, and decreasing exponential graphs.

We provide a theoretical framework to understand the generalization capabilities of

Mixup, highlighting its role in regularization and its impact on adversarial defenses. Furthermore, we demonstrate that our Local Mixup method can mitigate these effects, effectively striking a balance between bias and variance, resulting in superior generalization performance on standard computer vision benchmarks. In our experiments, Local Mixup consistently outperformed Mixup on well-established datasets such as SVHN, CIFAR-10, and FASHIONMNIST.

Future research directions may involve exploring improved metrics for constructing interpolations, extending our findings to more general scenarios, and investigating the impact of Mixup, particularly on the gradient of the loss function.

1.5.3 Entropy regularization of the feature to enhance transferability of deep learning architecture

The preceding contribution delves into the prospect of enhancing generalization in target tasks by training on a more intricate artificial task. In this work, we reverse this approach and investigate the outcome when training on simpler tasks. Does the model exhibit the capacity to generalize beyond its training tasks? We observe that it indeed possesses the ability to extract information from finer, related tasks through appropriate regularization techniques.

We introduce a novel regularization method named "Feature Information Entropy Regularized Cross Entropy" (FIERCE). Our method aims to amplify the entropy within the feature space of deep learning models while preserving essential information for subsequent tasks. Our experiments underscore the pivotal role of feature space entropy in transfer learning and the model's ability to discern more intricate labels.

Initially, we scrutinize whether a deep learning architecture, initially trained on specific tasks, can extend its generalization beyond them. To investigate, we transform a regularization task involving label refinement into a classification task entailing coarse labels. We illustrate that it is feasible to extract information related to the refined labels from the feature space when the model is trained on coarse labels. However, similar to the findings in (Shwartz-Ziv & Tishby, 2017), we discern two distinct phases in these performances. Initially, the feature space naturally conveys information about the refined labels, but subsequently, this information is progressively erased as the model prioritizes performance on the training tasks.

We assumed that these two phases are intricately linked to the entropy of the feature space. During the training phase with coarse labels, we apply our FIERCE regularization method to encourage the model to augment the entropy of its feature space. This facilitates improved generalization and mitigates early convergence during training. In our experiments, we evaluate our method across various benchmark datasets. The results underscore that our approach outperforms other entropy regularization techniques such as Label Smoothing and Cross Entropy in preserving information within the feature space. Additionally, we present a technique for estimating feature space entropy in a differentiable manner, enabling the training of deep learning models with our proposed regularization.

In summary, our findings advocate for the potential of the FIERCE method in coarse-to-refined classification and transfer learning scenarios. By elevating feature space entropy, our method equips deep learning models with the capability to generalize more effectively and to discern more intricate labels.

1.5.4 A theoretical framework on Transfer in Classification: How Well do Subsets of Classes Generalize?

Finally, our focus shifts to the broader context of transfer learning within the realm of classification. Transfer learning has emerged as a fundamental aspect of deep learning, enabling models to harness knowledge from one task and effectively apply it to another. Nevertheless, despite its extensive application, the underlying mechanisms driving this phenomenon remain enigmatic.

Our particular interest lies in the ability of neural networks to learn from a subset of classes and subsequently generalize to different subsets of classes, and our aim is to characterize this phenomenon comprehensively. We endeavor to provide both theoretical foundations and practical insights, with our primary objectives centered on establishing a rigorous framework for comprehending transferability within this specific scenario and utilizing this framework to elucidate fundamental questions in this domain.

Our pioneering theoretical model, embodied by Hasse Diagrams, serves as an essential tool for establishing an order relationship of learning. This tool enables us to precisely delineate the transferability relationship between distinct sets of classes, shedding light on which classes can generalize to others. Through empirical investigations, we seek to address pivotal questions such as identifying performance indicators when transitioning from one set of classes to another and determining which classes play a pivotal role in achieving optimal generalization and transferability.

We conduct experiments across a range of scenarios, providing empirical evidence and valuable insights into the transferability of deep learning models. Our framework lays the foundation for future research to explore the impact of various training methodologies, construct challenging few-shot learning datasets, and delve into transferability across diverse domains and tasks through the utilization of interpretability techniques.

Chapter 2

Inferring invariant operators for classification tasks

Contents

2.1 Introduction

2.1.1 Deep Learning Architecture: equivariance and invariance

The mathematical expression of layers and models play a crucial role in determining the geometric properties of its feature space, such as invariance or equivariance (Bronstein et al., 2017; T. Cohen & Welling, 2016; Mallat, 2016). For instance, convolution and pooling layers in CNNs (LeCun et al., 1998) combine to produce representations that are invariant to translation and small perturbations. This is why CNNs are widely used in computer vision and classification tasks, as it is recognized that the output of a neural network should be invariant to these operations in order to achieve accurate and robust results (Lenc & Vedaldi, 2015; Simonyan & Zisserman, 2014).

However, achieving invariance or equivariance to specific operators requires a good understanding of the dataset's geometry (T. S. Cohen et al., 2018; Masci et al., 2015a). Convolutional layers, for example, are based on the concept of Euclidean translations, particularly in the vertical and horizontal directions. While these operations are well-defined in the Euclidean case, they are not straightforward on non-Euclidean domains. Several studies have attempted to redefine convolutional layers on non-Euclidean manifolds particularly those with constant curvature, such as hyperspheres and hyperbolic manifolds (Bachmann et al., 2020; Masci et al., 2015a, 2015b), which could be beneficial in various applications. These operators are usually discovered by identifying the invariances and symmetries present in the geometry of the data (T. Cohen & Welling, 2016; Girault et al., 2015; Pasdeloup et al., 2018).

2.1.2 Graph Signal translations

Recently, efforts have been made to unify these investigations under the field of Geometric Deep Learning, which aims to provide a mathematical framework for deriving relevant architectures (Bronstein et al., 2017). The concept of translation, along with convolution, plays indeed a fundamental role in deep learning. It enables us to detect patterns and information within data, regardless of their position or orientation. This property is crucial for various tasks, including feature extraction, object recognition, and signal analysis. Additionally, the efficient parameter sharing in convolution helps optimize the learning process, making it indispensable for handling large-scale inputs like high-resolution images.

Graphs have gained considerable attention due to their diverse range of applications, including computer vision (Monti et al., 2017), social analysis (Backstrom & Leskovec, 2011), traffic prediction (Y. Li et al., 2017) and many others. Nevertheless defining translations becomes challenging when dealing with irregular domains, such as graphs (Girault et al., 2015; Girault et al., 2018; Monti et al., 2018; Sandryhaila & Moura, 2013; Shuman et al., 2012b).

There are fundamental reasons why it is difficult to define translations for graph signals. One of the reasons is that a graph inherently includes only a notion of neighborhood or similarity between its vertices, and no explicit notion of direction. It is worth noting that the construction of the graph can vary depending on the metrics used, and even small changes can significantly impact the derived operators (Shuman et al., 2013).

Moreover, it is not always clear which operation the architecture should be invariant to, as it depends both on the specific signal being processed and the considered task (Bronstein et al., 2017). Most proposed approaches are independent of the signal and primarily focus on the graph structure (Ortega, Frossard, Kovačević, Moura, et al., 2018;

Pasdeloup et al., 2018; Shuman et al., 2013). However, in certain cases, especially with images, this can lead to inconsistent operators (Pasdeloup et al., 2018).

In our work (Baena et al., 2021), we propose a solution that infers graph signal translations by leveraging not only the graph itself but also additional information such as labeled signals on the graph. In our applications, the graph represents the input geometry. For instance, images can be mapped onto grid-graphs, with the pixel values as signals. Thus, it's important to note that the graph nodes do not represent individual samples. For a given task, the graph structure (edges) remains constant but the signal (values of the node) varies for each example.

Our solution builds upon the idea of translation invariance of classification tasks. In more details, given a graph and samples from different classes, our goal is to find operators that are constrained by the graph's structure. These operators should allow us to create efficient deep learning models with shared weights, resulting in high accuracy for the classification task at hand. As such, the inferred operators can be interpreted as transformations that are invariant for the considered task. In the context of regular n-dimensional signals (e.g., grid graphs), we expect these transformations to include conventional translations, such as horizontal and vertical shifts. However, they may also involve other operators like directional dilations or contractions. Notably, this approach doesn't require strict assumptions about graph structure regularity, making it applicable even in abstract domains like relational networks.

But before we dive into the details of our proposed approach, let is introduce the fundamental tools for processing signals on graphs.

2.2 Graph Signal Processing

Recently, the field of Graph Signal Processing (GSP) arose with the aim of generalizing classical harmonic analysis to irregular domains described using graphs (Ortega, Frossard, Kovačević, Moura, et al., 2018). GSP introduces tools to manipulate signals on graphs. These tools include convolutions, filtering, smoothing, translations. The rationale is that such operators are defined by taking into account the graph structure (i.e. the graph edges). In this section, we will show that in the case of an oriented ring graph, the tools defined within the framework of GSP align perfectly with those defined for 1D signals (Sandryhaila & Moura, 2013).

2.2.1 Classical tools in Graph Signal Processing

Notations: For simplicity we will use slicing notation for vector to refers to specific values of a multi-way array, for instance $\mathbf{x}[i, j, k]$. This notation is particularly suited to the field of discrete signal processing.

Let us first define fundamental concepts in GSP (Shuman et al., 2013).

Definition 10: Graph

A ***graph*** is an pair $G = \langle V, E \rangle$, where V is a finite set of ***vertices*** and E is a set of pair of vertices called the ***edges***. Such a graph can be conveniently expressed using its binary ***adjacency matrix*** $\mathbf{A}$ defined as:

$$\mathbf{A}[i, j] = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases} . \tag{2}$$

An illustration of a graph with its adjacency matrix is given in Figure 8. A Graph is said to be ***undirected graph*** when its adjacency matrix is symmetric. One can also define the *degree matrix* of G as as:

$$\mathbf{D}[i,j] = \begin{cases} \sum_{i' \in V} \mathbf{A}[i,i'] & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} . \tag{3}$$

In this work, we are interested in processing signals on graphs. Provided a graph $G = \langle V, E \rangle$, we define a ***graph signal*** as a vector $\mathbf{s} \in \mathbb{R}^V$. In other words, a graph signal is a collection of scalar measures, one per vertex in the considered graph. Of particular interest are Dirac signals which are simple one-hot vectors. As an example, we depict a graph signal on three different graphs in Figure 9.

In the field of spectral graph theory, it is common to also introduce the (combinatorial) ***Laplacian*** of the graph as the matrix defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. Researchers have proposed multiple ways to normalize the Laplacian, such as the symmetrically normalized Laplacian (Ortega, Frossard, Kovačević, Moura, et al., 2018), the random walk Laplacian (Chung, 1997), or normalizing with the largest eigenvalue of the Laplacian (Girault et al., 2018), among others.

The Laplacian or its normalized derivatives are typically used to define the ***graph Fourier transform*** (GFT). The rationale is inspired from the classical Fourier transform which can be defined with the eigenfunctions of the one-dimensional Laplace operator. Analogously, the graph Fourier Transform is defined with the eigenvectors and eigenvalues $(\mathbf{u}_l, \lambda_l)$ of the Graph Laplacian. When the graph is undirect the Laplacian matrix is real and symmetric: the eigenvectors form a complete set of orthogonal vectors on which one can define the GFT: $\hat{f}(\lambda_l) = \sum_{i=1}^{N} \mathbf{f}[i]\mathbf{u}_l[i]$.

It is important to note that the GFT can only be well defined when the Laplacian matrix is diagonalizable. Consequently, in the case of a directed graph, there is no guarantee that the GFT exists. One potential solution is to symmetrize the adjacency matrix, effectively treating the graph as undirected. However, this approach results in the loss of directional information, and the adjacency matrix becomes an isotropic operator. For example, in the case of a ring graph (undirected), utilizing the Laplacian for translation will cause the signal to diffuse equally in both directions.

Furthermore, the GFT depends greatly of the Laplacian, i.e, the graph structure or the possible normalization used for the Laplacian (Ortega, Frossard, Kovačević, Moura, et al., 2018). In Figure 9 we depict the same signal on three different graphs. As observed, the spectral domains of the graphs exhibit significant differences.

2.2.2 Connection between GSP and Discrete Signal Processing

To illustrate the relationship between GSP and Discrete Signal Processing, let us consider a one-dimensional discrete periodic signal $\mathbf{s}$. As shown in Figure 8, this signal can be represented on an oriented Ring Graph (Sandryhaila & Moura, 2013).

Definition 11: Oriented Ring Graph

A ring graph, also known as a circular graph or cycle graph, is a type of mathematical graph that consists of a single closed loop. Formally, a ring graph can be defined as a graph $G = \langle V, E \rangle$, where the number of vertices is equal to the number of edges, and each vertex is connected to exactly one vertex, on its right,

forming a circular structure.

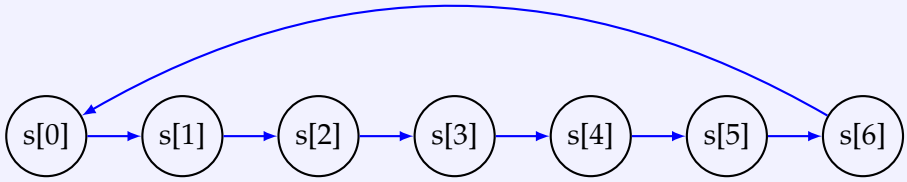


Figure 8 – Example of an Oriented Ring Graph representing a periodic signal $\mathbf{s}$.

The adjacency matrix of a ring graph can be written as $\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & & & & 1 \\ 1 & 0 & 0 & \cdots & 1 \end{pmatrix}$.
Later on, we will refers to this matrix as $\mathbf{J}$. This matrix is extremely helpful since any circulant matrix can be written as a polynomial in $\mathbf{J}$.

For a temporal signal, the adjacency matrix is sometimes called the *shift operator* because multiplying a signal $\mathbf{s} = [s_1, s_2, \cdots, s_N]^T$ with the matrix $\mathbf{J}$ shifts the signal: $\mathbf{s}^T\mathbf{J} = [s_N, s_1, \cdots, s_{N-1}]^T$. Similarly, the opposite direction is obtained by changing the orientation of the ring. In Euclidean geometry, such operations are translations.

Translations are among the most fundamental transformations in signal processing (Oppenheim, 1999). They are often used as a basic building block to define convolutions, Fourier transform, filters and related tools. In machine learning, they can be exploited to define ad-hoc operators that benefit from the underlying simple regular structure of processed signals, such as in the case of Convolutional Neural Networks (CNNs).

In fact, we will demonstrate that the discrete one-dimensional convolution can be defined and derived from the adjacency matrix $\mathbf{J}$ of the oriented ring graph.

Definition 12: Discrete 1-d Convolution

Let us consider two periodic signals $\mathbf{x}$ and $\mathbf{y}$ of periodicity N. The 1-d convolution is defined as:

$$(\mathbf{x} * \mathbf{y})[k] = \sum_{i=1}^{N} \mathbf{x}[i] \times \mathbf{y}[l-i],$$
$$\mathbf{y}[i] = \mathbf{y}[i \mod N]$$

The previous equation can be written as the product of $\mathbf{x}$ with a circulant matrix $\mathbf{T}$ or polynomial expression in the matrix $\mathbf{J}$:

$$\mathbf{x} * \mathbf{y}[k] = \begin{pmatrix} y[1] & y[n] & y[n-1] & \cdots & y[2] \\ y[2] & y[1] & y[n] & \cdots & y[3] \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ y[n] & y[n-1] & y[n-2] & \cdots & y[1] \end{pmatrix} = \sum_{k=i}^{N} (y[i]J^{i-1})\mathbf{x}. \tag{4}$$

In other words, the adjacency matrix of the ring graph serves as the core operator

of one-dimensional convolution. Another interesting observation is that the graph Fourier transform of the oriented ring graph exactly matches the one-dimensional discrete Fourier Transform. Specifically, its adjacency matrix $\mathbf{J}$, is diagonalizable, and its eigenvalues coincide with those of the one-dimensional Laplacian.

Unfortunately, this matching does not hold in higher dimensions. While a 2-dimensional periodic signal can be easily represented on a torus graph (Sandryhaila & Moura, 2013), the adjacency matrix and its Graph Fourier Transform do not correspond to the 2-dimensional translation and Fourier Transform (Girault et al., 2015).

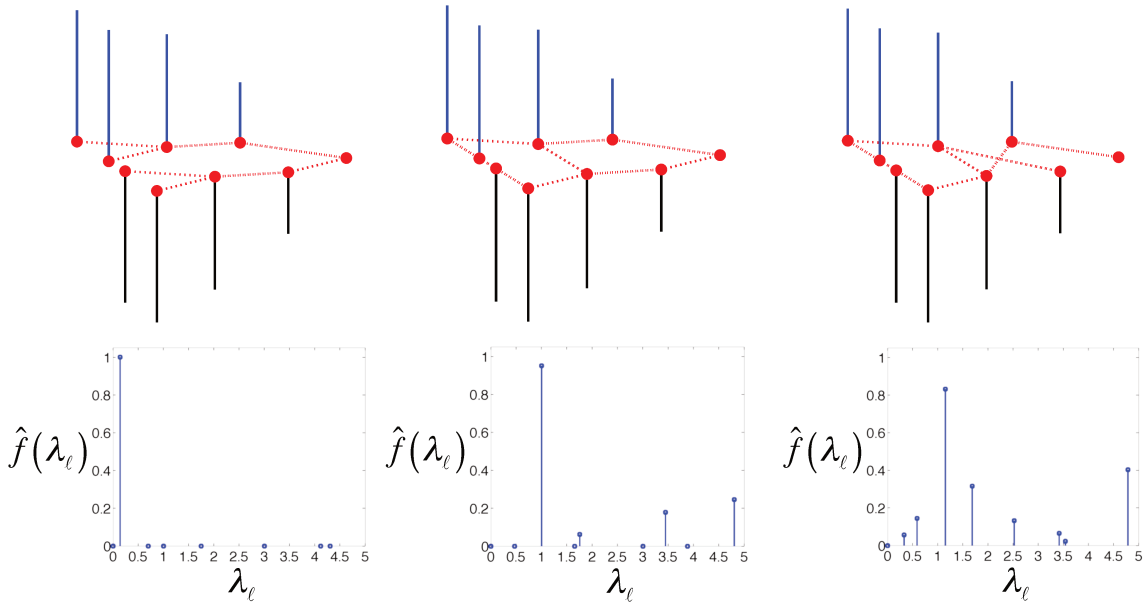


Figure 9 – Illustration of a graph signal signal $\mathbf{f}$ on three different graphs with the same set of vertices, but different edges. The top row shows the signal in the vertex domains, and the bottom row shows the signal in the respective graph spectral domains. One can note the significant differences between the distributions of the eigenvalues. On the left, the graph exhibits low frequencies, indicating smooth variations of the signal across clusters. On the right, the spectral domain is more heterogeneous, as significant variations in the signal are observed between neighboring vertices

2.3 Translation in Signal Processing and GSP

2.3.1 Graph Signal Translation

Among the numerous tools that were introduced in this field, translation has attracted a lot of attention (Shuman et al., 2013). Defining translations for graph signals poses challenges because graphs can represent both highly regular structures (such as grid graphs) and abstract ones (such as social networks). Consequently, the definition of translations and, by extension, harmonic operators must be adaptable and appropriate for these diverse domains.

The initial approach to defining translations involves working within the vertex domain. This is done by using a shift operator derived from either the adjacency matrix or its different Laplacians, as discussed in previous research (Sandryhaila and Moura, 2013; Shuman et al., 2013). As we observed in the case of 1-d signal, the adjacency matrix of the oriented ring graph allows to retrieve the classical translation, but such

analogy fails in higher dimensions.

Another solution proposed in the early days of GSP, was to define translations on top of convolutions. In (Hammond et al., 2011; Shuman et al., 2012a) the authors defined convolutions in three steps: first computing the GFT of the considered signals, then pointwise multipliing their spectral coordinates, and finally performing an inverse GFT on the resulting vector. Graph signal translations can then be obtained by convolving signals with a Dirac.

But the authors of (Girault et al., 2015) point out that translations defined through the spectral domain are not isometric. They introduce alternative definitions using complex exponentials of the Laplacian matrix. Problematically, it has been shown that these operators do not generalize well to classical circular translations on signals defined on torus or grid graphs (Pasdeloup et al., 2018).

While these previous rationales (Hammond et al., 2011; Sandryhaila & Moura, 2013; Shuman et al., 2012a) do not match the classical definition of signal processing, they are still widely used to defined Graph Neural networks (Scarselli et al., 2008), especially, graph spectral neural networks (Bruna et al., 2013) or graph convolutional neural networks (Kipf & Welling, 2016) (GCN).

Let us introduce the concept of graph layer convolutional layers, which forms the fundamental core of GCN. Consider a graph represented by its adjacency matrix $\mathbf{A}$ and degree matrix $\mathbf{D}$. A graph layer convolutional layer can be defined as $\sigma(\mathbf{D}^{-1/2}\mathbf{A}(\mathbf{A} + \mathbf{I}\mathbf{D}^{-1/2}\mathbf{H}\mathbf{W}))$, where $\mathbf{H}$ represents a graph signal and $\mathbf{W}$ denotes the layer weights. Notably, this definition exhibits similarities with conventional layers, but the operations (weights) performed are specifically constrained by the adjacency matrix of the graph. Indeed, the original definition of the adjacency or Laplacian matrix leads to an isotropic operator, causing the signal to diffuse to all neighboring nodes. This diffusion behavior is not always desirable in graph signal processing tasks, as it may result in the loss of localized information and hinder the ability to capture meaningful patterns or structures within the graph (Pasdeloup et al., 2018).

As an attempt to propose nonisotropic operators, the authors of (Monti et al., 2018) aim at identifying directions or relevant graph motifs. These motifs represent meaningful connectivity patterns, e.g triangle motifs which are crucial for social networks (Benson et al., 2016). Once a set of motifs is chosen, nonisotropic Laplacians are defined for each one. Convolutions are then defined as multivariate polynomials of the Laplacian matrices and can be used to define graph neural networks. Two key issues with this methods are the huge amount of parameters it relies upon and the difficulty of choosing relevant motifs.

Using a completely different approach, the authors in (Pasdeloup et al., 2018) defined translations of graph signals directly in the vertex domain (without using the GFT), thus providing an actual generalization of classical tools. In their work they characterize translations as functions ϕ, defined from a subset of vertices V', that are i) injective ($\phi(v) = \phi(v') \Rightarrow v = v', \forall v, v' \in V'$), ii) edge-constrained ($(v, \phi(v)) \in E, \forall v \in V'$) and iii) neighborhood-preserving ($(v, v') \in E \Leftrightarrow (\phi(v), \phi(v')) \in E, \forall v, v' \in V'$). Injectivity and neighborhood-preservation are key characteristics to ensure the matching with regular translations, but they are poorly suited for abstract graph structures such as social networks. Moreover, this approach comes with a large computational complexity, and struggles with abstract and irregular graph structures. Another issue is that the methodology relies only on the graph structure and therefore it is completely agnostic of the signal's nature.

In (Vialatte et al., 2017), the authors introduce pseudo-convolutions for deep neural networks that can be seen as implementing the edge constraint previously introduced. Namely, they introduce a tensor $\mathbf{S}$ and a vector $\mathbf{w}$. The binary tensor $\mathbf{S}$ is of dimension $N \times N \times K$, where N is the number of vertices in the considered graph and K is a hyperparameter. Moreover, $\mathbf{S}[i, j, k]$ is zero if $(i, j) \notin E$, and $\mathbf{S}[i, j, :]$ contains at most one nonzero entry. The vector $\mathbf{w}$ contains K coordinates. The tensor-matrix product along the third mode of $\mathbf{S}$ by $\mathbf{w}$, denoted as $\mathbf{S} \times_3 \mathbf{w}$ creates a $N \times N$ matrix $\mathbf{W}$ that can be seen as a weighted version of the adjacency matrix $\mathbf{A}$ of the considered graph. The authors show that for particular choices of $\mathbf{S}$, they can retrieve classical convolutions for regular grid graphs. More generally, slices $\mathbf{S}[:, :, k]$ can be interpreted as graph signal translations. We then decide to use a weight-sharing scheme and we propose to infer the tensor $\mathbf{S}$ using both the graph structure and a set of labeled signals.

2.3.2 Equivariant layers through weight sharing

Weight sharing is a technique commonly employed in Convolutional Neural Networks where the same set of parameters is utilized across different layers (LeCun et al., 1998), resulting in a reduction in the number of learnable parameters. Specifically applied in CNNs for image processing tasks, weight sharing facilitates the processing of images using shared filters, thereby enhancing the network's generalization capabilities by capturing local patterns and enabling the acquisition of more abstract and general representations (Krizhevsky et al., 2017). Additionally, weight sharing promotes translational invariance within CNNs, as it compels the network to learn pattern recognition irrespective of the pattern's position, ensuring consistency across different locations in an image (T. Cohen & Welling, 2016).

Recently, the authors of (T. Cohen & Welling, 2016) have proposed Group Equivariant Convolutional Networks G-CNNs, an extension of CNN. They highlight that traditional CNNs lack the ability to retain spatial relationships under general transformations such as rotations and reflections, which limits their applicability in certain domains. To address this limitation, they proposes G-CNN to maintain spatial relationships by incorporating the concept of group theory to learn equivariant representations.

G-CNNs utilize group convolutional filters that leverage weight sharing across different spatial locations and orientations. The sharing of weights enables the network to learn shared patterns that remain consistent across different transformations. Consequently, G-CNNs can produce more robust and invariant representations. Similarly, in another study (Yeh et al., 2022a), the authors propose two parameter-sharing schemes that achieve equivariance with respect to any discrete group-action.

One limitation of the aforementioned works (T. Cohen & Welling, 2016; Yeh et al., 2022a) is that they rely on prior knowledge of the equivariance properties of the data. However, such knowledge may not be readily available, particularly when encountering a new domain. In contrast, posterior to our work (Yeh et al., 2022b) proposes a method to discover equivariant layers by solving an optimization problem over a model's parameter-sharing schemes.

2.4 Problem Statement and Methodology

The motivation for employing Convolutional Neural Networks (CNNs) lies in harnessing the inherent invariance of input labels to translations (LeCun et al., 1998). This is accomplished through the implementation of weight sharing mechanisms. To elaborate

further, translations serve as the basis for the definition of convolutions. When convolutions layers are integrated with pooling operations, they have the capacity to generate representations that exhibit invariance with respect to translations. As a result, CNNs produced in this manner can achieve significant improvements in accuracy when compared to translation-agnostic architectures like multi-layer perceptrons (Krizhevsky et al., 2017; LeCun, 2015a).

The key idea of our proposed methodology is to reverse this reasoning. **Namely, we propose to define learnable operators that are aligned with the graph structure, from which we build pseudo-convolutions by learning tailored weight sharing schemes**. When combined with pooling operations, this results in architectures that can be seamlessly trained from end to end to address classification tasks. Once we have identified a network exhibiting acceptable performance, we can subsequently integrate the learned operators into the architecture as pseudo-translations, or more broadly, as operations invariant to classification.

To delve deeper into the subject, let's explore a straightforward example involving a ring graph, with an associated adjacency matrix denoted as $\mathbf{A}$. As previously elucidated, this graph can accommodate a periodic graph signal, which we'll denote as $\mathbf{s}$. To keep things uncomplicated, we assume that $\mathbf{s}$ is of dimension $N = 4$. Within this example, we can defined $k = 3$ translations and each of which can be represented by a matrix $(\mathbf{T}_k)$, where $\mathbf{T}_k \in \mathbb{R}^{N\times N}$. The translations are $T_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ as the identity, $T_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ and $T_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$ circular translations corresponding to the two orientations of the ring. We define a tensor $\mathbf{S} \in \mathbb{R}^{N\times N\times K}$ by concatenating the matrices $(\mathbf{T}_k)_k$. Additionally, we introduce a convolutional kernel $\mathbf{w}$, with its elements indexed by the K translations. Then it holds that:

$$\begin{aligned} \mathbf{S} \times_3 \mathbf{w} &= \sum_k \mathbf{w}[k]\mathbf{S}[:,:,k] \\ &= \sum_k \mathbf{w}[k](\mathbf{T}_k) \\ &= \begin{pmatrix} w_0 & w_2 & 0 & w_1 \\ w_1 & w_0 & w_2 & 0 \\ 0 & w_1 & w_0 & w_2 \\ w_2 & 0 & w_1 & w_0 \end{pmatrix}. \end{aligned} \tag{5}$$

Indeed, we can identify this as a Toeplitz circulant convolution matrix. These equations can be readily extended to any regular nD graph and, with appropriate constraints on the structure of $\mathbf{S}$, to any arbitrary graph as well. Consequently, we can define the graph convolution operation $\star$ as follows:

$$\mathbf{s} \star \mathbf{w} = \mathbf{s}^\top (\mathbf{S} \times_3 \mathbf{w}). \tag{6}$$

Our approach involves the optimization of deep neural networks with the dual objective of classifying graph signals and learning the matrices $(\mathbf{T}_k)_k$ concurrently. This learning process adheres to a crucial constraint: $\mathbf{T}_k[i,j] \neq 0$ if and only if $\mathbf{A}[i,j] \neq 0$,

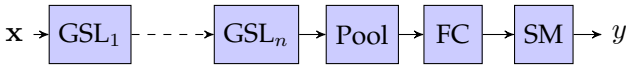


Figure 10 – Depiction of the used deep learning architecture. GSL stands for Graph-Signal Layer, Pool for a global average pooling, FC for a fully connected layer and SM for a `softmax`.

where $\mathbf{A}$ signifies the graph adjacency matrix. In essence, the matrices $(\mathbf{T}_k)_k$ represent transformations that are bound by the edges of the graph, ensuring edge-constrained transformations.

2.4.1 Problem Statement

As previously mentioned, our objective is to learn a set of operators that exhibit invariance properties for classification tasks. In practice, this set is represented by a tensor $\mathbf{S}$ in $\mathbb{R}^{d\times d\times k}$, where d represents the dimension of the output graph signal (the number of nodes), k is the number of learned operators, and the individual slices $\mathbf{S}[:,:,k]$ correspond to invariant operators. These learned operators form the core foundation of Graph-Signal Layers (GSLs), serving as the primary building blocks for these layers, which, in turn, are employed in deep learning architectures.

Similar to convolutional layers where the weight tensor $\mathbf{W}$ implements a convolution operation, GSL layers provide a generalization of convolutional layers. In GSL layers, a signal $\mathbf{s}$ is transformed as follows:

$$\mathbf{s} \mapsto \sigma(\mathbf{s}^\top(\mathbf{S} \times_3 \mathbf{w}))$$

where the slices $\mathbf{S}[:,:,k]$ are subject to edge constraints: $\mathbf{S}[i,j,k] \neq 0$ if and only if $\mathbf{A}[i,j] \neq 0$, with $\mathbf{A}$ representing the adjacency matrix.

Now, let's delve into the process of learning the tensor of operators $\mathbf{S}$. To simplify the explanation, we will describe the procedure for tensors with only one filter, i.e., a single $\mathbf{w}$. It's crucial to note that all the equations presented here can be extended to the case of multiple filters, which essentially involves adding an additional dimension to all tensors and computations.

In the context of deep neural networks, we can represent the mapping from input to output as a function denoted as f. This function f is constructed by combining elementary functions known as layers. These layers typically take the form of:

$$\mathbf{x} \mapsto \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}),$$

where $\mathbf{W}$ represents a weight matrix, $\mathbf{b}$ is a bias vector, and σ is a nonlinear function that is usually applied component-wise. The weight matrix $\mathbf{W}$ and its corresponding bias vector $\mathbf{b}$ are the trainable parameters denoted as θ of the network.

In classification settings, the goal is to train the function f to map raw inputs (e.g., images) to their respective classes. This typically involves using two datasets: a training dataset denoted as $\mathcal{D}_{\text{train}}$, which is used to learn the parameters, and a validation dataset used to evaluate the performance of the trained function f in correctly classifying previously unseen inputs. The network function f concludes by applying a `softmax` operator.

The most common approach for training a classifier involves using a cross-entropy loss function $\mathcal{L}$. Given a pair $(\mathbf{x}, y) \in \mathcal{D}_{train}$, where $\mathbf{x}$ is an input and y is its corresponding

output label, the loss function is defined as:

$$\mathcal{L}(\mathbf{x}, y) = -\log(f(\mathbf{x})[y]).$$

Then, we aim to optimize the following equation:

$$\arg\min_{\theta} \sum_{(\mathbf{x},y)\in\mathcal{D}_{train}} \mathcal{L}(\mathbf{x}, y).$$

In practice, optimization is accomplished through the utilization of various Stochastic Gradient Descent (SGD) algorithm variants. Now, consider a scenario involving a deep neural network function denoted as f with parameters $\mathbf{S}, \omega, \theta$, which encompasses Graph Shift Layers (GSLs). In this context, $\mathbf{S}$ represents the graph transformations, implicitly employed in Convolutional Neural Networks (CNNs), ω signifies the parameters associated with GSLs, and θ encompasses the remaining parameters, such as those related to fully connected layers. Our goal now is to determine the solution to the following optimization problem:

$$\arg\min_{\mathbf{S},\omega,\theta} \sum_{(\mathbf{x},y)\in\mathcal{D}_{train}} \mathcal{L}(\mathbf{x}, y).$$

We are particularly concerned with solutions in which $\mathbf{S}[i, :, k]$ takes on the characteristics of one-hot vectors, facilitating the interpretation of the slices $\mathbf{S}[:, :, k]$ as pseudo-translations. In the following subsection, we will offer a more thorough explanation of our approach to enforcing this constraint.

2.4.2 Temperature-Based Optimization for One-Hot Constraints

As outlined in the introduction, Convolutional Neural Networks (CNNs) with pooling layers possess the advantage of yielding translation-invariant decisions. However, applying pooling to graph signals can be challenging due to the necessity of computing graph downsampling (Shuman et al., 2013). To address this, we employ a simple workaround wherein we exclusively perform a single pooling operation at the penultimate layer of our proposed architecture, just before the final fully connected layer. This pooling operation is global, causing a complete reduction in the graph dimension: all vertex values are averaged into a single value for each filter considered. An illustration of the proposed architecture can be found in Figure 10.

Optimizing deep learning architecture over a discrete domain is a complex endeavor, primarily because it entails binary matrix constraints that are not easily enforced (Courbariaux et al., 2016). Given our objective of obtaining one-hot vectors, which aligns with the approach in (Hacene et al., 2019), we adopt a similar strategy. Specifically, we apply a $\texttt{softmax}$ operator to the second dimension of $\mathbf{S}$, with a variable temperature t ($\mathbf{x} \mapsto \texttt{softmax}(\mathbf{x}/t)$). This temperature begins with an initial value of t_{init}, typically large. In this case, the $\texttt{softmax}$ operator effectively maintains the lines $\mathbf{S}[i, :, k]$ constant where defined (recall that $\mathbf{S}[:, :, k]$ is edge-constrained). Towards the end of training, the final temperature is reduced to t_{final}, typically small, causing the $\texttt{softmax}$ operator to approximate a regular $\max$ operator, thereby transforming the lines $\mathbf{S}[i, :, k]$ into one-hot vectors.

We experimented various strategies to interpolate the temperature between t_{init} and t_{final}. Our most consistent results were obtained using an exponential interpolation: $t(s) = t_{init}\left(\frac{t_{final}}{t_{init}}\right)^{\frac{s}{s_{total}}}$, where s is the current step in the training phase, and s_{total} is

the total number of steps used for training. At the end of the training process, we use a temperature of 0 to interpret the slices of $\mathbf{S}[:, :, k]$ as pseudo-translations. In the next section, we present experiments on toy and real datasets.

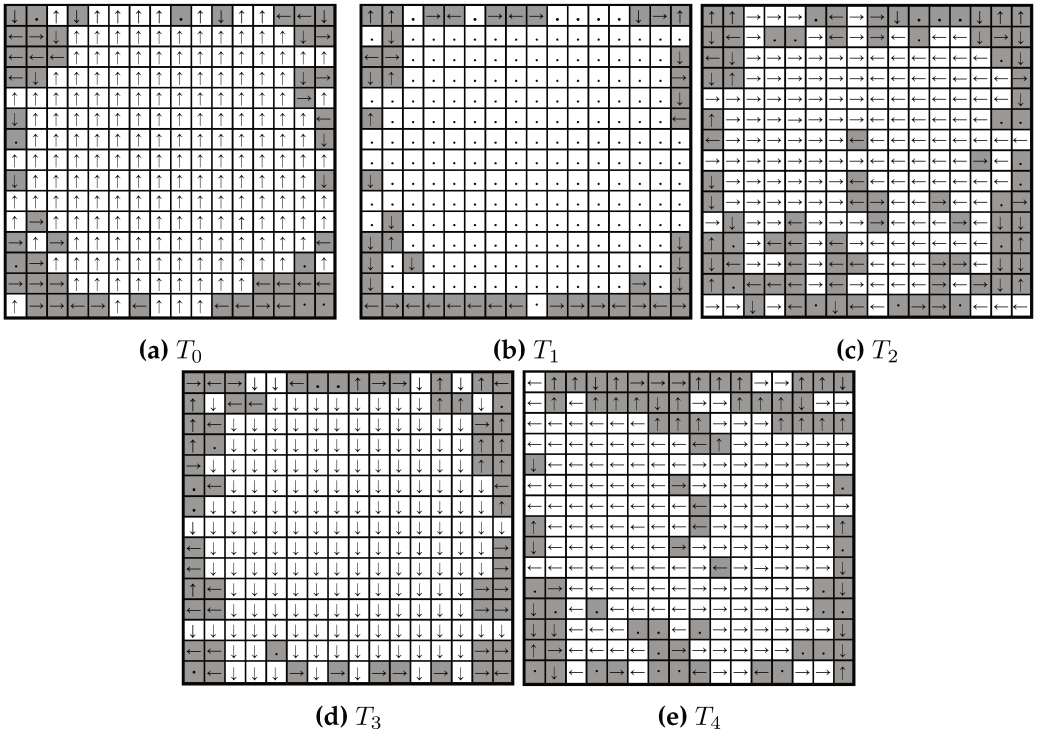
2.5 Experiments

In this section, we conduct experiments across multiple graph types, ranging from regular structures (e.g., images supported on 2D grid graphs) to more abstract ones (e.g., hyperlink networks). Our method is assessed using two distinct datasets: CIFAR-10 (Krizhevsky, Hinton, et al., 2009) and webKB (Pei et al., 2020). CIFAR-10 is a classification dataset comprising images, each composed of 32×32 pixels and utilizing three primary color channels, distributed among 10 distinct classes. WebKB, on the other hand, consists of 877 web pages sourced from computer science departments of universities, categorized into one of five classes: student, project, course, staff, and faculty. The dataset is characterized by word-based feature vectors, each of dimension 1703, corresponding to the content of the webpages, as well as a hyperlink graph representing the relationships between these pages. This dataset is commonly employed in semi-supervised classification scenarios, where only a subset of the webpages are labeled.

2.5.1 Sanity check with regular grid graphs

In our initial experiment, our objective is to assess the capability of our proposed method to recover classical translations when dealing with 2D signals and structures. To accomplish this, we employ the CIFAR-10 dataset, which has been downscaled to 16×16 pixel images. We represent the image signals on a regular grid graph. In greater detail, the grid graph is constructed such that each vertex corresponds to a pixel, and each pixel is connected to its four immediate neighbors via edges.

Figure 11 illustrates the outcomes of our proposed method. We represent inferred pseudo-translations $\mathbf{T}$ as grids of size 16×16. Each vertex is denoted by an arrow pointing to the neighboring vertex it connects to through $\mathbf{T}$ (it is important to note that inferred pseudo-translations are edge-constrained, ensuring the validity of this representation). For each $\mathbf{T}_k$, we highlight the vertices corresponding to the predominant direction. Interestingly, we observe that $\mathbf{T}_0$ and $\mathbf{T}_3$ tend to approximate conventional translations. It's worth noting that $\mathbf{T}_1$ is nearly an identity function. Surprisingly, $\mathbf{T}_4$ and $\mathbf{T}_2$ resemble horizontal dilation and compression, respectively. Importantly, such transformations remain valid within our framework and are typically invariant for the classification task at hand.

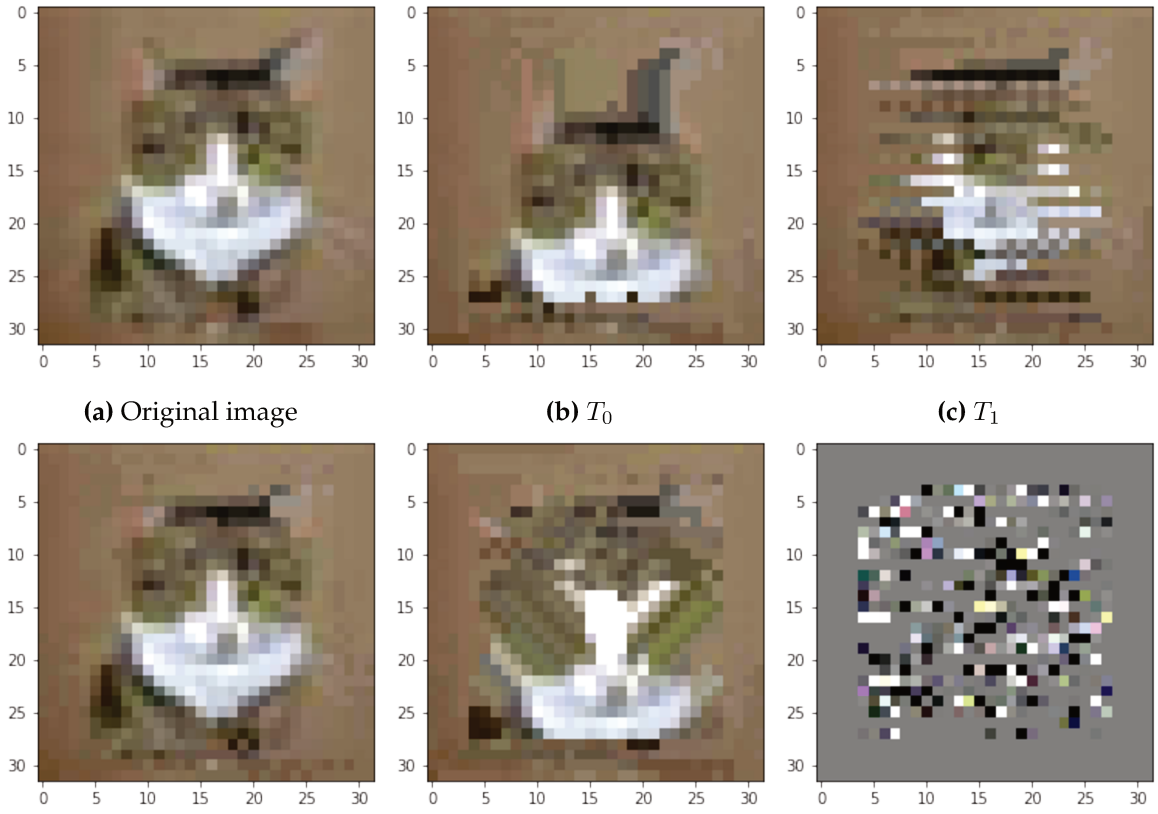


(a) T_0 **(b)** T_1 **(c)** T_2

(d) T_3 **(e)** T_4

Figure 11 – Depiction of inferred pseudo-translations when considering the CIFAR-10 dataset on a regular 2D grid graph.

2.5.2 Experiments with a near-regular inferred graph structure



(a) Original image **(b)** T_0 **(c)** T_1

(d) T_2 **(e)** T_3 **(f)** GSP translation

Figure 12 – Inferred translations $T_0, T_1, \ldots, T_3$ and comparison with the translation defined in (Hammond et al., 2011) on a near-regular graph.

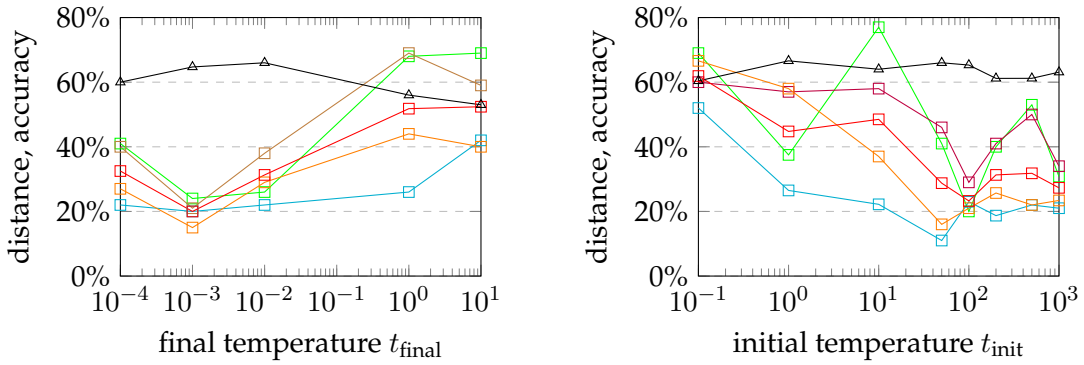
2.5.3 Experiments with hyperlink networks

To demonstrate the versatility of our approach, we conducted an experiment using the WebKB dataset. Given the absence of a more appropriate method for evaluating the transformations obtained, we compared the accuracy achieved using our proposed methodology with that of a conventional method from the literature, the graph convolutional neural network (GCN) (Kipf & Welling, 2017). We computed the average accuracy over 10 different splits of training, validation, and test sets. GCN yielded an average accuracy of 86%, while our method achieved 87%. It's noteworthy that GCN and our proposed methodology deliver similar performance, despite their distinct approaches: GCN relies on isotropic diffusion of signals, whereas our focus lies in inferring directional edge-constrained translations. Furthermore, unlike GCN, our approach is not primarily designed to optimize classification performance but to infer meaningful edge-constrained transformations.

2.5.4 Influence of hyperparameters

Finally, in a final set of experiments, we explore the sensitivity of the proposed method concerning the hyperparameters t_{init} and t_{final}. In Figure 13a, we hold t_{init} constant while varying t_{final}, while in Figure 13b, we keep t_{final} fixed and vary t_{init}. Throughout these experiments, we assess the influence of the initial and final temperatures on both the network's accuracy and the transformations generated. The "distance" met-

ric quantifies the number of disparities between the obtained transformations and the closest 2D translation, dilation, or contraction. For this assessment, we employ the CIFAR-10 dataset, assuming that the images are based on the grid-graph structure. As the results indicate, the method exhibits a notable robustness to changes in these hyperparameters.



(a) Impact of t_{final}.

(b) Impact of t_{init}.

Figure 13 – Comparison of the impacts of t_{final} and t_{init} on accuracy and distance of the obtained translation: identity (orange), up (green), down (purple), dilation (blue), and the average distance (red).

2.6 Conclusion

In this chapter, we've introduced an novel deep learning method that allows us to learn graph signal translations using both a graph structure and a set of labeled signals. Through experiments, we've demonstrated the method's effectiveness in retrieving conventional 2D translations from regular images. Additionally, we conducted experiments on an abstract hyperlink network and achieved performance comparable to state-of-the-art methods. Further research include exploring alternative approaches for learning the translations, different criteria, datasets and refining the selection of hyperparameters. From a broader perspective, these findings pave the way for extending this methodology to derive invariant operators suited for irregular domains, tailored to address specific target tasks. This perspective work aligns with previous research (d'Ascoli et al., 2021), which suggests that fixing learned operators could potentially enhance accuracy by using a task-specific architecture.

In the upcoming chapter, we'll delve into another avenue for improving generalization: by modifying the training process itself. Typically, this involves the inclusion of regularization techniques and data augmentation. Our focuswill be on *Mixup*, a method that generates new data samples through linear interpolation.

Chapter 3

Enhancing Generalization through Local Mixup: leveraging locality to prevent manifold intrusion

Contents

3.1 Introduction

3.1.1 Regularization methods to enhance generalization

As discussed in chapter III, the architecture and the layers used to build the model have a significant impact over the generalization achieved on the target tasks. But, another way to enhance generalization and address overfitting issues is to rely on regularization techniques that have been widely employed in the field of deep learning (Goodfellow et al., 2016). In this chapter, our primary focus will be on Mixup, a technique designed to generate virtual samples and improve generalization, as outlined in (H. Guo et al., 2019). This method can be viewed as a regularization technique that artificially introduces complexity into the training task.

Regularization techniques can be categorized into two categories: data-independent or data-dependent (H. Guo et al., 2019). Typically, data-independent regularization techniques constrain the model by penalizing the norm of the parameters, for instance through weight decay (Loshchilov & Hutter, 2017). Another frequently employed technique is dropout, which involves the random omission of the weights during the training process, as described in (Srivastava et al., 2014).

Among data-dependent regularization techniques, data augmentation plays a significant role (H. Guo et al., 2019). Data augmentation artificially generates new samples, thereby increasing the size of the training dataset $\mathcal{D}_{train}$ (Simard et al., 2001). It can be applied to either the outputs (Sukhbaatar et al., 2014) or the inputs (Cubuk et al., 2018; DeVries & Taylor, 2017; He et al., 2016; Krizhevsky et al., 2017; Yun et al., 2019; R. Zhang et al., 2016). In computer vision, for example, it is very common to generate new samples using class-invariant transformations (He et al., 2016; Krizhevsky et al., 2017).

Data augmentation can be viewed as a method to introduce complexity into training tasks by enforcing certain invariance properties through geometric transformations. While it is evident that certain transformations –particularly in computer vision applications– are beneficial and contribute to improved generalization, the advantages of other augmentation methods remain somewhat mysterious (Verma et al., 2019).

3.1.2 Mixing method as data augmentation

Recently, data-dependent methods incorporating some form of *mixing* methods have emerged (J. Chen et al., 2020; Chou et al., 2020; DeVries & Taylor, 2017; Hendrycks et al., 2019; Kim et al., 2020; Z. Liu et al., 2021; Rame et al., 2021; Verma et al., 2019; Yin et al., 2021; Yun et al., 2019; H. Zhang et al., 2017). These methods typically involve mixing two or more inputs along with their corresponding labels. The pioneering method in this category is *Mixup* (H. Zhang et al., 2017), where mixed samples $(\tilde{x}, \tilde{y})$ are generated by linear interpolations between pairs, i.e., $\tilde{\mathbf{x}}_{i,j,\lambda} = \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j$ and $\tilde{\mathbf{y}}_{i,j,\lambda} = \lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_j$, where $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_j, \mathbf{y}_j)$ are training samples, and $\lambda \in [0, 1]$. *Mixup* has been shown to improve generalization of state-of-the-art models on ImageNet, CIFAR10, speech, even tabular datasets (H. Zhang et al., 2017) and successfully in the context of few shot learning (Dhillon et al., 2019; Fei-Fei et al., 2006).

3.1.3 Side Effect of mixup

By using linear interpolation, *virtual* samples can sometimes contradict one another or even generate inputs that are outside the input distribution. These spurious examples are more likely to occur when the input signals are situated on a complex topology.

For instance, directly interpolating pixels in natural images is unlikely to produce a coherent or meaningful image. This phenomenon has been recently described in (H. Guo et al., 2019), where the authors use the term *manifold intrusion*. As such, it is not clear if *Mixup* is always desirable. More generally, the question arises of whether *Mixup* could be constrained to reduce the risk of generating such spurious interpolations.

By definition, *Mixup* encourages the model f to associate linearly interpolated inputs with the corresponding linearly interpolated outputs (H. Zhang et al., 2017). But, several authors have questioned the positive effect of this linear behavior between samples and have aimed to theoretically and empirically explain the behavior of *Mixup*.

For example, in (Carratino et al., 2020) the authors showed that *Mixup* can be interpreted as a combination of a data transformation and a data perturbation, where a first transformation shrinks both inputs and outputs towards their mean, and a second transformation applies a zero-mean perturbation. The proof was given by reformulating the *Mixup* loss. In (Gyawali et al., 2020), they highlighted that *Mixup* impacts the Lipschitz constant L of the gradient of the model function.

In (Chidambaram et al., 2021), the authors identify cases where *Mixup* fails to minimize the empirical risk (ERM). They also proved that in certain situations, *Mixup* can achieve better generalization in terms of margin, while in other situations, there is no difference.

Posterior to our work (Baena et al., 2022b), the authors in (Oh & Yun, 2023) show that the minimization of the logistic loss to estimate the optimal Bayes classifier requires an exponentially increasing number of data, while *Mixup* grows quadratically. In (Park et al., 2022), the authors propose a unified theoretical analysis of mixed sample data augmentation (MSDA), such as Mixup and CutMix. They demonstrate that regardless of the specific mixing strategy, MSDA serves as a regularization technique for the training loss and the first layer parameters. They also prove that MSDA improves adversarial robustness and generalization compared to vanilla training.

In addition to *Mixup*, several works have proposed improvements using various approaches. For example, in (Chou et al., 2020), the idea is to use different λ_x and λ_y values to mix the inputs and outputs; in (Z. Liu et al., 2021; Rame et al., 2021; Yun et al., 2019), the authors explore the use of other (nonlinear) interpolation methods., while approaches described in (J. Chen et al., 2020; Greenewald et al., 2021; Yin et al., 2021) extend the mixing to more than two elements is proposed.

3.2 Problem Statement and proposed Approach

3.2.1 Problem Statement

We aim to investigate the reasons behind the improved generalization achieved through *Mixup* during training and address the phenomenon known as *manifold intrusion* introduced in (H. Guo et al., 2019). This phenomenon occurs when Mixup generates virtual samples that extend beyond the expected boundaries of the manifold domain for a given class. This scenario is illustrated in Figure 14, where the inter-class interpolations between samples from the red class intersect and conflict with the ground truth of the blue class.

We explore the possibility of enhancing the benefits of this regularization by introducing a modified approach called *Local Mixup*. In this approach, we assign weights to the virtual samples within the training loss. The weight of each virtual sample is based

on the distance between the endpoints of the corresponding segment, represented by the input samples $(\mathbf{x}_i, \mathbf{x}_j)$. By incorporating this weight assignment, we can prevent the interpolation between samples that are too far apart in the input domain. Naturally, we anticipate that this methodology will yield better performance when the input space metric aligns more meaningfully with the considered task. While the method described (H. Guo et al., 2019) learns to detect which interpolations should be disregarded through training, our study advocates for a purely geometric approach where the loss is weighted based on the distance between interpolated samples.

Our novel mixup method is characterized by a single parameter that ranges from the traditional *Mixup* approach to Vanilla (baseline without any mixup). In the context of one-dimensional data, our analysis demonstrates that *Local Mixup* provides the ability to strike a balance between bias and variance. Extending our investigation to higher dimensions, we present evidence that *Local Mixup* outperforms classical *Mixup* when applied to standard vision datasets, resulting in more accurate models. We contribute to a broader understanding of how the inclusion of *Mixup* affects the training process.

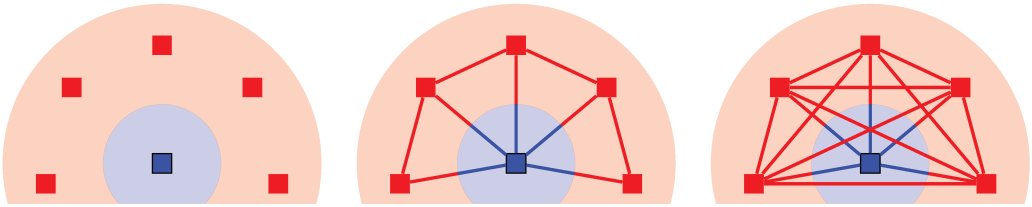


Figure 14 – Illustration of the proposed *Local Mixup* method. On the left, only vanilla samples are used, without data augmentation. The ground truth regions are depicted in filled regions. On the middle we depict *Local Mixup* where we only interpolate samples which are close enough, leading to no contradiction with the ground thuth. On the right we depict *Mixup* in which we interpolate all samples, leading to contradictory virtual samples. For better readability, we color the segments with the argmax of the interpolation of the outputs instead of a gradient of color.

3.2.2 Proposed approach: Local Mixup

Our proposed approach pondered the interpolations according to a weight matrix $\mathbf{W}$ where larger weights correspond to closer pairs of samples. In this study, we explore various approaches to obtain $\mathbf{W}$, but the underlying principle remains consistent. The loss is then weighted using $\mathbf{W}$, resulting in the expression:

$$L_{\text{local mixup}} = \sum_{\mathcal{D}^2_{train}} \mathbf{W}[i,j]\mathcal{L}\left(\tilde{\mathbf{y}}_{i,j,\lambda}, f(\tilde{\mathbf{x}}_{i,j,\lambda})\right). \tag{7}$$

The matrix $\mathbf{W}$ can be interpreted as the weight of a graph $G_{\mathcal{D}} = \langle V, \mathbf{W}\rangle$, where the nodes $V = \{\mathbf{x} \mid \exists \mathbf{y}, (\mathbf{x}, \mathbf{y}) \in \mathcal{D}\}$. The real symmetric matrix $\mathbf{W}$ is based on the pairwise distance matrix D, denoted as $D[i,j] = d_{\mathcal{X}}(\mathbf{x}_i, \mathbf{x}_j)$. To compute the matrix we consider, K-nearest neighbor graphs, where the weights of target vertices are set to 1 for the K closest samples associated with a given source vertex, and 0 otherwise. We also explore thresholded graphs, where $\mathbf{W}[i,j] = \phi(D[i,j])$ and $\phi(d) = \mathbf{1}_{d\leq\varepsilon}$, as well as smooth decreasing exponential graphs, where $\mathbf{W}[i,j] = \exp(-\alpha\mathbf{D}[i,j])$.

To manage computational costs, we compute a graph for each batch (random subset of samples) during stochastic gradient descent. As a result, the weights associating two

samples can vary depending on the selected graph and the randomly chosen batch. In cases where certain weights are 0, the corresponding virtual samples are discarded during gradient descent. Consequently, the method focuses solely on local interpolations of samples, giving rise to the name *Local Mixup*.

In Figure 14 we depict our proposed approach with a thresholded graph. By restricting interpolation we can observe than contrary to Mixup, the inter-class interpolation between red segments does not contradict the ground truth of the blue class.

3.3 Optimal Mixup Criterion Function in dimension 1

3.3.1 Mixup Criterion

Let us introduce proper notations before defining the Mixup Criterion. We denote $\mathcal{D}_{train}$ a training dataset used for learning the model's parameters, and $\mathcal{D}_{test}$ a test dataset used to evaluate the model's performance on unseen inputs (LeCun, 2015a). We assume that both input and output data lie in normed vector spaces with $(\mathcal{X}, \|\cdot\|_X)$ and $(\mathcal{Y}, \|\cdot\|_Y)$, with $\mathcal{X}$ and $\mathcal{Y}$ typically being Euclidean spaces with the usual norms. We refer to $f : \mathcal{X} \to \mathcal{Y}$ as a parametric model to be trained, and $\mathcal{F}$ as an hypothesis set, which contains all candidate parametrizations of the model.

We denote $\mathcal{L}$ the *loss function* that quantifies the discrepancy between the model outputs and the expected outputs. We consider typically the cross-entropy as loss function which we will denote as $L_{vanilla}$:

$$L_{vanilla} = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \mathcal{L}(f(\mathbf{x}), \mathbf{y}).$$

Definition 13: Mixup Criterion

Given $\lambda \sim \text{Beta}[\alpha, \beta]$, where α and β are parameters of the Beta distribution, n represents the size of the dataset, and i and j are discrete variables that are uniformly drawn with replacements from the set $0, \ldots, n-1$. The function f^* that minimizes the Mixup criterion is:

$$f^* = \underset{f\in\mathcal{F}}{\text{argmin}} \frac{1}{n^2}\mathbb{E}_\lambda \left[\underbrace{\sum_{\mathcal{D}^2_{train}} \mathcal{L}\left(\tilde{\mathbf{y}}_{i,j,\lambda}, f(\tilde{\mathbf{x}}_{i,j,\lambda})\right)}_{L_{mixup}} \right].$$

3.3.2 Optimal Mixup Criterion Function

Consider the case where the model, denoted as f, is defined on $\mathbb{R}$. Without loss of generality, let us assume that the training set $\mathcal{D}_{train} = \{x_i, y_i\}$ is ordered in ascending order of inputs, i.e., $x_i \leq x_{i+1}$.

For a given input $\tilde{x}$, the loss function of the Mixup technique implies that the output $f^*(\tilde{x})$ of the model is determined by the set $\mathcal{E}(\tilde{x})$, which consists of all convex combinations that yield $\tilde{x}$ from any pair of training inputs x_i and x_j. Mathematically, we can

express $\mathcal{E}(\tilde{x})$ as follows: $\mathcal{E}(\tilde{x}) = \{i, j, \lambda_{i,j} | \tilde{x} = \lambda_{ij} x_i + (1 - \lambda_{ij})\, x_j\}$. In one-dimensional space, the set $\mathcal{E}(\tilde{x})$ is non-empty and finite for any $\tilde{x} \in [x_0, x_{n-1}]$.

In practice, the distribution of λ is often considered uniform, as in previous works such as Mixup (H. Zhang et al., 2017) and Manifold Mixup (Verma et al., 2019): $\lambda \sim Beta(\alpha = 1, \beta = 1) = \mathcal{U}(0, 1)$. Under this assumption, we demonstrate that the output $f^*(\tilde{x})$ for an input $x \in [x_0, x_n]$ corresponds to the barycenter (weighted mean) of the target values associated with the elements in $\mathcal{E}(\tilde{x})$.

Lemma 1: Expression of f^*

Let us assume that the error function $\mathcal{L}$ is either the cross entropy or the L2 loss. In this case, the function f^* can be described for any x in the interval $[x_0, x_{n-1}]$ by the following equation:

$$f^*(\tilde{x}) = \frac{1}{card(\mathcal{E}(\tilde{x}))} \sum_{(i,j,\lambda_{i,j}) \in \mathcal{E}(\tilde{x})} \lambda_{i,j} y_i + (1 - \lambda_{i,j}) y_j. \tag{8}$$

Proof: Lemma 1

Let us consider an input $\tilde{x} \in [x_0, x_{n-1}]$ and $0 \leq \lambda \leq 1$. For a given triplet $(i, j, \lambda) \in \mathcal{E}(\tilde{x})$, we have $\mathbb{E}[\mathcal{L}(y_i, j, \lambda_{i,j}, f^*(\tilde{x}))|\tilde{x}, i, j, \lambda_{ij}] = \mathcal{L}(y_i, j, \lambda_{i,j}, f^*(\tilde{x}))$ as the values of $y_i, j, \lambda_{i,j}$ and $\tilde{x}$ are known. We aim to minimize the error for all $y_{i,j,\lambda_{i,j}}$ in $\mathcal{E}(\tilde{x})$. Thus, the value of $f^*(x)$ is determined solely by the sum of the losses over $\mathcal{E}(\tilde{x})$, assuming equal probabilities for the elements of $\mathcal{E}(\tilde{x})$ due to uniform distributions of i, j, λ.

$$\begin{aligned} \mathbb{E}[\mathcal{L}(f^*(\tilde{x}), y_{i,j,\lambda_{i,j}}) &= \sum_{\mathcal{E}(\tilde{x})} \mathbb{E}[\mathcal{L}(f^*(\tilde{x}), y_{i,j,\lambda_{i,j}})|\tilde{x}, i, j, \lambda_{ij}] \\ &= \sum_{\mathcal{E}(\tilde{x})} \mathcal{L}(f^*(\tilde{x}), y_{i,j,\lambda_{i,j}}) \end{aligned} \tag{9}$$

Assuming that the error function $\mathcal{L}$ is either the cross-entropy (used in classification tasks) or the squared L2 loss (used in regression tasks), we can derive an equation for $f^*(\tilde{x})$ by setting the derivative of Equation (9) with respect to $f^*(\tilde{x})$ equal to zero, we obtain:

$$f^*(\tilde{x}) = \frac{1}{card(\mathcal{E}(\tilde{x}))} \sum_{\mathcal{E}(\tilde{x})} y_{i,j,\lambda_{i,j}}$$

The following theorem is a consequence of the preceding lemma:

Theorem 1

Considering the L2 loss or the cross entropy, the function f^* that minimizes the loss on the training set is piecewise linear on the interval $[x_0, x_{n-1}]$. Specifically, it is linear on each segment $[x_i, x_{i+1}]$ for $0 \leq i < n - 1$. The definition of f^* on each segment is given by Equation (8).

When the input $\tilde{x}$ varies within the interval $[x_i, x_{i+1}]$, the set of possible combinations of training samples that result in $\tilde{x}$ remains unchanged, except for the linear variation of

the corresponding coefficients λ. As Equation (8) is linear with respect to each of these coefficients, the function f^* is linear with respect to $\tilde{x}$. The set of possible combinations changes whenever $\tilde{x}$ transitions to another interval, such as $[x_{i-1}, x_i]$. In such cases, new combinations may emerge, while others may disappear, resulting in a different linear function. Nonetheless, f^* remains continuous everywhere because the appearance or disappearance of combinations is associated with $\lambda = 0$ or $\lambda = 1$ for $\tilde{x} = x_j$, where $j \in \{1, \cdots, n\}$.

In practical machine learning scenarios, it is often undesirable to infer a function f^* that minimizes the *Mixup* criterion. Instead, the aim is to find a function f with a sufficiently small loss, which has a regularization effect. This is because f^* is unlikely to generalize well. However, it should be noted that f^* tends towards an average of convex combinations, resulting in a model with low variance.

3.4 Local Mixup

3.4.1 The Bias/Variance Trade-off

In this section, our objective is to demonstrate that *Local Mixup* provides the ability to adjust the bias and variance trade-off in trained models. To facilitate our analysis, we simplify the problem to one dimension and consider a thresholded graph. Notably, by varying the threshold, we can create a range of settings where a threshold of 0 corresponds to vanilla training, while a threshold of N, where N represents the number of training samples, corresponds to classical *Mixup*. To begin, let us revisit the definitions of bias and variance in the context of a machine learning problem.

Definition 14: Bias and Variance

Consider a training set $\mathcal{D}_{train}$ and a function f mapping from $\mathcal{X}$ to $\mathcal{Y}$. The bias and variance of this model are defined as follows:

- *Bias*: $Bias(f)^2 = \mathbb{E}_{train}[(f(x) - y)^2]$

Bias measures how far off our model's predictions are from the true values on average. In simpler terms, it quantifies the error introduced by approximating a real-world problem, which may be complex, by a much simpler model. A high bias indicates that the model oversimplifies the data and may not capture important patterns, leading to systematic errors in predictions. This is often referred to as underfitting.

- *Variance*: $Var(f) = \mathbb{E}_{train}[(f - \mathbb{E}_{train}[f])^2]$

Variance serves as a quantification of the model's susceptibility to fluctuations within the training data. It effectively measures the extent to which predictions for a specific data point may diverge when trained on distinct datasets. Elevated variance signifies that the model is excessively responsive to the inherent noise in the training data, thereby accommodating incidental fluctuations rather than encapsulating the intrinsic underlying patterns. This propensity is commonly referred to as overfitting.

The pivotal notion of a trade-off between bias and variance emanates from the inherent challenge in machine learning, which is finding a model capable of generalizing to unseen data. During the model training process, an intricate balance must be struck. On one hand, overly simplistic models (characterized by high

bias) will inadequately represent the intricacies inherent in the data, resulting in underfitting. Conversely, an alternative extreme arises when constructing overly intricate models (typified by high variance). These models may exhibit an exceptional fit to the training dataset, encompassing even the intrinsic noise or random fluctuations present within the data. Nevertheless, this seemingly favorable attribute precipitates a lackluster capacity to generalize to fresh data instances.

In the following, we consider two scenarios. In the first scenario, the input domain $\mathbb{Z}/n\mathbb{Z}$ is periodic, thereby resulting in a finite number of samples. In the second scenario, the input domain $\mathbb{Z}$ is infinite, and the outputs are independent and identically distributed (i.i.d) using a random variable. In both scenarios, the dataset size N can be arbitrarily large, allowing us to study the asymptotic cases.

3.4.2 Periodic setting

Let us consider a training set $\mathcal{D}_{train}$ consisting of pairs (x, y), where $\{x \mid \exists y, (x, y) \in \mathcal{D}_{train}\} = \mathbb{Z}/n\mathbb{Z}$. Additionally, we assume that the distance metric $d_{\mathcal{X}}(x, x') = |x - x'|$ takes values in the range $\{0, \cdots, n-1\}$. This assumption simplifies the proofs by assuming regularly spaced samples, although we believe that similar results could be obtained even if the inputs are not defined over $\mathbb{Z}/n\mathbb{Z}$.

Since the input domain is discrete, we can indicate the threshold parameter K of the graph as an integer. Notably, each time K is increased by 1, every sample becomes connected to two additional neighbors. Under these circumstances, we can derive an explicit formulation of f_K^*, which represents the function that minimizes the *Local Mixup* criterion for K-thresholded graphs. By employing arguments similar to those used to derive Equation (8), we can determine that, for a given x_i,the optimal value of $f^*_{K(x_i)}$ is an average of the $\tilde{y}$ values corresponding to the possible interpolations. This can be expressed as follows:

$$\forall x_i \in \mathbb{Z}/n\mathbb{Z}, f_K^*(x_i) = \frac{1}{K(K+3)/2}(2Ky_i + S_K(x_i)), \tag{10}$$

where $S_K(x_i)$ is defined recursively as follows:

$$S_{K+1} = \begin{cases} 0 & \text{if } K = 0 \\ S_K(x_i) + A_{K+1}(x_i) & \forall K \geq 1 \end{cases}. \tag{11}$$

and:

$$A_K(x_i) = \frac{1}{K}\sum_{k=1}^{K-1}(K-k)\cdot y_{i-k} + k \cdot y_{i+K-k}.$$

Proof: Equation (10)

These expressions can be proven through induction on the parameter K. Let $x_i \in \mathbb{Z}$ and $K > 1$. The term $\frac{1}{K(K+3)/2}$ represents the cardinality of $\mathcal{E}(x_i)$, which denotes the number of interpolations satisfying $x_i = \lambda x + (1-\lambda)x'$. To be more precise, we distinguish between *direct interpolations* and *indirect interpolations*.

Direct interpolations correspond to interpolations between x_i and its immediate neighbors x_j. For example, when $K = 2$, the direct neighbors are x_{i+1}, x_{i+2}, x_{i-1},

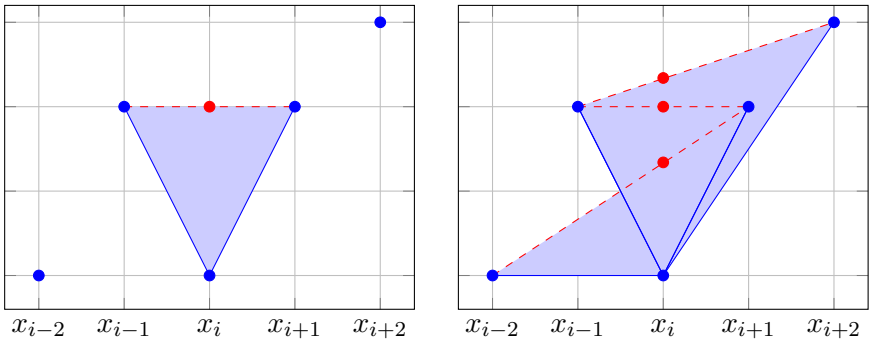


Figure 15 – We depict here the terms of $f_K^*(x_i)$ given by Eq (10) for different K. In blue the interpolations corresponding to $2Ky_i$ and in red the terms of the sum S_K. On the right, $K = 2$ and on the left $K = 1$.

and x_{i-2}. When evaluating $S_K(x_i)$, the direct interpolations are performed using $\lambda = 1 : x_i = 1.x_i + 0.x_j$. Thus, the corresponding interpolated output y value is determined as y_i since $y = \lambda y_i + (1 - \lambda)y_j = y_i$.

Indirect interpolations refer to interpolations between points other than x_i that intersect with x_i, for instance the interpolation between x_{i-1} and x_{x+1}. Figure 15 illustrates these interpolations. As K increases, the influence of S_K (depicted as red points in the figure) also increases, as we will prove below.

We can prove by induction over K that there are at $\frac{1}{K(K+3)/2}$ interpolations, specifically $2K$ direct interpolations and $K(K-1)/2$ indirect interpolations.

- For $K = 1$, the data point x_i is connected to x_{i-1} and x_{i+1}, resulting in 2 neighbors. In this case, no indirect interpolation occurs because $|x_{i-1} - x_{i+1}| = 2 > K = 1$.
- For $K = 2$, the point x_i is connected to x_{i-1}, x_{i+1}, x_{i-2}, and x_{i+2}, resulting in 4 direct neighbors. Finally, there is 1 indirect interpolation: $1/2(y_{i-1} + y_{i+1})$ (see Figure 15).
- Assuming the expression holds true for some K, the function f_{K+1}^* includes the $2K + K(K-1)/2$ interpolations from f^*K. Additionally, it includes the new direct interpolations : (x_i, x_{i+K+1}), (x_i, x_{i-K-1}), and the indirect interpolations: $(x_{i-1}, x_{i+K}), (x_{i-1}, x_{i+K}), \cdots, (x_{i-K}, x_{i+1})$. In total, there are $K(K+3)/2 + 2 + K = (K+1)(K+1+3)/2$ interpolations.

Upon closer examination, one can observe that in Equation 10 direct interpolations are associated with the term $2Ky_i$ $(\lambda = 1)$, whereas indirect interpolations correspond to the term S_K.

Directly, we obtain the following lemma, which demonstrates the invariance of the expected value of f_K^* with respect to K:

Lemma 2: Expected value of f_k^*

For any K, the expected value of f_K^* is

$$\mathbb{E}_{train}[f_K^*] = \mathbb{E}_{train}[y]. \tag{12}$$

Proof: Lemma 2

By definition we can write:

$$\mathbb{E}_{train}[f_K^*] = \frac{1}{n}\sum_{i=1}^{n} f_K^*(x_i)$$

$$= \frac{2}{nK(K+3)}(2nK\mathbb{E}_{train}[y] + \sum_{i=1}^{n}\sum_{k=1}^{K} A_k(x_i)).$$

and using the fact that $y_{i+n} = y_i$:

$$\sum_{i=1}^{n}\sum_{k=1}^{K} A_k(x_i) = \sum_{i=1}^{n}\sum_{k=1}^{K}\sum_{l=1}^{k-1} \frac{k-l}{k} y_{i-l} + \frac{l}{k} y_{i+k-l}$$
$$= \sum_{i=1}^{n} y_i \sum_{k=1}^{K}\sum_{l=1}^{k-1} 1 = n\mathbb{E}_{train}[y]\frac{K(K-1)}{2}.$$

then $\mathbb{E}_{train}[f_K^*] = \mathbb{E}_{train}[y]$.

In the periodic setting, we obtain the following theorem:

Theorem 2: Convergence of f_K^* in the periodic setting

As K grows, it holds that:

$$\forall x_i \in \mathbb{Z}/n\mathbb{Z},$$
$$f_K^*(x_i) \to \mathbb{E}_{D_{train}}[y], \tag{13}$$
$$Bias^2(f_K^*) \to \mathbb{E}_{train}[(y_i - \mathbb{E}_{train}[y])^2], \tag{14}$$
$$Var(f_K^*) = \mathbb{E}_{train}[(f_K^*(x_i) - \mathbb{E}_{train}[f_K^*(x_i)])^2] \to 0, \tag{15}$$
$$Var(f_K^*) \text{ is eventually nonincreasing.}$$

This theorem presents two main results: In the case of *Mixup*, the function f^* that minimizes the loss exhibits zero variance and converges to the expected value $\mathbb{E}_{train}[y]$.

Eventually, the variance of the function that minimizes the Local Mixup criterion decreases, demonstrating that the proposed Local Mixup approach effectively adjusts the trade-off between bias and variance.

In order to establish the theorem, we will explicitly express the limit of f_K^*, ie., S_K . To begin, we first demonstrate the following lemma:

Lemma 3

Let $K = Mn + r$, $M \in \mathbb{N}^*$ and $0 < r < n - 1$. We assume $\mathbb{E}_{train}(y) \geq 0$, then:

$$(M+1)n \cdot \mathbb{E}_{train}(y) + \eta \geq A_K \geq Mn \cdot \mathbb{E}_{train}(y) - \eta, \tag{16}$$

with $\eta = \mathcal{O}(K\mathbb{E}_{train}(y))$.

Proof: Lemma 3

Let be $K = Mn + r,\ M > 1,\ n - 1 > r > 0$. We have:

$$A_K = \frac{1}{K} \sum_{k=1}^{K-1} (K-k) \cdot y_{i-k} + k \cdot y_{i+K-k}$$

The sum above can be decomposed into the sum:

$$\begin{aligned} A_K = \frac{1}{K} \sum_{m=0}^{M-1} \sum_{k=1}^{n-1} & (K-k-mn) y_{i-k-mn} \\ & + (k+mn) y_{i+K-(mn+k)} \\ & + \sum_{k=1}^{r-1} (K-k-Mn) y_{i-k} + (K-k) y_{i+K-k} \end{aligned}$$

For $m \geq 0$ and $1 \leq k \leq n-1$, we can establish that $y_{i-mn-k} = y_{i-k}$ and $y_{i+K-(mn+k)} = y_{i+(M-m)n+r-k} = x_{i+r-k}$. This is due to the periodic nature of signal y with a period of n, and the fact that $K = Mn + r$. Then:

$$\begin{aligned} A_K &= \frac{1}{K} \sum_{m=0}^{M-1} \sum_{k=1}^{n} (K-k-mn) y_{i-k} + (k+mn) y_{i+r-k} \\ &\quad + \sum_{k=1}^{r-1} (K-k-Mn) y_{i-k} + (k+Mn) y_{i+r-k} \\ &= \frac{1}{K} \sum_{m=0}^{M-1} \sum_{k=1}^{n} (K-k-mn) y_{i-k} + (k+r+mn) y_{i-k} \\ &\quad + \sum_{k=1}^{r-1} (K-k-Mn) y_{i-k} + (k+Mn) y_{i+r-k} \\ &= \sum_{k=1}^{n} y_{i-k} M(1 + r/K) + \eta \end{aligned}$$

with $\eta = \| \sum_{k=1}^{r-1} (K-k-Mn) y_{i-k} + (k+Mn) y_{i+r-k} = \mathcal{O}(K\mathbb{E}[y]) \|$, the value of η compared to $K\mathbb{E}[y] \geq 0$ will be negligible. For the time being, let us assume that $\mathbb{E}[y] \geq 0$. We can then proceed with the following expressions:

$$Mn\mathbb{E}[y] - \eta \leq A_K \leq (M+1)n\mathbb{E}[y] + \eta. \tag{17}$$

Similarly, if $\mathbb{E}[y] \leq 0$, we can derive the following inequalities:

$$(M+1)n\mathbb{E}[y] - \eta \leq A_K \leq Mn\mathbb{E}[y] + \eta. \tag{18}$$

Thus, in both cases, $K = Mn + r \sim Mn$ and $A_K \sim KE[y]$.

By combining the previous lemma and Equation (11), we can demonstrate the convergence of the sum S_K and determine its limit using the following corollary:

Corrolary 1

For $K = nM \to \infty$

$$S_K \to \frac{1}{2}\sum_{i=1}^{n} y_i M^2 n = \frac{1}{2}\mathbb{E}_{train}(y)K^2. \tag{19}$$

Proof: Corrolary 1

First, we demonstrate the convergence of S_K/K^2. This can be shown directly using the equations provided: we can express $\|A_K\| \sim K\|\mathbb{E}[y]\|$ and $\sum_{k=1}^{K} K\mathbb{E}[y]/K^2 = \mathbb{E}[y]$. Thus, S_K/K^2 converges absolutely.

Next, we aim to find an equivalent expression for S_K/K^2. To accomplish this, we utilize either Equation (17) or (18), depending on the sign of $\mathbb{E}[y]$. Let us consider Equation (17) without loss of generality:

Given that $S_K = \sum_{k=1}^{K} A_k$, when $K = Mn$, we have the following expression:

$$n\sum_{m=0}^{M-1}(m+1)n\mathbb{E}[y] \geq S_K \geq \sum_{m=0}^{M-1} mn\mathbb{E}[y]$$
$$n^2 M(M+1)/2\,\mathbb{E}[y] \geq S_K \geq n^2(M-1)(M+1)/2\,\mathbb{E}[y]$$

Thus, we have:

$$S_k \to \frac{1}{2}\mathbb{E}(y)K^2 \tag{20}$$

This result holds true even when $\mathbb{E}[y] \leq 0$.

In order to demonstrate the monotonicity of the variance, our goal is to establish the inequality: $Var(f^*_{K+1}) \leq Var(f^*_K)$ for for a K large enough. To accomplish this, we employ the König-Huygens theorem and make use of Lemma 2 to calculate the discrepancy between the two variances:

$$\begin{aligned} &Var(f_{K+1}) - Var(f_K) \\ &= \mathbb{E}_{D_{train}}[(f_{K+1}(x))^2] - \mathbb{E}_{D_{train}}[(f_K(x))^2] \\ &= \mathbb{E}_{D_{train}}[(f_{K+1}(x))^2 - (f_K(x))^2]. \end{aligned}$$

Next, we establish that for any $x \in [x_0, x_{n-1}]$ and a sufficiently large value of K, it holds that $(f_{K+1}(x))^2 \leq (f_K(x))^2$. To do so, we obtain an asymptotic equivalence:

$$(f_{K+1}(x))^2 - (f_K(x))^2 \sim -\frac{K}{C} \cdot E^2_{train}[y],$$

where C is a positive constant.

3.4.3 Independent and Identically Distributed Random Output Setting

Now, let us consider a training set consisting of inputs $\{x \mid \exists y, (x,y) \in \mathcal{D}_{train}\} = \mathbb{Z}$ and outputs y_i are independent and identically distributed (i.i.d.) according to a distribution of variance σ^2.

Theorem 3

For a signal with i.i.d outputs, the variance is eventually bounded by:

$$\frac{4^2\sigma^2}{K^2} \leq Var(f_K(x_i)) \leq \frac{8\sigma^2}{K}. \tag{21}$$

Proof: Theorem 3

Let us select x_i and $K > 1$. It can be observed that $f_K^*(x_i)$ is a sum of random variables. By redefining S_K with coefficients $a_k^K = \sum_{l=k+1}^{K} \frac{l-k}{l}$ we express S_K as follow$S_K = \sum_{k=1}^{K-1}(y_{i-k} + y_{i+k})a_k^K$. Consequently, we obtain the variance of $f_K^*(x_i)$ as:

$$Var(f_K^*(x_i)) = Var\left(\frac{2 \cdot (2Ky_i + S_K)}{K(K+3)}.\right)$$

which leads to:

$$Var(f_K^*(x_i)) = 4^2\left(\frac{K}{K(K+3)}\right)^2 Var(y_i)$$
$$+ \sum_{k=1}^{K-1}\left(\frac{2a_k^{(K)}}{K(K+3)}\right)^2 (Var(y_{i-k}) + Var(y_{i+k})).$$

We use the fact that $\frac{1}{K} \leq a_k^K \leq K$. As K tends to infinity, the following inequalities hold:

$$\frac{4^2\sigma^2}{K^2} \leq Var(f_K(x_i)) \leq \frac{8\sigma^2}{K}.$$

This proof can be easily extended to signals on $\mathbb{R}$ as long as the dataset is finite and sufficiently large.

Invariance of linear models

Remarkably, it can be shown that both *Mixup* and *Local Mixup* yield the same optimal linear models, as stated in the following theorem:

Theorem 4

For a linear model: $f(x) = ax + b,\ a, b \in \mathbb{R}$, the functions that minimizes the loss of *Mixup* and *Local Mixup* are equal (that function is denoted as f^*).

Proof: Theorem 4

We previously demonstrated in Equation (8) that the function f^* resulting from *Mixup* is a piecewise linear function. The same equation applies to *Local Mixup*, except that the set E_x is smaller in *Local Mixup* due to the restricted number of endpoints. As a piecewise linear function, which is linear on each segment $[x_i, x_{i+1}]$, f^* can be represented as $f^* = a_i x + b_i$, where each (a_i, b_i) is defined over $[x_i, x_{i+1}]$. Let $\mathcal{F}$ denote the class of restricted linear functions, then the coefficients a and b are obtained as the averages of the corresponding (a_i, b_i) coefficients.

3.4.4 High Dimension and Lipschitz constraint

The proofs provided in low-dimensional settings have certain limitations. Essentially, the phenomenon of averaging occurs because any point x within the interval $[x_1, x_n]$ can be expressed as a convex combination of pairs from the training set. Contradictions may arise, as demonstrated earlier, when multiple combinations correspond to x. In higher dimensions, explicit contradictions of this nature are not necessarily expected, as the probability of a training sample being an interpolation of two other training samples tends to zero (Balestriero et al., 2021). However, we demonstrate that *Local Mixup* has an impact on the Lipschitz constant of the networks.

First, let us recall the definition of a q-Lipschitz function:

Definition 15: Lipschitz Continuity and Lipschitz Constant

Given two metric spaces $(\mathcal{X}, d_X), (\mathcal{Y}, d_\mathcal{Y})$ and a function $f : \mathcal{X} \to \mathcal{Y}$, f is Lipschitz continuous if there exists a real constant $q \geq 0$ s.t for all x_i and x_j in $\mathcal{X}$,

$$d_\mathcal{Y}\left(f(x_i), f(x_j)\right) \leq q d_\mathcal{X}(x_i, x_j). \tag{22}$$

If f is q-Lipschitz continuous, we define the optimal Lipschitz constant Q_{sup} as

$$Q_{sup} = \sup_{x_i, x_j \in X, x_i \neq x_j} \frac{d_\mathcal{Y}\left(f(x_i), f(x_j)\right)}{d_\mathcal{X}(x_i, x_j)}. \tag{23}$$

For simplicity, let us consider a classification problem where $d_\mathcal{Y}$ is 0 if the two considered samples are of the same class, and 1 otherwise.

Then, the training set imposes a lower bound on the optimal Lipschitz constant:

$$Q_{sup} \geq \underbrace{\left(\min_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}, y_i \neq y_j} d_\mathcal{X}(x_i, x_j)\right)^{-1}}_{Q(D)}. \tag{24}$$

For *Mixup* and *Local Mixup*, the virtual samples increase the size of the training set, resulting in stronger constraints on the optimal Lipschitz constant.

In more detail, let us consider the case of a thresholded graph with parameter ε when using *Local Mixup*. In this case, the augmented training set for each class $\mathbf{y}$ can be written as $S_\varepsilon(\mathbf{y}) = \{\lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j \mid 0 \leq \lambda \leq 1\ \mathbf{y}_i = \mathbf{y}_j = \mathbf{y}, d_\mathcal{X}(\mathbf{x}_i, \mathbf{x}_j) \leq \varepsilon\}$, which represents the set of all segments constructed from two samples that are sufficiently close in the input domain and share the same label $\mathbf{y}$. We can then state the following theorem:

Theorem 5

The lower bound $Q(D)$ is increasing with ε.

Proof: Theorem 5

We directly use the inclusion $S_\varepsilon(\mathbf{y}) \subset S_{\varepsilon'}(\mathbf{y}), \forall \varepsilon \leq \varepsilon'$.

In the experiments, we will demonstrate that ε can indeed impact $Q(D)$ on standard vision datasets.

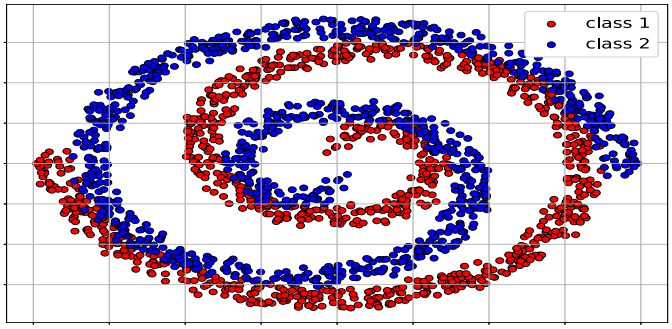


Figure 16 – Illustration of the two coiling spiral dataset with 1000 samples per class and $\sigma = 1.5$.

3.5 Experiments

METHOD	CIFAR-10	CIFAR-100	Fashion-MNIST	SVHN
Baseline	4.98 ± 0.03	30.6 ± 0.27	6.20 ± 0.2	10.01 ± 0.15
Mixup	4.13 ± 0.03	29.23 ± 0.4	6.36 ± 0.16	8.31 ± 0.14
Local Mixup	4.03 ± 0.03	29.08 ± 0.34	5.97 ± 0.2	8.20 ± 0.13

Table 2 – Error rates (%) on CIFAR-10, CIFAR-100 (Resnet18) Fashion-MNIST (DenseNet) and SVHN (LeNet). The values are averaged over 100 runs for CIFAR-10 and 10 runs for CIFAR-100, Fashion-MNIST and SVHN. Mean errors with their confidence interval are represented.

3.5.1 Low dimension

As mentioned in the introduction and in the study (H. Guo et al., 2019), the use of *Mixup* can result in interpolated samples that mislead the model. To demonstrate this effect, we employ a 2D toy dataset consisting of two coiling spirals where such interpolations occur frequently. The dataset represents a binary classification problem, with each spiral corresponding to a distinct class. We expect *Local Mixup* to outperform *Mixup* in this scenario, as local interpolations are more likely to remain within the same spiral, thus avoiding intrusion into the other manifold. In this experiment, **we employ a thresholded graph** with parameter ε.

To conduct this experiment, we generate 1000 samples for each class and introduce Gaussian noise with a standard deviation of $\sigma = 1.5$ to control the thickness of the spirals. The dataset is depicted in Figure 16. We randomly split the dataset into a training set containing 80% of the samples and a test set containing the remaining 20% (used for computing the error rates).

We then use a fully connected neural network consisting of layers composed of 100 neurons, and ReLU function as the non-linearity. The test errors are averaged over 1000 runs. For small values of ε, many weights of the graph become zero, resulting in the corresponding interpolations being disregarded in the loss calculation. This means that for a given batch, only a small proportion of samples are considered when computing the loss. Without any correction, different values of ε lead to different batch sizes. To avoid potential side effects, we vary the batch size such that, on average, the same number of samples is used to update the loss.

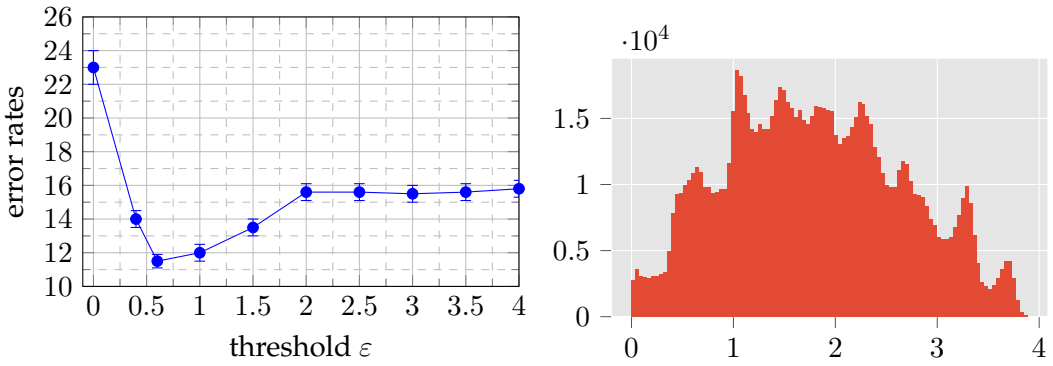


Figure 17 – On the right: Error rate as a function of ε for the two coiling spirals dataset. Values are averaged over 1000 runs. Extremes correspond respectively to Vanilla ($\varepsilon = 0$) and *Mixup* ($\varepsilon > 4$). On the left: Histogram of Euclidean distances $d_{\mathcal{X}}$ between pairs of inputs on the two coiling spirals dataset.

To select an appropriate value of ε, we examined the distribution of distances between pairs of inputs in the training set. The distribution is illustrated in Figure 17. We note that the distribution is relatively uniform within the range of 0 to 4. Therefore, in our experiments, we varied ε between 0 and 4 using increments of 0.5.

In Figure 17, we present the average error rate as a function of the parameter ε. It is important to recall that the extremes, $\varepsilon = 0$ and $\varepsilon = 4$, correspond to Vanilla and *Mixup* respectively. Notably, both *Mixup* and *Local Mixup* exhibit significant improvements over Vanilla. As expected, *Local Mixup* achieves a minimum error rate that is substantially lower than that of *Mixup*. It is noteworthy that the minimum error rate is attained with a value of ε smaller than the first quartile. This suggests that, for this particular dataset, interpolations above this threshold are either ineffective or misleading for the network's training.

It is important to emphasize that this toy dataset is specifically designed to generate contradictory virtual samples. In the subsequent subsection, we will delve into more complex and real-world datasets.

3.5.2 High dimension

Lipschitz lower bound: To demonstrate the influence of ε on the optimal Lipschitz constant, we employ the CIFAR-10 dataset (Krizhevsky, Hinton, et al., 2009). Our objective is to examine the variation of $Q(D)$ as ε is modified. The results are illustrated in Figure 18.

For classical *Mixup*, we obtained $Q(D) = 0.11$, and for Vanilla, $Q(D) = 0.073$. It is worth noting that these two extremes are achieved with *Local Mixup* when $\varepsilon = 0$ and $\varepsilon \geq 50$. We observe that ε can be utilized to smoothly adjust the lower bound $Q(D)$. In practice, a lower $Q(D)$ is preferred, as it corresponds to a smaller optimal Lipschitz constant. However, this only accounts for the Lipschitz constraint. Larger values of ε result in larger training sets, which can potentially lead to improved generalization.

3.5.3 Experiments on Classification Datasets

We proceed to test our approach on various classification datasets and architectures. In the following experiments we used **a smooth decreasing exponential graphs.**

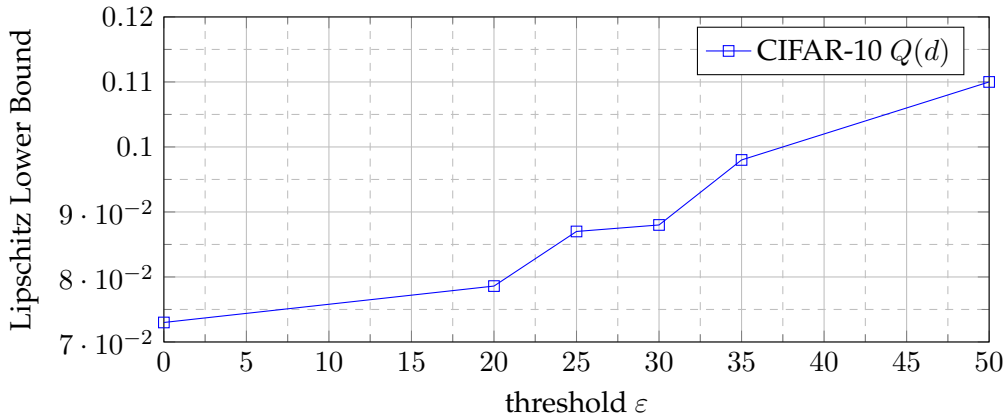


Figure 18 – Evolution of $Q(D)$ on the dataset cifar10. Note that $\varepsilon = 0$ corresponds to Vanilla. $\varepsilon = 50$ corresponds to classical *Mixup*.

Classification
The datasets considered are CIFAR10 (Krizhevsky, Hinton, et al., 2009), SVHN (Netzer et al., 2011), and Fashion-MNIST (Xiao et al., 2017). Fashion-MNIST consists of grayscale images of clothing items with dimensions of 28×28 pixels. The training set contains 60,000 images classified into 10 classes. SVHN is a real-world image dataset comprising small cropped digits with dimensions of 32×32 pixels and 3 color channels. The training set consists of 73,257 digits categorized into 10 classes.

For the CIFAR10 dataset, we use a ResNet18 (He et al., 2016) following the approach in (H. Zhang et al., 2017). In table 2, we average the error rates over 100 runs and report the mean with a 95% confidence interval. We observed that *Local Mixup* with a value of $\alpha = 0.003$ achieved a lower error rate than both the Vanilla network and *Mixup*, with non-overlapping confidence intervals.

For the Fashion-MNIST dataset, we employ a Densenet (Huang et al., 2017) architecture and average the error rates over 10 runs. We report the mean and 95% confidence intervals as well. Once again, *Local Mixup* with $\alpha = 1e - 3$ yielded a lower error rate compared to both the baseline and *Mixup*. Notably, for this dataset *Mixup* negatively impacted the error rate, suggesting that *Mixup* generated spurious interpolations, as discussed in (H. Guo et al., 2019). For the SVHN dataset, we implemented a LeNet-5 (LeCun et al., 1998) architecture consisting of three convolutional layers. As before, *Local Mixup* outperformed Vanilla and *Mixup* in terms of error rate.

In these experiments, we also attempted to use a K-nearest neighbor graph or a thresholded graph, but we were unable to achieve lower error rates compared to *Mixup* or even Vanilla. This may indicate that some segments generated by *Mixup* are crucial for acting as a regularizer during training, despite the possibility of manifold intrusions. By adjusting the value of α, we can control the importance of this regularization.

Adversarial Attack
Following the methodology outlined in the original Mixup paper(H. Zhang et al., 2017), we conducted a black box attack on the CIFAR-10 dataset. The images were rescaled to the range $[0, 1]$, and Gaussian noise $\mathcal{N}(0, \varepsilon)$ was added with varying standard deviations. The error rates for different noise levels are presented in Table 3.

We observed that for low noise values, *Mixup* and *Local Mixup* produced similar results. However, as the noise standard deviation increased, the performance gap between the two methods widened, with *Local Mixup* outperforming *Mixup*. This outcome aligns

with our theoretical analysis, which demonstrates that our approach relaxes the Lipschitz constraint imposed by *Mixup*, resulting in smaller values of the Lipschitz constant.

Table 3 provides an overview of the error rates obtained for different noise levels, showcasing the superiority of *Local Mixup* in scenarios with higher noise level.

Epsilon ε	**Vanilla**	**Local Mixup**	**Mixup**
0.0025	$7,55$	$6,30$	$\mathbf{5,75}$
0.005	$16,70$	$12,82$	$\mathbf{12,28}$
0.0075	$30,90$	$\mathbf{22,81}$	$23,60$
0.01	$45,60$	$\mathbf{33,59}$	$37,00$

Table 3 – Black box attack: error rates (%) on CIFAR-10, for different values of noise ε.

Comparaison with *Adamixup*

We conducted a comparison between our proposed approach, *Local Mixup*, and another method called *Adamixup* (Verma et al., 2019), which is also designed to prevent manifold intrusion. To perform this comparison, we utilized the GitHub repository of the *Adamixup* author and made modifications to implement our method, while keeping the rest of the framework unchanged.

For the CIFAR-10 dataset, we used 1400 epochs for both *Local Mixup* and *Adamixup*, as the authors of *Adamixup* used this number in their experiments. Upon analysis, we observed a slight advantage for *Adamixup* on the MNIST dataset, with an error rate of $0.49\% \pm 0.03$ for *Adamixup* and $0.54\% \pm 0.02$ for *Local Mixup* (averaged over 10 runs). However, on the CIFAR-10 dataset, our method, *Local Mixup*, outperformed *Adamixup*, with error rates of $4.11\% \pm 0.12$ for *Adamixup* and $3.89\% \pm 0.12$ for *Local Mixup* (averaged over 5 runs).

It is important to note that while *Adamixup* completely discards interpolations that are considered to cause manifold intrusions, *Local Mixup* weighs these interpolations instead of discarding them. The fact that *Local Mixup* achieves comparable or better results than *Adamixup* suggests that even interpolated samples causing manifold intrusions can be beneficial as long as they do not dominate the loss function. This highlights the potential usefulness of such samples in training.

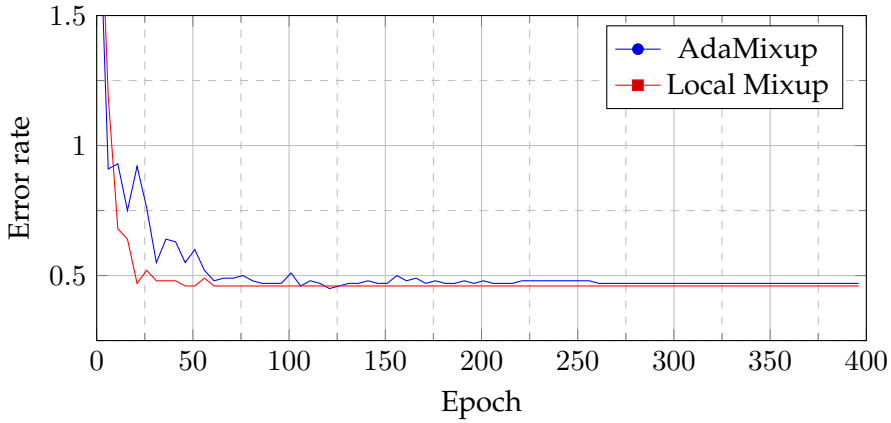


Figure 19 – Evolution of the error rate for *Local Mixup* and *Mixup* on MNIST. We observe a faster convergence for *Local Mixup*.

In terms of convergence speed, our proposed approach, *Local Mixup*, seems to converge faster on the MNIST dataset, as demonstrated in Figure 19. One advantage of our

method is its simplicity and the small number of parameters required. Specifically, for the CIFAR-10 dataset, *Local Mixup* only requires 836,522 parameters compared to 11,171,146 parameters for *Adamixup*.

3.6 Additional studies

3.6.1 Graph Construction in High Dimension

Instead of using smooth decreasing exponential graphs, we explored the use of K-nearest neighbor graphs for high-dimensional datasets. The graph is computed for each batch. In Table 4, we report the test error rates on CIFAR-10 for different values of K. Unfortunately, the results were significantly worse compared to the baseline results reported in Table 2.

K = 1	**K = 5**	**K = 10**
0.607%	0.606%	0.601%

Table 4 – Error rates for different value of K when using a K-nearest neighbor graph on CIFAR-10.

We attempted to compute the graph on the entire dataset, but we did not observe any significant improvements in the error rates. These results suggest that it may be beneficial not to completely discard interpolations outside the manifold, but rather to reduce their influence on the loss function.

3.6.2 Inter and Intra Mixup

It is worth considering whether *Local Mixup* is beneficial solely because it restricts the interpolations to samples of the same class (Intra Mixup). To investigate this, we conducted additional experiments on CIFAR-10 where we allowed interpolations only between samples of the same class (Intra Mixup) or only between samples of different classes (Inter-Mixup). In both cases, the error rates were worse than those obtained with classical *Mixup*. Specifically, the error rate was 4.5% for *Inter-Mixup* and 4.7% for *Intra-Mixup*. Moreover, on the spiral dataset, we calculated the proportion of inter-class interpolations versus intra-class interpolations carried out by *Local Mixup*. We found that 30% of the interpolations were intra-class, while 70% were inter-class. These results indicate that using both intra-class and inter-class mixing is necessary for improved performance.

3.6.3 Hyperparameter α

In this section, we provide ablation studies for different datasets to validate our hyperparameter α. Due to the large range of the confidence intervals, our goal was to select an appropriate order of magnitude rather than specific values. On Fashion-MNIST, the error rate is {6.5%, $\alpha = 1e-1$}, {5.97%, $\alpha = 1e-3$} and {6.02%, $\alpha = 1e-4$} . On cifar10, the error rate is {4.3%, $\alpha = 1e-2$}, {4.05%, $\alpha = 1e-3$} and {4.03%, $\alpha = 3e-3$}. Hence, we use the hyperparameters given in Table 5 to carry out the experiments in Table 2.

CIFAR-10	CIFAR-100	Fashion-MNIST	SVHN
$3e-3$	$3e-3$	$1e-3$	$5e-2$

Table 5 – Value of hyperparameter α for the different datasets used in experiment reported in Table 2.

3.7 Limitations and perspectives

3.7.1 Limitations

Experiments conducted in both low and high dimensions have demonstrated the ability of *Local Mixup* to outperform *Mixup* by leveraging locality. However, the choice of hyperparameters such as α, ε, or K is crucial and depends on the data. In our current work, we reported results by selecting the parameter values that led to the best test error rate among a small number of possibilities.

Additionally, it is important to note that we used the Euclidean metric to embed the notion of locality, even though datasets generally lie on nonlinear manifolds. For example, on CIFAR-10, (Abouelnaga et al., 2016) demonstrated that using the Euclidean metric can achieve classification scores significantly better than chance level but still far from state-of-the-art performance. There are several possibilities for improvement, including the use of pullback metrics(Jost & Jost, 2008; Kalatzis et al., 2020), which involve the Euclidean distance between samples once they are projected into the feature space corresponding to the penultimate layer.

3.7.2 Perspectives

Further research includes delving into intriguing possibilities for enhancing the Mixup technique. To begin, extending the theoretical findings to encompass broader contexts (higher dimension) would yield valuable insights into the influence of locality on the Mixup framework.

Another avenue for exploration involves the application of Local Mixup within the latent space, where the underlying geometry tends to conform more closely to Euclidean principles. This shift in perspective offers the potential to generate new samples using more relevant distance metrics and interpolation functions. Moreover, it opens the door to the fascinating prospect of employing generative neural networks, such as normalizing flows, for interpolation within the latent space, facilitating subsequent natural image reconstruction.

In addition to improving Mixup's geometric aspects, further research can delve into its role as a regularization technique. An in-depth examination of Mixup as a means to introduce anisotropic noise along specific directions, guided by the interpolation coefficient, holds promise. Furthermore, exploring Mixup's capacity to rebalance the gradient contributions of training samples, particularly those that become marginalized as training progresses, is a compelling avenue of investigation. Specifically, Mixup has the potential to reintroduce significance to these samples by interpolating them with more challenging examples.

3.8 Conclusion

We introduced a methodology called *Local Mixup*, which incorporates the notion of locality into the Mixup framework by interpolating and weighting pairs of samples

based on their distance in the input domain. By introducing a hyperparameter, *Local Mixup* provides a continuous range of solutions between Vanilla and classical Mixup.

Through our experiments, we demonstrated that *Local Mixup* can effectively control the bias/variance trade-off of trained models. Furthermore, in more general settings, we showed that *Local Mixup* can tune a lower bound on the Lipschitz constant of the trained model. By comparing it with Vanilla and classical Mixup on real-world datasets, we observed that *Local Mixup* achieves better generalization, as measured by the test error rate.

Both Mixup and Local Mixup techniques can be seen as methods to artificially increase the complexity of the training task with the aim of enhancing generalization on the final, less complex target task. In the upcoming chapter, we will explore an inverse approach: training on simpler tasks and then tackling more challenging, closely related target tasks. Surprisingly, we will discover that with appropriate regularization, neural networks can exhibit the capacity to generalize beyond the training task.

Chapter 4

From coarse to refined labels: generalizing beyond the training task

Contents

4.1 Introduction

4.1.1 From Coarse to Refined Labels

As previously discussed, deep learning models often struggle with overfitting during training. To address this issue, we have mentioned methods like regularization and adjusting the training criteria to enhance the generalization. For instance, techniques like Mixup and Local Mixup involve creating an artificial training task more complex to improve generalization on the primary target task.

Now, our focus turns to another aspect of generalization: the ability of neural architectures to learn to solve broader tasks on their own while having been trained on a specific problem, without explicit guidance.

More specifically, we are examining a supervised classification scenario that involves two types of labels: refined labels, which are not available during training, and coarse labels, which are used for the training process. Typically, this represents a subtle classification or regression task represented by refined labels, which has been simplified into a more straightforward classification task using"coarse labels. In line with prior work (Huh et al., 2016; Touvron et al., 2021), we are interested in investigating whether training on these coarse classes can yield features capable of refined recognition. Such scenarios are likely to occur in practical applications, as coarse labeling can be significantly less expensive, (Xu et al., 2021). Conversely, numerous classification problems, and hierarchical datasets (J. Deng et al., 2009; Horn et al., 2018), can be seen as discretizations of more intricate regression problems, where the labels are not inherently discrete (Grandvalet & Bengio, 2004). In certain studies concerning hierarchical image classification (Eshratifar et al., 2021; Y. Guo et al., 2018; Ristin et al., 2015; Taherkhani et al., 2019), a coarse annotation is provided for all training images, whereas only a subset of these images is meticulously labeled with refined labels.

4.1.2 The Loss of Information Caused by the Cross Entropy Loss

The training process has a significant impact of the inferred representation (Janocha & Czarnecki, 2017b; Müller et al., 2019) and the output distributions $q_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ (C. Guo et al., 2017; Hinton et al., 2015; Meister et al., 2020; Pereyra et al., 2017). The emergence of self-supervised learning has led to the proposal of numerous criteria, resulting in highly generalized representations, which can be directly employed as feature extractors to solve diverse tasks without even requiring fine-tuning (Radford et al., 2015; Raina et al., 2007; Y. Wang et al., 2019)). With abundant data and computational resources, large architectures have emerged with the ability of leveraging knowledge gained from their task task and apply it to another one (Krizhevsky et al., 2017; Zhuang et al., 2020a)). These criteria aim to uncover the representations with the best transferability (Bommasani et al., 2021; Dosovitskiy et al., 2020).

However, we decide to focus on the Cross-Entropy loss since it is the standard criterion used in supervised learning for classification problems (Goodfellow et al., 2016) and we show that it can actually retrieve information on the refined labels. However, this ability is not granted for free. Indeed, when employing the Stochastic Gradient Descent (SGD) optimization algorithm or its variants to minimize the Cross Entropy, non-informative features for the given task tend to be eliminated. This phenomenon was observed by the authors in (Shwartz-Ziv & Tishby, 2017) through two distinct sequential phases.

During the first phase, the ***mutual information*** $I(\mathbf{y}, \mathbf{r})$ between the features $\mathbf{r}$ and the output $\mathbf{y}$ increases. We recall that Mutual information is formally defined as:

Definition 16: Mutual Information

The ***mutual information*** $I(\mathbf{X}, \mathbf{Y})$ between two continuous random variables $\mathbf{X}$ and $\mathbf{Y}$ is defined as follows:

$$I(\mathbf{X}, \mathbf{Y}) = \int_{\mathcal{X}} \int_{\mathcal{Y}} p_{\mathbf{X},\mathbf{Y}}(\mathbf{x}, \mathbf{y}) \log \left(\frac{p_{\mathbf{X},\mathbf{Y}}(\mathbf{x}, \mathbf{y})}{p_{\mathbf{X}}(\mathbf{x}) p_{\mathbf{Y}}(\mathbf{y})} \right) .dx.dy \tag{25}$$

Where:

- $p_{\mathbf{X}}$ is the probability mass function of random variable $\mathbf{X}$.
- $p_{\mathbf{Y}}$ is the probability mass function of random variable $\mathbf{Y}$.
- $p_{\mathbf{X},\mathbf{Y}}$ is the joint probability mass function of $\mathbf{X}$ and $\mathbf{Y}$.

In this context, the mutual information $I(\mathbf{y}, \mathbf{r})$ serves as a measure of the statistical dependence between the feature vector $\mathbf{r}$ and the output vector $\mathbf{y}$. This quantification underscores the growing extent of information shared between them, signifying the increasing relevance of the feature representation for the associated tasks. Subsequently, in a second, longer phase, the mutual information $I(\mathbf{x}, \mathbf{r})$ between the input and the features decreases: the information is compressed.

In summary, the architecture first identifies the features that enable accurate label prediction before compress

In classification settings, the output of the architecture $f_{\boldsymbol{\theta}}(\mathbf{x})$ can be interpreted as the conditional class probabilities $q_{\boldsymbol{\theta}}(\mathbf{y}_i|\mathbf{x})$, once a softmax function with a temperature parameter τ, has been applied (Hinton et al., 2015):

$$q_{\boldsymbol{\theta}}(y_i|\mathbf{x}) = \frac{\exp(f_{\boldsymbol{\theta}}(\mathbf{x})_i/\tau)}{\sum_j \exp(f_{\boldsymbol{\theta}}(\mathbf{x})_j/\tau)}. \tag{26}$$

Hence $q_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ represents the inferred distribution that ideally aligns with the true distribution $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$. This transformation is governed by the temperature parameter τ, which regulates the level of uncertainty in the model's predictions. A higher τ results in a more uniform and uncertain distribution, while a lower τ makes the distribution sharper and more confident.

Similarly to our previous discussion, a study (C. Guo et al., 2017)) demonstrated that as architectures minimize classification errors, there is simultaneous degradation in probabilistic errors and miscalibration between $q_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ and the true distribution $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$. Miscalibration refers to inconsistencies between model-predicted probabilities or confidences and the actual probabilities or frequencies of predicted events.

This miscalibration might serve as an indicator of lost information within the feature space. One common approach to alleviate miscalibration is to introduce a regularization term $\mathcal{R}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta})$ into the loss function (C. Guo et al., 2017), leading to the following optimization problem:

$$\min_{\theta} \mathbb{E}_{\mathcal{D}_{train}}[\mathcal{L}_{CE}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta})) + \mathcal{R}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta})]. \tag{27}$$

4.1.3 Entropy Regularization and *refined labels*

One effective approach to regularize deep neural networks is by incorporating *refined labels*, which prevents them from becoming overly specialized to the training task (Bagherinezhad et al., 2018). There are various methods to achieve label refinement, such as utilizing predefined taxonomies as described in (Wu et al., 2017), extracting semantic information from pre-trained networks (Bagherinezhad et al., 2018; Hinton et al., 2015), or employing data augmentation techniques like noisy or interpolated labels, as well as smoothed labels (Baena et al., 2022c; J. Li et al., 2020; Szegedy et al., 2016; H. Zhang et al., 2017). These pseudo-*refined labels* may be based on an underlying truth, such as a taxonomy, but more often they are artificially generated, employing techniques such as noisy or smoothed labels.

In contrast, our aim is to retrieve the ground truth *refined labels*, either from the feature space or through output regularization using pseudo labels. Notably, several authors have shown that techniques like label smoothing (Szegedy et al., 2016), distillation (Szegedy et al., 2016), and confidence penalization (Pereyra et al., 2017) can be seen as forms of entropy regularization on the output distribution (Dubey et al., 2017; Meister et al., 2020; Pereyra et al., 2017). Entropy regularization of outputs leads to smoother and more accurate output distributions (Dubey et al., 2017; Pereyra et al., 2017), while also influencing the geometry of the feature space (Müller et al., 2019; Shen et al., 2021). More detailed explanations of these techniques are provided in the next paragraphs.

Confidence penalization involves incorporating a regularization term that represents the negative entropy of the output distribution: $\mathcal{R}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}) = -H(q_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}))$ (Pereyra et al., 2017) . The purpose is to penalize confident output distributions, as they typically correspond to low-entropy distributions. This results in a smoothed output distribution and can improve generalization performance under certain conditions.

Label Smoothing (Szegedy et al., 2016) is another technique that produces similar effects to confidence penalization (Pereyra et al., 2017). It helps distinguish classes that are semantically similar, which can be advantageous in classification tasks (Shen et al., 2021). Label Smoothing is a form of output regularization that smooths the target distribution $p(\mathbf{y}|\mathbf{x})$. In classification, this distribution is represented by *coarse labels* encoded as one-hot vectors, where the correct label has a probability of 1, and all other labels have a probability of 0. In uniform Label Smoothing with coefficient σ, a probability boost of $1-\sigma$ is assigned to the correct label, and a penalty of $\sigma/(c-1)$ is applied to the probabilities of other labels (where c is the number of classes).

Distillation involves utilizing the class probabilities from a larger model to train a smaller one. These probabilities are considered as *refined labels*, and when they exhibit high entropy, they are believed to provide more information than *coarse labels* (Hinton et al., 2015). Distillation can be viewed as a form of non-uniform label smoothing (Yuan et al., 2019; Z. Zhang & Sabuncu, 2020), where the output distribution of the larger model serves as a prior for the refined target distribution. However, other methods propose inferring the smoothed labels as well, as seen in (Bagherinezhad et al., 2018; Yu et al., 2021). The authors of (Dubey et al., 2018; Dubey et al., 2017) argue that distillation can lead to more generalizable features and encourage the classifier to reduce the specificity of the features.

While these techniques have shown promising results, they have also sparked some

controversies. The authors of (Müller et al., 2019) emphasize that the loss of information in distillation may negatively impact the quality of the inferred *refined labels*. More recently, the authors of (Kornblith et al., 2021) demonstrated that "better losses" such as Label Smoothing, which promote greater class separation, can be detrimental to transfer performance as they produce less generic features. In our experiments, we also found that when the inferred features are reused for regression tasks, Label Smoothing can have negative effects.

4.1.4 Problem Statement

In this study (Baena et al., 2022a), our objective is to demonstrate the potential of revealing the disregarded *refined labels* while training a model on derived *coarse labels*. A challenge when learning from *coarse labels* is that the model tends to lose its capacity for refined generalization due to overfitting to the training task. This phenomenon is commonly observed in transfer learning involving deep neural networks, especially in few-shot problems. In such cases, stopping the training early can result in better performance on subsequent tasks (Mangla et al., 2020). This behavior can be explained by the two-phase nature of cross-entropy loss training described in (Shwartz-Ziv & Tishby, 2017), where mutual information between the feature space and the inputs diminishes in the second phase as the focus shifts towards excelling at the training task.

Considering these observations, we conducted an experiment to further illustrate the impact of retrieving *refined labels* when models are trained solely on *coarse labels*. Our experiment focused on an age estimation problem (Rothe et al., 2018) aiming to predict individuals' ages from their photos. We transformed this regression task into a classification task with approximated labels, training a neural network accordingly. We noted a significant relationship between the entropy of the feature space and the ability to extract information regarding the regression task, Mean Square Error (MSE), similar to the phases observed in (Shwartz-Ziv & Tishby, 2017). Initially, the MSE of the regression task decreased with the classification error rate. However, in a longer, second phase, the MSE reached a minimum before progressively increasing, while the classification error rate remained stable.

To address the challenge of enhancing generalization to subtle, unseen labels, we introduce a regularization method named Feature Information Entropy Regularized Cross Entropy (FIERCE). FIERCE is an entropy-based regularization defined in the feature space of trained architectures. The rationale behind employing entropy as a regularization is that entropy promotes diversity (Qin & Zhu, 2013), countering the tendency for aggressive selection of the best features observed with vanilla cross-entropy. We illustrate theoretically how this criterion promotes diversity.

Through multiple experiments, we show that with our criterion the ability to transfer to *refined labels*, once acquired, remains uncompromised with further training. Consequently, the trained models demonstrate improved generalization capabilities beyond their training task.

4.2 Impact of the Criteria on the Feature Space

In this section, we will explore the relationship between feature selection and the Cross Entropy and Label Smoothing criteria within a Bayesian framework. This analysis will demonstrate why promoting entropy in the feature space can potentially enhance performance in tasks involving *refined labels*.

4.2.1 Links between Feature Selection and Cross Entropy and Label Smoothing

In classification settings, a deep learning architecture typically transforms inputs into feature vectors, which are then classified using logistic regression. Let $\mathbf{r}$ represents the features with respect to $\mathbf{x}$, and y denotes the class of a sample, such that $y = \text{argmax}i\ \mathbf{y}[i]$. To simplify the equations, we adopt a Bayesian framework, where the classification is determined by $p(y = y_i|\mathbf{r})$, derived from the inferred feature distribution $p_{\boldsymbol{\theta}}(\mathbf{r}|\mathbf{x})$. For the sake of simplicity, we consider the classification layer as fixed in the equation, noting that this does not affect the generality of the proof since the gradient is updated using the chain rule. Within this framework, the model first samples features $\mathbf{r}$ according to $\mathbf{x}$, following the distribution $p_{\boldsymbol{\theta}}(\mathbf{r}|\mathbf{x})$. Subsequently, it assigns class probabilities $\log(p(y = y_i|\mathbf{r}))$ based on the output probability from the classifier. In the following equations, we assume that the distributions are differentiable and that their derivatives are bounded by integrable functions, allowing us to interchange integrals and derivatives.

Cross Entropy: In the Bayesian framework, the Cross Entropy can be rewritten as:

$$\mathcal{L}_{CE}^{Bayesian}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}) = \mathbb{E}_{\mathcal{D}}\left[-\mathbb{E}_{\mathbf{r}\sim p_{\boldsymbol{\theta}}(\mathbf{r}|\mathbf{x})}[\log(p(y = y_i|\mathbf{r}))]\right], \quad \text{where } y_i \text{ is the coarse label of } \mathbf{x}. \tag{28}$$

By computing the gradient of $\mathcal{L}_{CE}^{Bayesian}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta})$ w.r.t to $\boldsymbol{\theta}$, we obtain:

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{CE}^{Bayesian}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}) = \mathbb{E}_{\mathcal{D}}\left[-\int \nabla_{\boldsymbol{\theta}}(p_{\boldsymbol{\theta}}(\mathbf{r}|\mathbf{x}))\log(p(y = y_i|\mathbf{r})d\mathbf{r}\right]. \tag{29}$$

Remarkably, we observe that $\nabla_{\boldsymbol{\theta}}(p_{\boldsymbol{\theta}}(\mathbf{r}|\mathbf{x}))$ is weighted by $\log(p(y = y_i|\mathbf{r}))$ indicating that the more selective the features $\mathbf{r}$ are for the classifier, the more they will be encouraged to be sampled. To visualize this effect on the gradient, let us consider a binary classification problem with a fixed classifier. Suppose the neural network can infer a 1-dimensional feature space where the features are linearly separable by the classifier. The features that are further away from the decision boundary will have a greater impact on $\log(p(y = y_i|\mathbf{r}))$. Therefore, despite already having linearly separable features, the Cross Entropy gradient will eventually force the features to be pushed as far as possible towards the margin, resulting in the collapse of the features towards the two extremes of the feature space. Consequently, Cross Entropy can lead to overconfidence in the network's predictions, as reported in (C. Guo et al., 2017).

Label Smoothing: A similar derivation can be carried out for Label Smoothing, where the *coarse labels* $\mathbf{y}$ are uniformly smoothed by a factor $\sigma < 0.5$: $\mathbf{y} = (1-\sigma)\mathbf{y} + \sigma\mathbf{1}$, or $y[i] = \begin{cases} (1-\sigma) \text{ if } i \text{ is the class of x,} \\ \sigma \text{ otherwise} \end{cases}$, where $\mathbf{1} = [1, 1, \cdots, 1]^T$. The gradient becomes:

$$\mathbb{E}_{\mathbf{x}}\left[-\int \nabla_{\boldsymbol{\theta}}(p_{\boldsymbol{\theta}}(\mathbf{r}|\mathbf{x}))[(1-\sigma)\log(p(y = y_i|\mathbf{r}) + \frac{\sigma}{c-1}\log(\prod_{j,j\neq i} p(y = y_j|\mathbf{r}))]d\mathbf{r}\right]. \tag{30}$$

We observe that $\log(p(y = y_i|\mathbf{r}))$ is reduced by a factor of $(1-\sigma)$. However, Label Smoothing introduces additional terms $\frac{\sigma}{c-1}\log(\prod_{j,j\neq i} p(y = y_j|\mathbf{r}))$, which encourage

the most discriminant features for the other classes j. Thus, while Label Smoothing reinforces discriminant features with respect to relevant classes, it also encourages discriminant features for all classes. It is worth noting that the same derivation can be applied to confidence penalization.

In conclusion, Label Smoothing promotes diversity but only among discriminant features. Features that contribute to high accuracy ($p(y = y_i|\mathbf{r}) > 0.5$) are not explicitly encouraged. This observation aligns with previous studies (Müller et al., 2019; Shen et al., 2021), which reported that Label Smoothing diminishes intra-class information in order to promote inter-class information.

4.2.2 Entropy regularization to encourage diversity in the feature space

Based on experimental evidence highlighting the correlation between the entropy of the feature space and regression performance, we investigate the use of entropy regularization on the features to promote diversity. While Cross Entropy and Label Smoothing tend to prioritize the most discriminant features, our approach aims to maintain a diverse set of features.

We incorporate the entropy of the feature space directly as a regularization term. Within the Bayesian framework, the criterion can be then written as follow:

$$\begin{aligned}\mathcal{L}_{Entropy}^{Bayesian}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x}}\left[-\mathbb{E}_{\mathbf{r}\sim p_{\boldsymbol{\theta}}(\mathbf{r}|\mathbf{x})}[\log(p(y = y_i|\mathbf{r}))]\right] \\ -\lambda H_{\boldsymbol{\theta}}(\mathbf{r}).\end{aligned} \tag{31}$$

where the entropy $H_{\boldsymbol{\theta}}(\mathbf{r})$ is estimated for each batch and $\lambda > 0$ is a scalar hyperparameter. The derivation of the gradient of $H_{\boldsymbol{\theta}}(\mathbf{r})$ is:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}(-H_{\boldsymbol{\theta}}(\mathbf{r})) &= \nabla_{\boldsymbol{\theta}} \int p_{\boldsymbol{\theta}}(\mathbf{r}(\mathbf{s}) \log(p_{\boldsymbol{\theta}}(\mathbf{r}(\mathbf{s})))d\mathbf{s} \\ &= \nabla_{\boldsymbol{\theta}} \int \mathbb{E}_{\mathbf{x}}\left[p_{\boldsymbol{\theta}}(\mathbf{r}(\mathbf{s})|\mathbf{x}) \log(p_{\boldsymbol{\theta}}(\mathbf{r}(\mathbf{s})))d\mathbf{s}\right] \\ &= \mathbb{E}_{\mathbf{x}}\left[\int \nabla_{\boldsymbol{\theta}}(p_{\boldsymbol{\theta}}(\mathbf{r}(\mathbf{s})|\mathbf{x}))(\log p_{\boldsymbol{\theta}}(\mathbf{r}(\mathbf{s})))]d\mathbf{s}\right] \\ &+ \int p_{\boldsymbol{\theta}}(\mathbf{r}(\mathbf{s}))\frac{1}{p_{\boldsymbol{\theta}}(\mathbf{r}(\mathbf{s}))}\nabla_{\boldsymbol{\theta}}(p_{\boldsymbol{\theta}}(\mathbf{r}(\mathbf{s})))d\mathbf{s} \\ &= \mathbb{E}_{\mathbf{x}}\left[\int (\nabla_{\boldsymbol{\theta}}(p_{\boldsymbol{\theta}}(\mathbf{r}(\mathbf{s})|\mathbf{x}))(\log p_{\boldsymbol{\theta}}(\mathbf{r}(\mathbf{s}))) + \nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\mathbf{r}(\mathbf{s})|\mathbf{x}))d\mathbf{s}\right].\end{aligned}$$

Hence, the final gradient of Cross Entropy with the entropy regularization of the feature can be expressed as:

$$\begin{aligned}-\mathbb{E}_{\mathbf{x}}\Bigg[\int \nabla_{\boldsymbol{\theta}}(p_{\boldsymbol{\theta}}(\mathbf{r}(\mathbf{s})|\mathbf{x}))[\log(p(y = y_i|\mathbf{r}(\mathbf{s})) \\ -\lambda(\log(p_{\boldsymbol{\theta}}(\mathbf{r}(\mathbf{s}))) + 1)]d\mathbf{s}\Bigg].\end{aligned} \tag{32}$$

Remarkably in this formulation, if a feature has a high probability, i.e., $p(\mathbf{r})$ is large, the gradient will be penalized. The second term acts as a constant regularization. By adjusting the hyperparameter λ, we can control the extent to which the model retains discriminant features while encouraging also diversity in the feature space.

4.3 Implementation of feature entropy regularization

4.3.1 Estimation of the features entropy

With FIERCE, we propose to incorporate the negative entropy of the features, denoted as $\tilde{H}_{\boldsymbol{\theta}}(\tilde{\mathbf{r}})$, into the Cross Entropy loss. However, directly computing and differentiating the entropy of the feature space $H_{\boldsymbol{\theta}}(\mathbf{r})$ is not feasible since it requires computing the probability distribution $p(\mathbf{r})$. To overcome this challenge, we rely on an approximation of the real entropy, denoted as $\tilde{H}_{\boldsymbol{\theta}}(\tilde{\mathbf{r}})$:

$$\mathcal{L}_{Entropy}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}) = \mathcal{L}_{CE}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}) - \tilde{H}_{\boldsymbol{\theta}}(\tilde{\mathbf{r}}),$$

To facilitate the estimation of entropy, we approximate the conditional distribution of the feature space $p_{\boldsymbol{\theta}}(\mathbf{r}|\mathbf{x})$ as a categorical distribution $p_{\boldsymbol{\theta}}(\tilde{\mathbf{r}}|\mathbf{x})$, where $\tilde{\mathbf{r}}$ is an approximation of the true features $\mathbf{r}$. Specifically, we define categories using e anchor points $\tilde{\mathbf{r}} \in \mathbb{R}^d$, where d is the dimension of the feature space. For a given input $\mathbf{x}$, its feature $\mathbf{r}$ is mapped to the anchor point $\tilde{\mathbf{r}}_i$ that exhibits the highest similarity (cosine similarity or Euclidean distance):

$$\tilde{\mathbf{r}}(\mathbf{x}) = \underset{\tilde{\mathbf{r}}_\mathbf{i}}{\operatorname{argmax}}\ sim(\mathbf{r}(\mathbf{x}), \tilde{\mathbf{r}}_\mathbf{i}) \tag{33}$$

Within a batch, the probability coefficients $p_{\boldsymbol{\theta}}(\tilde{\mathbf{r}}_\mathbf{i})$ for the mapping $\tilde{\mathbf{r}}_i$ are estimated by the number of features mapped to the i-th anchor point divided by the batch size. Consequently, the entropy of the anchor points can be computed.

The anchor points are initially randomly sampled uniformly and kept fixed throughout the training process. Although it is possible to update these anchor points with a differentiable approach to better align the observed features. We left it for future research since in practice is was not beneficial.

4.3.2 Differentiation of the feature entropies

Equation (33) is not differentiable, which poses a challenge for gradient-based optimization. To address this issue, we employ the softmax-gumbel function (Jang et al., 2017; Maddison et al., 2017), which enables auto-differentiation. This method utilizes the Gumbel-Max trick (Maddison et al., 2017) to efficiently sample from a categorical distribution in a differentiable manner:

$$\begin{aligned} p_{\boldsymbol{\theta}}(\tilde{\mathbf{r}} = i|\mathbf{x}) &= \frac{\exp((\log(\boldsymbol{\pi}_i(\mathbf{x})) + g_i)/\tau)}{\sum_j \exp((\log(\boldsymbol{\pi}_i(\mathbf{x})) + g_i)/\tau)} \\ \boldsymbol{\pi}(\mathbf{x}) &= \frac{sim(\mathbf{r}(\mathbf{x}), \tilde{\mathbf{r}}_i)}{\sum_{j=1}^{e} sim(\mathbf{r}(\mathbf{x}, \tilde{\mathbf{r}}_j)} \end{aligned} \tag{34}$$

Here, $g_1, g_2, \ldots, g_e$ are i.i.d. samples drawn from a Gumbel distribution with parameters: Gumbel$(0, 1)$. As the softmax temperature τ approaches 0, the samples from the Gumbel-Softmax distribution become one-hot vectors, and the Gumbel-Softmax distribution converges to the categorical distribution $p(\mathbf{z})$. This allows us to perform differentiable computations involving the categorical distribution and its entropy.

4.4 Experiments

In this section, we conduct experiments to demonstrate the relationship between the entropy of the feature space and the ability of the features to be reused for inference on *refined labels*. We approach this by transforming a regression problem into a coarse classification problem and then recycling the feature space to infer *refined labels*.

4.4.1 Relationship between the Entropy of the Feature Space and Transfer Ability

Let $\mathcal{D}^{reg}$ be a dataset representing a regression problem. We convert $\mathcal{D}^{reg}$ into a coarse classification problem where only *coarse labels* (approximations of the regression labels) are available. The dataset is split into training and test sets: $\mathcal{D}_{train}^{classification} = \{(\mathbf{x}_i, \mathbf{y}_i = \text{coarsen}(\mathbf{z}_i)),\ 1 \leq i \leq N_{train}\}$, where coarsen is a function that transforms refined input $\mathbf{z}_i$ into a coarse output $\mathbf{y}i$. We then train a model using the *coarse labels* from $\mathcal{D}_{train}^{classification}$ and utilize the feature space to predict the *refined labels* of $\mathcal{D}^{reg}$.

We compare three criteria in this experiment: Cross Entropy, Label Smoothing, and our proposed method, Feature Information Entropy Regularized Cross Entropy (FIERCE). We use the age estimation dataset based on face pictures (Rothe et al., 2018). The objective is to predict the ages corresponding to the face images. The dataset can be transformed into a binary classification problem by creating *coarse labels* $\mathbf{y} = \text{one\_hot}(\mathbb{1}_{age<36})$, where 36 is the median age. Note that this regression dataset has already been converted into a classification dataset by splitting the ages into multiple classes (X. Liu et al., 2015; Rothe et al., 2015).

We utilize a Resnet-18 (He et al., 2016) model, but instead of using the usual penultimate layer, we average the feature maps to obtain a one-dimensional feature space. We expect that a one-dimensional feature space will preserve the order relation that exists in the true age labels to some extent.

To estimate the refined ages, we compute the features of 10,000 samples. We rank the features by their values and, assuming that the probability distribution of the ages is known, we map it to the distribution of the features using an Optimal Transport 1D mapping (Peyré & Cuturi, 2019). For more robust results, we interpolate this prediction with the average age of individuals in the considered class. We report the evolution of the Mean Squared Error (MSE) and the entropy on Figure 20.

When using Label Smoothing or raw Cross-Entropy, we observe two phases. First, the MSE decreases as the accuracy improves. Then, a much longer second phase starts (around epoch 50) where the MSE increases while the accuracy remains stable. Similarly, the entropy of the feature space reaches a maximum before decreasing, and the decrease is negatively correlated with the drop in regression ability. These phases are very similar to the evolution of the mutual information $I(\mathbf{r}, \mathbf{x})$ described in (Shwartz-Ziv & Tishby, 2017), providing strong motivation for the introduction of FIERCE.

Our proposed method, FIERCE, provides the lowest MSE and impacts the entropy of the feature space by increasing it around epoch 200. In Figure 24, we show the evolution of the MSE and the interpolation coefficient with respect to the hyperparameter λ. Tuning λ allows us to retrieve more information on the *coarse labels* from the feature space, as the optimal coefficient of interpolation keeps increasing. This serves as more evidence of the relationship between entropy in the latent space and the ability to generalize to finer labels.

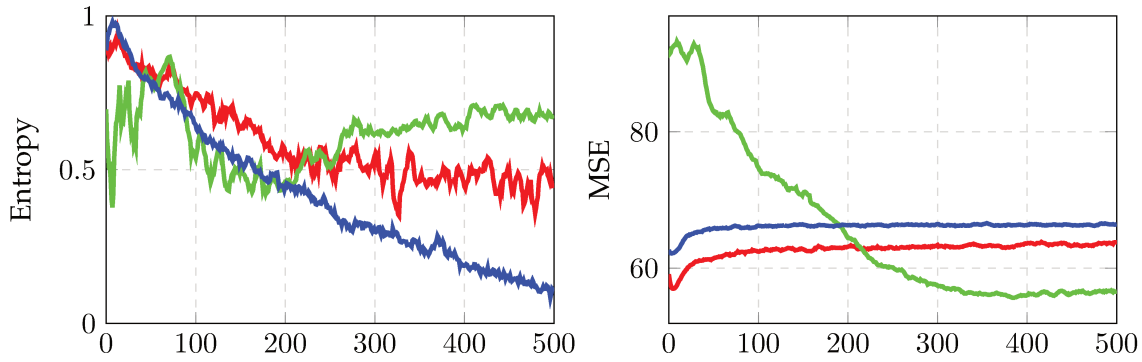
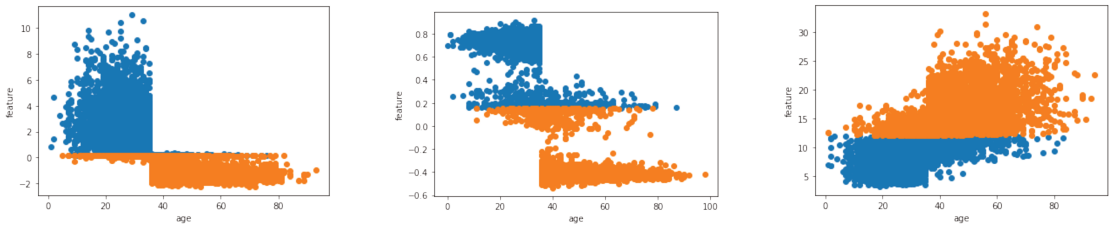


Figure 20 – Evolution of the Entropy of the feature space (rescaled between 0 and 1) and Mean Squared Error on the age regression dataset (Rothe et al., 2018) for the different criteria: Cross Entropy (red), FIERCE (proposed) with $\lambda = 0.3$ (green), Label Smoothing (blue). We observe that models trained with standard cross-entropy or label smoothing briefly display a minimal MSE before stabilizing to a larger value. On the contrary, the proposed FIERCE method reaches an overall lower MSE that is maintained through additional training epochs. This ability to reach and maintain lower MSE is negatively correlated with the entropy measured in the feature space, shown in the left figure.

We also observe that Label Smoothing does not perform well in this dataset. The final MSE is worse than the one given by Cross Entropy, and the entropy keeps dropping. In Figure 21, we present the feature space of each criterion. The dimension of the feature space is one-dimensional (1-d), allowing us to directly visualize the feature spaces.

Ideally, the features should be aligned on a line or at least contained within an ellipse-like shape. For Cross Entropy, we observe that the features are well separated by the median of the ages (36), but they appear randomly distributed in each part of the graph. As for label smoothing, we notice an increase in the area of uncertainty in the middle without any improvement in the MSE. This suggests that this uncertainty does not provide relevant information regarding the ages, as the features are randomly distributed in this area. In contrast, with FIERCE, the features are much more aligned and exhibit a more elliptical distribution, which correlates well with the lowest MSE achieved. This alignment and elliptical shape indicate that FIERCE effectively captures meaningful patterns in the feature space, leading to improved age estimation performance.



(a) Cross Entropy **(b)** Label Smoothing **(c)** FIERCE

Figure 21 – Feature space (1-d) with respect to the ages for each criterion on the Age Estimation dataset. X-axis corresponds to the ages and Y-axis to the values of features.

4.4.2 Retrieving information on refined label from the Output distribution

This section investigates the relationship between the smoothness of the output distribution and its ability to convey information regarding *refined labels*. Previous studies

have demonstrated that smoothing the output distribution using entropy regularization techniques reduces peaks and provides a wider range of output values (Pereyra et al., 2017; Szegedy et al., 2016). The common interpretation of the smoothed output values is as a measure of uncertainty in predictions (C. Guo et al., 2017). However, an alternative viewpoint is to consider these output values as information on the *refined labels*, such as degrees of similarity with each class.

To explore the validity of this interpretation, we utilize a dataset where the output uncertainty can be readily interpreted as the *refined labels*. For this purpose, we use a dataset consisting of crops from a single hyperspectral image in the remote sensing domain (Ghamisi et al., 2017). This dataset is composed of $w \times h$ crops from a hyperspectral image, capturing reflectance values for c contiguous wavelengths (channels) in the visible and near-infrared domains. Two essential interconnected problems in hyperspectral imaging are supervised semantic pixel classification and spectral unmixing (Ghamisi et al., 2017). In the classification task, the objective is to assign a class to each pixel of a predefined set of identified classes. On the other hand, unmixing can be seen as a refinement of classification, accounting for the possibility that objects to be detected may be smaller than the size of a pixel. Therefore, the goal in unmixing is to predict the proportion of each material (referred to as abundances) in each pixel.

To transform the regression problem of unmixing into a classification problem, we use *coarse labels* $\mathbf{y} = \text{one\_hot}(\text{argmax}_i\ \mathbf{z}_i)$ where $\mathbf{z}_i$ is the proportion of material i: $\mathbf{z} \in [0,1]^m, \sum_i(\mathbf{z}_i) = 1$. We train a fully connected neural network $f\boldsymbol{\theta}$ with two hidden layers, where $\boldsymbol{\theta}$ denotes the network's parameters.

The class probabilities of the network can easily be interpreted as the proportions of each material. Thus we evaluate the regression performance of the network using the output of the network on the entire image with the Mean Squared Error (MSE) metric. The raw MSE is then defined as follows:

$$\text{raw MSE}(f_{\boldsymbol{\theta}}) = \mathbb{E}_{\mathcal{D}^{reg}}\left[\sum_i \left(q_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})_i - \mathbf{z}i\right)^2\right]^{1/2}, \tag{35}$$

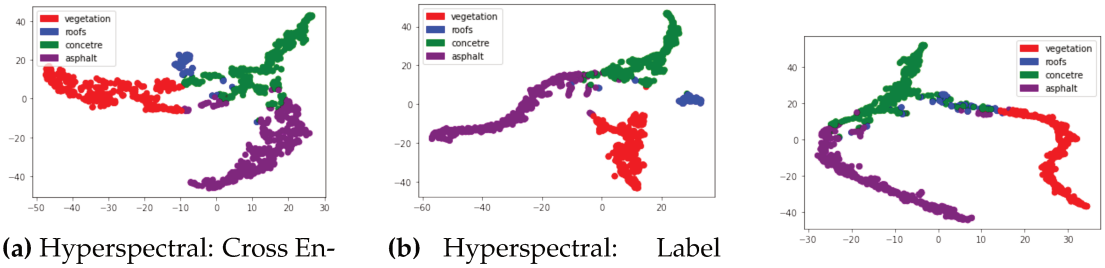
where $\mathcal{D}^{reg}$ represents the dataset for regression, and $q_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ denotes the predicted class probabilities. We also compute another metric called transfer MSE, which evaluates the transferability of the inferred feature spaces for performing the regression task. This assessment involves employing the refined labels and, and for each criterion, we retrain the last layer of the network with a regression criterion (Mean Squared Error), keeping the others layers (feature space) inferred with the classification criteria frozen. The transfer MSE, denoted as $transfer\ MSE(f_{\boldsymbol{\theta}})$, is then computed with the new heads. It is important to note that in practical scenarios, this approach is not viable, as refined labels are assumed to be inaccessible. However, in our context, it serves as a valuable metric to evaluate which feature space retains the most pertinent information from the refined labels.

Table 7 presents the MSEs estimated on the hyperspectral dataset for the different criteria. We observe that both Label Smoothing and FIERCE demonstrate a strong ability to recover the regression labels directly from the output, with lower MSEs (averaged over all materials) compared to Cross Entropy. However, FIERCE exhibits the highest transferability (lowest transfer MSE), indicating that its feature space allows for the retrieval of more information on the *refined labels* than other criteria. As an attempt to better understand the geometric impact of our proposed method, we embed features into a 2D space using t-Distributed Stochastic Neighbor Embedding (TSNE) (Van der

Table 6 – MSEs estimated on the hyperspectral dataset for the different criteria. Raw MSE given directly from the output of the network, transfer MSE given by transfer learning.

	Cross Entropy	Label Smoothing	FIERCE ($\lambda = 1$)
Raw MSE	0.186 ± 0.001	$\mathbf{0.066} \pm 0.1$	0.130 ± 0.008
Transfer MSE	0.177 ± 0.3	0.017 ± 0.08	$\mathbf{0.006} \pm 0.001$

Maaten & Hinton, 2008). We observe that both Cross Entropy and Label Smoothing split the feature representations of different classes into separate and distinct clusters. However, FIERCE stands out by preserving the continuity of features from one class to another. The transitions between clusters are likely to correspond to mixed pixels containing proportions of several materials, such as asphalt and concrete.



(a) Hyperspectral: Cross Entropy

(b) Hyperspectral: Label Smoothing

(c) Hyperspectral: FIERCE

Figure 22 – Hyperspectral dataset: We perform an embedding of the features into a 2D space using t-Distributed Stochastic Neighbor Embedding (TSNE).

4.4.3 Output Distribution as an indicator of transferability

Considering the difference between the raw MSE and transfer MSE, we suspect that the output distribution alone is not an accurate indicator of the ability to transfer knowledge. Hence, we explore additional metrics, such as reliability diagrams (see Figure 23), Expected Calibration Error (ECE), Maximum Calibration Error (MCE), mutual information, and stability (see Table 7, to distinguish Label Smoothing from FIERCE.

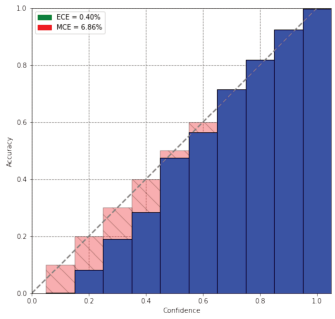
As reported in Table 7, FIERCE achieves the lowest mutual information on the hyperspectral dataset, while also demonstrating a lower raw MSE compared to Cross Entropy, along with the lowest transfer MSE compared to all criteria. Conversely, on the age estimation data, FIERCE exhibits the highest mutual information. Notably, we observed some degree of relationship between Mutual Information and Stability metrics, but they both struggled to distinguish the criteria effectively. For instance, on the hyperspectral dataset, Label Smoothing and FIERCE exhibit very similar Stability values but different transfer MSE. Similarly, on the age estimation data, all metrics are very close for each criterion, despite substantial differences in MSEs. Consequently, these metrics appear unreliable in evaluating transfer capabilities.

We also present the reliability diagrams in Figure 23, where we can observe that neither Label Smoothing nor FIERCE significantly improve reliability. Interestingly, Label Smoothing appears to hinder confident predictions (outputs close to 1), which could explain why the raw MSE is smaller on the hyperspectral dataset.

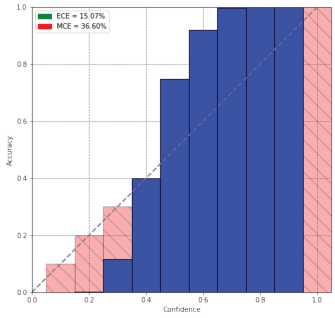
In conclusion, the output distribution alone is not a reliable indicator of the information contained in the feature space. The raw MSE suggests that FIERCE and Label Smooth-

Table 7 – Stability and Mutual information estimated on the hyperspectral dataset for the different criteria. Raw MSE given directly from the output of the network, transfer MSE given by transfer learning.

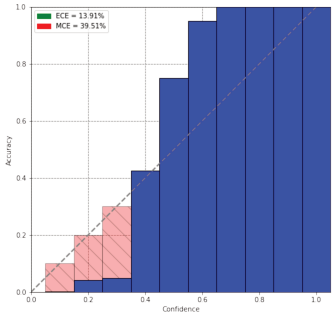
		Cross Entropy	Label Smoothing	FIERCE
Hyperspectral	Mutual Information	1.71	0.55	0.39
	Stability	0.88	0.93	0.93
Age estimation	Mutual Information	0.10	0.12	0.14
	Stability	0.81	0.80	0.78



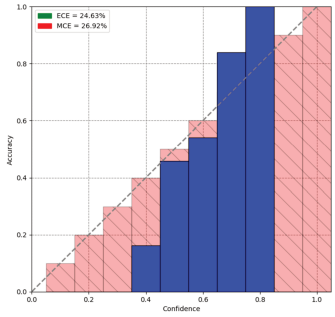
(a) Hyperspectral: Cross Entropy



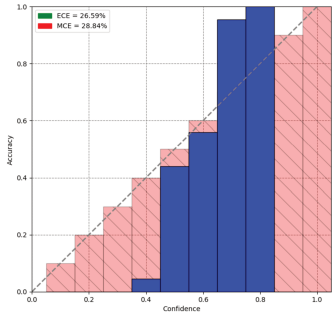
(b) Hyperspectral: Label Smoothing



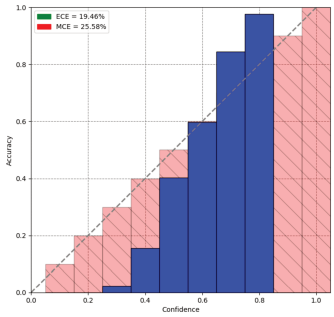
(c) Hyperspectral: FIERCE



(d) Age Estimation: Cross Entropy



(e) Age Estimation: Label Smoothing



(f) Age Estimation: FIERCE

Figure 23 – Reliability diagrams for each criterion on the hyperspectral and age estimation datasets.

ing would perform similarly in transfer, while the transfer MSE of Cross Entropy is significantly worse than the raw MSE. This indicates that the feature space of Cross Entropy is overspecialized and biased, hindering transferability.

4.4.4 Transfer learning: fewshot applications

As demonstrated by the previous experiments, the FIERCE method allows for the retrieval of more information from the feature space compared to Cross Entropy or Label Smoothing. While this may not be as relevant in classification tasks where the goal is to select only the meaningful features for class discrimination, we believe that the proposed method could be advantageous in transfer learning scenarios, where the feature space can be leveraged for different, possibly more subtle, tasks. In the following section, we present results obtained by testing FIERCE on different few-shot prob-

lems (Mangla et al., 2020).

Few-shot learning involves first training a model on a generic dataset and then using the feature space to classify new classes with a limited number of samples.

Table 8 – Classification accuracy (1-shot) on novel tasks for CIFAR-FS and CUB. Displayed accuracy are averaged over 10,000 random fewshot runs. Confidence intervals are computed over 10 randomly initialized training of each model under each criterion.

	Cross Entropy	Label Smoothing	FIERCE
CIFAR-FS	64.32 ± 0.7	65.76 ± 0.57	$\mathbf{66.16} \pm 1.04$
CUB	59.61 ± 0.84	59.87 ± 0.81	$\mathbf{62.18} \pm 0.59$

To evaluate our method, we use two datasets: CIFAR-FS (Y. Liu et al., n.d.) with a Resnet18 model, and CUB (Wah et al., 2011). We compare the three criteria: Cross Entropy, Label Smoothing, and FIERCE. CIFAR-FS is a dataset consisting of 60,000 color (RGB) images of size 32×32 pixels. The dataset is divided into three splits, each containing 64 training classes, 16 validation classes, and 20 testing classes, with 600 examples in each class. CUB is composed of 11,788 images of size 84×84 pixels and has 200 classes. We use the splits: 100 base classes and 50 novel classes recommended in (Hu et al., 2021).

On both datasets, our method outperforms the baseline and Label Smoothing. Interestingly, despite the distinct impact on the feature space of Label Smoothing and FIERCE, both methods improve the accuracy in transfer.

4.4.5 Additional experiments: Influence of hyperparameters

In this section we provide the ablation studies used to select the hyperparameters in the experiments reported in Section 5. Let us recall that FIERCE introduces two hyperparameters: λ, tuning the influence of the entropy constraint and e, the number of anchor points of $\mathcal{E}$ used to approximate the entropy of the features.

Influence of the number of anchor points e: On Figure 24 we plot for the hyperspectral and age estimation datasets the influence of e on the MSEs. We note that after a certain limit, $e = 100$ for the hyperspectral dataset and $e = 75$ for age estimation, the MSE remains stable. These values are around the batch size for each dataset: 200 for the hyperspectral dataset and 64 for age estimation. Therefore, we take for the other experiments a value of e close to the batch sizes.

Influence of λ: we plot for each dataset the influence of λ. For the hyperspectral dataset we report the evolution of the raw MSE and the transfer MSE; for age estimation the evolution of the MSE.

TSNE-Representation on CIFAR10: In addition to the representations we have generated for the age estimation dataset and the hyperspectral dataset, we have applied the same methodology to the CIFAR-10 dataset. The results are illustrated in Figure 25. We observe that FIERCE effectively reduces the distances between different classes while smoothing the overall representation. Notably, we observe a significant reduction in the intra-class distance between dogs and cats compared to Label Smoothing or Cross Entropy. This reduction suggests that the network is better at detecting semantic similarities between classes, potentially leading to improved classification performance.

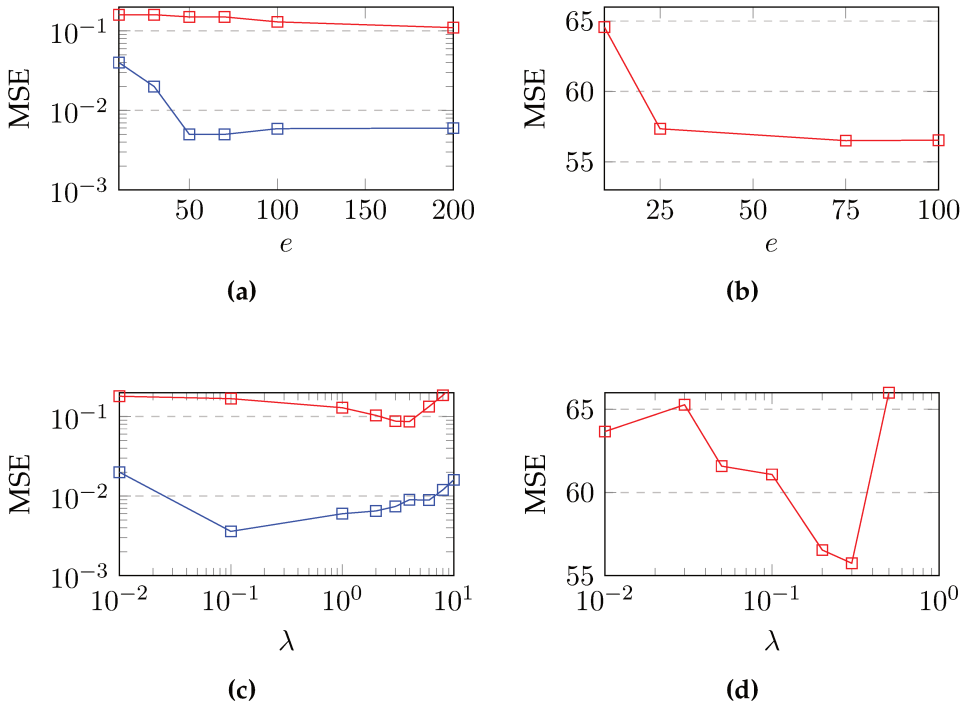
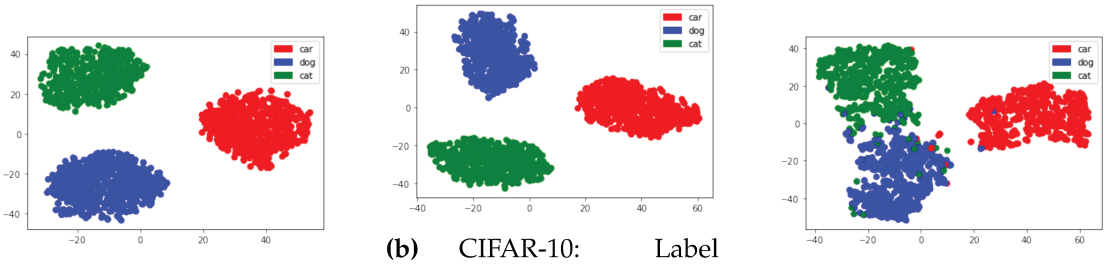


Figure 24 – Evolution of the MSE with respect to the number of entries e (first row) and λ (second row). On the left, the hyperspectral dataset: raw MSE (red), transfer MSE (blue).



(a) CIFAR-10: Cross Entropy

(b) CIFAR-10: Label Smoothing

(c) CIFAR-10: FIERCE

Figure 25 – TSNE representation of the feature space (penultimate layer) for the different criteria. First row on the hyperspectral dataset (dim = 100), second row on CIFAR-10.

4.4.6 Does our method correctly approximate entropy?

To assess the quality of our entropy approximation, we conduct an experiment on the hyperspectral dataset using Cross Entropy. During training, we track the entropy of a feature space, and concurrently, we estimate the entropy using our non-differentiable method with the one-hot tensor. Both entropy measures are presented in Figure 26, with both Entropy and Approximation values rescaled between 0 and 1.

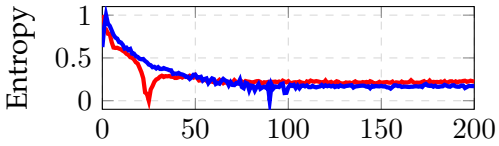


Figure 26 – Entropy of the feature space (in blue) and approximation of our method (in red) on the hyperspectral dataset.

Overall, our approximation closely follows the actual entropy of the feature spaces, with a correlation coefficient of 0.86 between the two measures. While there are a few anomalies, the general trend indicates that our method provides an accurate estimation of entropy for the feature spaces.

4.5 Limitations and Perspectives

4.5.1 Limitations

We observed that the entropy of the feature space was related to the transferability, particularly in regression tasks. However, the entropy may not be the optimal criterion to use as a regularizer. For instance, in (Shwartz-Ziv & Tishby, 2017) suggests that the mutual information could be used although estimating it a differentiable way is to an easy task (Tschannen et al., 2019). On regression datasets, our method yields the lowest MSEs, but it does not significantly outperform raw Cross Entropy during the first epochs of training. The main advantage of the proposed method lies in its ability to avoid early stopping.

Another challenge in our method is the computation of a high-dimensional feature space, which must also be auto-differentiable. Our estimation relies on random anchor points, and the generation and distribution of these anchor points may impact the geometry of the feature space. With high values of λ, we force the features to be uniformly aligned with these anchor points. One potential solution would be to use a learnable dictionary, as done in VQ-VAE (Van Den Oord, Vinyals, et al., 2017), but this would introduce an additional loss term and another hyperparameter.

Finally, tuning the hyperparameter λ may be challenging in practice, especially in domains where there is no clear insight into the future use of the coarsely trained model.

4.5.2 Perspectives

In recent years, the field of Self-Supervised Learning has seen the emergence of various criteria, all aimed at attaining highly abstract and universally applicable data representations (Radford et al., 2015). One notable approach involves the application of the contrastive loss (T. Chen et al., 2020; Tian et al., 2020b). This loss operates on pairs of data points, typically called "anchor" and "positive" samples, and aims to maximize the similarity (or minimize the distance) between similar pairs while pushing apart dissimilar pairs. The loss function encourages the model to map similar data points closer together in the feature space and push dissimilar points further apart.

Contrastive learning can be seen as a mechanism that encourages the model to disperse representations of distinct data points in the feature space, promoting a higher level of entropy. This approach serves a dual purpose: it enables the model to learn meaningful data representations through positive sample pairs, while concurrently maximizing entropy via the incorporation of negative sample pairs.

However, it is important to note that contrastive loss is not the sole criterion that addresses these objectives (Caron et al., 2020; Radford et al., 2015; Zhuang et al., 2020a); other approaches such as the triplet loss (Schroff et al., 2015) and InfoNCE Loss (Oord et al., 2018) also target similar outcomes. As a future research direction it would be interesting to conduct a comprehensive study comparing various methods closely related to entropic regularization of the feature space, and analyse their impacts on transferability. This endeavor aims to identify the most effective techniques and gain a deeper understanding of their underlying principles.

Additionally, it is essential to consider the phenomenon known as dimensional collapse in neural networks (Glorot & Bengio, 2010; Jing et al., 2021), which can occur when the learned representations become excessively compressed or entangled, potentially limiting the model's capacity to capture meaningful information in high-dimensional data spaces. Intuitively, dimensional collapse seems closely related to the second phase that we observe in this chapter and in (Shwartz-Ziv & Tishby, 2017). **It would be beneficial to investigate whether this relationship holds true**.

To monitor changes in the geometry of the feature space during learning, various tools such as graphs (Cosentino et al., 2022; Papernot & McDaniel, 2018) or metrics like the rank of the covariance matrix of the feature space (Garrido et al., 2023; Jing et al., 2021) could be employed. Furthermore, by constraining the geometry of the feature space as done in (Bontonou et al., 2019), it might be possible to ensure better transferability. In a more recent study (Y. Guo et al., 2023), the authors propose a method in which features are mapped to randomly distributed anchors on a hypersphere. This approach enables the introduction of repulsive forces between samples and clustering classes, offering a novel perspective on feature space manipulation. The idea of using anchors points is similar to our except that they imposes a geometric prior.

4.6 Conclusion

In summary, we have showcased the neural network's capability to extend its generalization beyond the tasks it was trained on, particularly when utilizing *coarse labels*. We have observed two distinct phases during the learning process. Initially, information regarding *refined labels* is retrieved, followed by a second phase where the focus shifts towards achieving accurate predictions for *coarse labels*. Our experimentation across diverse datasets and benchmarks has shed light on the crucial role of feature space entropy in the realms of transfer learning and achieving more precise label predictions.

Through the development of our proposed method, Feature Information Entropy Regularized Cross Entropy (FIERCE), we aimed to enhance the entropy of the feature space while retaining the essential information required for addressing more intricate tasks. The experimental results demonstrated that our FIERCE method outperforms other entropy regularization techniques, such as Label Smoothing and Cross Entropy, in preserving information within the feature space. Furthermore, we have introduced a novel differentiable approach to estimate the entropy of the feature space, thereby enabling the training of deep learning models with the proposed regularization. Overall, the findings from this study indicate that the FIERCE method holds great promise for tasks involving coarse to refined classification and transfer learning.

Chapter 5

Transfer in Classification: How Well do Subsets of Classes Generalize?

Contents

5.1 Introduction

5.1.1 Generalizing beyond the training task: Foundation models

In the preceding chapter, our investigation centered on the capacity of deep learning architectures to exhibit the ability to generalize beyond their training tasks, specifically within specific scenarios. Our findings, as well as others papers (Touvron et al., 2021; Wu et al., 2017), demonstrated that deep learning models, when trained using *coarse labels*, exhibited the ability to retrieve information on *refined/fine labels* related to the *coarse labels*. In such scenarios the input domains remained consistent –only the labels were changed.

In the present chapter, our focus shifts to a more general context within the realm of classification that aligns more with the current context in the field with for instance the advent of foundation models. Indeed, as highlighted in the introduction of the manuscript, there was a resurgence in neural network-based models during the early 2010s (Bommasani et al., 2021). This revival was propelled by the availability of large datasets and architectures with substantial parameters (J. Deng et al., 2009; LeCun, 2015b). Quickly, researchers realized that these large pre-trained models could be reused in various domains and tasks (J. Deng et al., 2009). Subsequently, these pretrained models evolved into larger models called foundation models (Bommasani et al., 2021). Such models are trained on auxiliary tasks and demonstrate remarkable knowledge transfer capabilities across a wide spectrum of tasks and domains (Redmon et al., 2016; C. Zhou et al., 2021; K. Zhou et al., 2022). The remarkable versatility and effectiveness of foundation models triggered a shift of paradigm within the research community, commonly referred to as "homogenization" (Bommasani et al., 2021). This shift led a large number of researchers to adopt widely-used pre-trained models like BERT (Devlin et al., 2018), GPT (Radford et al., 2018), and CLIP (Radford et al., 2021), which have since become established benchmarks in the field.

On the paper, foundation models can be understood as serving a role akin to transfer learning. When training on a comprehensive, representative universal dataset, the necessity to adapt to distinct domains diminishes. Nevertheless, recent research papers, as illustrated in (Touvron et al., 2021; Wu et al., 2017) as well as our previous chapter, highlight the advantages of transfer learning, even in scenarios where the downstream task involves fine-grained refinements within the original domain. These recent advancements compel us to develop a pragmatic theoretical framework for exploring and comprehending the concept of transferability.

5.1.2 Transfer Learning

Transfer learning aims to leverage knowledge gained from a source task and dataset, even if they bear little resemblance to the target task or dataset, with the goal of reducing the learning cost and facilitating the solution of the target task (Zhuang et al., 2020a). An approach particularly valuable when acquiring a substantial volume of labeled data proves costly or when the risk of overfitting needs to be mitigated.

Initially, models were trained for specific tasks, such as classification using labeled datasets that correspond to specific data domains (Caruana, 1997). Then the challenge lied in extending their capabilities to address other tasks or adapt to different domains. Originally, transfer learning was limited to domain adaptation. However, as the field has evolved, the concept of transfer has expanded to encompass adjustments in both task and data domain, as mentioned in (Zhuang et al., 2020a), (Pan & Yang, 2010), and (Radford et al., 2015).

Such ability has been gained thanks to new training methods, in particular within Self-Supervised Learning, where models are trained on pretext tasks that do not require human-labeled annotations, thereby allowing models to acquire meaningful and general data representations (He et al., 2020) and SimCLR (T. Chen et al., 2020).

Transfer learning can be categorized into three types: transductive, inductive, and unsupervised (Pan & Yang, 2010; Zhuang et al., 2020a). Transductive transfer learning arises when only the source data is labeled, while inductive transfer learning encompasses labeling both the source and target data. Unsupervised deep transfer learning, on the other hand, deals with scenarios where none of the data is labeled.

Furthermore, transfer learning can be dissected into four fundamental approaches, as outlined in (Pan & Yang, 2010; Zhuang et al., 2020a): instance-based (Jiang & Zhai, 2007), feature-based or mapping-based (Raina et al., 2007), parameter-based or model-based (Finn et al., 2018), and relational-based or adversarial-based strategies (Davis & Domingos, 2009). In this chapter, we will focus on feature-based methods as they are widely leveraged in many transfer learning applications, such as Few-shot Learning, for instance.

5.1.3 Few-Shot Learning

Few-Shot Learning (Fei-Fei et al., 2004) constitutes a research area that aligns with the theoretical framework that we present in this chapter. The objective of Few-Shot is to infer model that recognize new classes (*novel classes*), with a limited number of labeled examples (shots) per class, even when the model was originally trained on a different set of classes (*base classes*). This transfer learning scenario reflects well real-world situations where acquiring abundant labeled data for each new class is expensive. Various approaches to few-shot learning exist, with one common method being feature-based (Y. Wang et al., 2020). In this approach, the model, initially trained on *base classes*, functions as a feature extractor for *novel class* samples. Subsequently, a new classifier, often a simple linear one, is trained on these features to adapt to novel classes.

5.1.4 Problem Statement

Despite the widespread success of transfer learning, the underlying mechanisms that enable this ability remain unclear and, to our knowledge, insufficiently formulated. Progress in transfer learning has incited several authors to delve into fundamental questions within the domain. In (Atanov et al., 2022), the authors introduce a method called *Task Discovery* for identifying tasks (subsets of classes) on which a trained network can generalize effectively. This method involves computing similarity scores, with higher similarity scores suggesting that networks are more likely to converge to stable solutions. The researchers demonstrate that employing human labels leads to superior generalization compared to random labels, facilitating the identification of tasks or sets of classes that result in optimal generalization in terms of error rate. However, it is crucial to note that our work differs from theirs as we provide a comprehensive theoretical framework, whereas their focus centers on identifying subsets of classes that lead to stable solutions.

Prominent theoretical contributions in transfer learning encompass also (Zhuang et al., 2020a) and (Pan & Yang, 2010), where authors categorize transfer learning approaches and furnish definitions for key concepts such as tasks and data domains. Additionally, in (Tripuraneni et al., 2020; Yang et al., 2013), the authors derive theoretical bounds on convergence rates in specific scenarios.

While these existing approaches offer valuable insights, we think it is possible to complement them by introducing new perspectives. We aim to provide a simple yet meaningful definition of transfer learning within classification and to leverage it for investigating fundamental questions in the field, such as which subsets of classes generalize to which others.

We present a comprehensive theoretical framework for characterizing the transferability of models trained on one set of classes (C) to another set of classes (C'), even in scenarios where the input domains may differ. As discussed earlier, this generalization is particularly sought in the context of few-shot learning that involves tasks where the training and target domains do not overlap (Fei-Fei et al., 2004; Y. Wang et al., 2020), at least in terms of semantics.

More generally, this framework serves as a tool for investigating fundamental questions related to learning, such as:

1. Can we identify informative indicators of how well a model trained on one set of classes (C) will perform on another set of classes (C')?
2. Which classes play the most crucial role in the learning process when fine-tuning a model? Which set of classes lead to the best generalization and transferability ?

To investigate these fundamental questions and illustrate the effectiveness of our methodology, we carry out experiments in various scenarios. In doing so, we highlight our ability to identify the optimal subsets of classes during processes such as fine-tuning, training models from scratch, and few-shot learning.

5.2 Proposed Theoretical Framework

In this section, we introduce a theoretical model within the classification domain, where the goal is to separate each class from the others. To this aim, we employ ***models*** that function as bi-partitions. For instance, a hyperplane splits a Euclidean space into two distinct regions, with the ideal scenario being one class occupying one side and the other class residing on the opposite side of the hyperplane. While deep learning architectures often employ a one-vs-all classifier, our approach takes a more fundamental stance: **we consider a set classes to be successfully separated only when each pair of classes can be distinctly separated**. In essence, we approach the classification problem as a partitioning challenge, wherein, for every pair of classes, there must exist at least one partition in which one class is allocated to one set while the other class belongs to the opposite set. Embracing this perspective reveals that certain bipartitions or models exhibit greater expressive power than others. Consequently, we introduce an ordering relationship among these models.

5.2.1 General Case

Let us denote E a space, and C a set consisting of n disjoint subsets of E. Each element in C is referred to as a class. We define a ***model*** as a bipartition of the space E, denoted as $\mathcal{M} = \{M^1, M^2\}$, where M^1 and M^2 are two disjoint subsets of E. The bipartition separates two classes I and J when one class is included in subset M^1 while the other class is included in subset M^2.

Definition 17: Model

A ***model*** $M = \{M^1, M^2\}$ is a bipartition of the space E, i.e, $E = M^2 = E \setminus M^1$.

Let us say that a ***model*** M separates two classes I and $J \in C$, if: $(I \cap M^1 = \emptyset) \wedge (J \cap M^2 = \emptyset)$ and we note $I \perp_M J$.

In practical applications, models are typically inferred from data. In this context, we make the assumption that for every pair of classes, a dedicated model is learned to effectively separate that particular pair. **Without this hypothesis, the problem would be inherently inseparable**. Consequently, each model is uniquely associated with a specific pair of classes on which it was trained.

Definition 18: Model associated with a pair of classes

We note $M_{I,J}$ the ***model*** learned on $\{I, J\}$ and that separates I and J: $I \perp_{M_{I,J}} J$.

The method by which we derive these models is intentionally left implicit at this stage, and will be elaborated upon in the methodology section. Furthermore, it is worth noting that with the previous definition, some models may separate not only the specific pair they are associated with but also others pairs of classes. Therefore, we introduce for each model M, the set $\mathcal{C}(M)$ that comprises all the pairs of classes effectively separated by the model M.

Definition 19: Set of separable pairs

Let us note $\mathcal{C}(M)$ the set of pairs of elements in C that are separated by M.

This definition reveals that certain models separate more pairs than others, demonstrating greater expressiveness. For instance, a model M may successfully separate all the pairs that another model M' separates. This observation naturally prompts us to establish the following order relationship:

Definition 20: Ordering models

Let us consider two models M, M'. The model M is said to be less expressive than the model M' if: $\mathcal{C}(M) \subseteq \mathcal{C}(M')$, and we use the notation $M \leq M'$. Moreover, the relation $\leq$ is an order relationship.

Models that separate the same set of pairs are said to be equivalent.

Definition 21: Equivalent models

Two models M, M' are equivalent if $M \leq M'$ and $M' \leq M$, denoted $M \equiv M'$.

The previous order relationship can be effectively visualized with Hasse Diagrams – providing a clear representation of the partial order among models. In the Hasse Diagram, each model is depicted as a node, and the presence of an arrow from M' to M signifies that M is less expressive than or equivalent to M'. It facilitates the identification of the most expressive models. Models with lower expressiveness can be disregarded since they can be superseded by other models for distinguishing the pairs they separate. Figure 27 provides illustrations of Hasse diagrams for two distinct datasets, each restricted to only 4 classes.

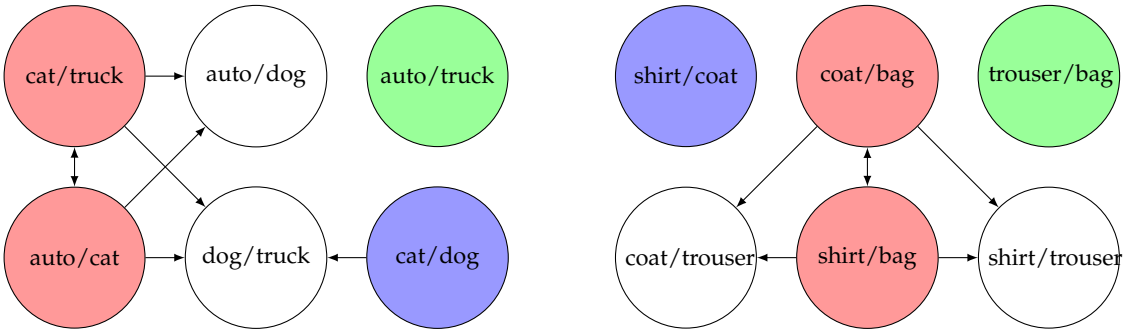


Figure 27 – Hasse Diagrams illustrating the hierarchical order of models learned on pairs of classes. Each model's is characterized by the pairs of classes it successfully separates. Arrows pointing from one model A, to another B, signify that model A exhibits greater expressive capacity, since it effectively separates all the pairs that model B separates. The models highlighted in color represent the most expressive and are referred to as 'fundamental' models. Models sharing the same colors are considered equivalent, as they separate identical pairs of classes. Conversely, uncolored models are considered redundant since the pairs they separate can be separated by others models. On the left, we examine a scenario with 4 classes from CIFAR-10 : auto, cat, dog, truck. The diagram illustrates that cat/truck and auto/cat are not only equivalent but also more expressive than 'auto/dog' and 'dog/truck'. In simpler terms, models 'auto/dog' and 'dog/truck' can be omitted since they do not offer any additional separations compared to cat/truck and auto/car. On the right, considering 4 classes from FASHION-MNIST : bag,coat,shirt,trouser. The diagram reveals that coat/bag and shirt/bag are equivalent and possess greater expressiveness than coat/trouser and shirt/trouser.

Our focus now shifts to the models with the highest expressiveness –those that cannot be acquired through any other means except by using an equivalent model. These models are trained on specific pairs of classes referred to as ***fundamental*** pairs, as they are indispensable and cannot be disregarded.

Definition 22: Fundamental pair

A pair of classes $\{I, J\}$ is said to be ***fundamental*** if $I \perp_{M_{K,L}} J \Rightarrow M_{I,J} \equiv M_{K,L}$.

Of significant interest is the determination of the minimum number of pairs required to separate all the classes. This number is referred to as the 'fundamental number' because it involves that it is impossible to learn all fundamental models with a smaller number of pairs.

Definition 23: Fundamental number

The ***fundemental number*** $\mathcal{F}(C)$ is defined as the minimum cardinal of a set S of models such that: $\forall I, J \in C, \exists M \in S, I \perp_M J$. Note that by definition we have an upper bound: $\mathcal{F}(C) \leq \binom{n}{2}$.

Instead of defining the fundamental number based on pairs, we can utilize models, as demonstrated by the following theorem. This approach enables us to perceive the fundamental number as the minimum number of models required to effectively separate all the pairs.

Theorem 6

Consider the set of all fundamental pairs. Let us remove pairs until this set contains only the ones whose associated models are not equivalent to one another. Then the cardinality of this set is noted $\mathcal{F}(C)$.

Proof. Let us consider such a set of fundamental pairs P and a set of separators S of cardinal $\mathcal{F}(C)$.

Let us consider p an element of P. By construction of P, p is separated by a single separator of S. Otherwise the element would have an equivalent leading to a contradiction. Hence the cardinal of P is smaller than or equal to the cardinal of S.

Conversely, let us consider a separator $M \in S$. We show it separates one single fundamental pair. By definition of P, the following set: $P \cap \mathcal{C}(M)$ has no more than one element. By contradiction, let us assume that this set is empty, i.e, $\mathcal{C}(M)$ contains only pairs which are not fundamental. So, for any element $c \in \mathcal{C}(M)$, there is a fundamental pair $p \in P$ s.t $c < p$. But p is necessarily separated by a separator $M' \in S$ and consequently c is also separated by M'. At the end, we have two separators M, M' for c. Since the reasoning is true for any $c \in \mathcal{C}(M)$, then $S \setminus M$ separates all pairs of C. Hence a contradiction with the cardinal of S being equal to $\mathcal{F}(C)$. Therefore, each model of S separates a single fundamental pair; the cardinal of S is smaller or equal to the cardinal of P. In conclusion, the cardinal of P equals $\mathcal{F}(C)$. □

Definition 7 established an upper bound for $\mathcal{F}(C)$. Interestingly, it is possible to establish a lower bound for $\mathcal{F}(C)$ as well. Remarkably, both of these bounds can be achieved by at least one example.

Theorem 7

Let $n \geq 2$. Then the fundamental number $\mathcal{F}(C)$ is bounded as follows:

$$\log_2(n) \leq \mathcal{F}(C) \leq \binom{n}{2}.$$

Both bounds are achieved by at least one example.

Proof. Let us prove the lower bound. We consider $n \geq 2$, we want to distribute the pairs of C on a minimal number of models $\mathcal{F}(C)$. First, we note that in any case, the minimum can be achieved when all elements of C belong to one partition of each model: if one element X does not belong to any partition of a model M it can be added without consequences to one of the partitions M^1, M^2.

By doing so, the models describe all elements of C which can be represented by a binary encoding: 0 for element included in M^1 and 1 for element included in M^2. The problem is then equivalent to the encoding of a source alphabet C into a decodable code over an binary alphabet. Then according to the Kraft-McMillan inequality, the following bound holds: $\mathcal{F}(C) \geq \log_2(n)$ (Kraft, 1949).

□

We provide one example that reaches the upper bound.

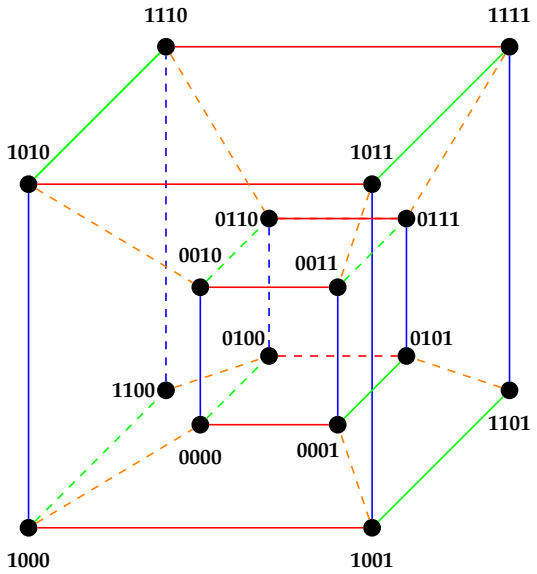


Figure 28 – Encoding of 16 classes with a minimal number of bi-partitions (models): $log_2(16) = 4$. Pairs/edges with same color correspond to equivalent models.

Proof. Let be n an even number, we define $C = \{\{1,2\},\{3,4\},\cdots,\{n-1,n\}\}$ composed of n sets. Let us consider a pair $I, J \subset C$. Without loss of generality, we assume that $I < J$. We define the model $M_{I,J}$ a biparition $(M^1_{I,J}, M^2_{I,J})$ as follows:

$$
\begin{aligned}
&I \subset M^1_{I,J} \text{ and } J \subset M^2_{I,J},\\
\forall X \in \{1,\cdots,n\} \setminus (I \cup J),\ &X \mod 2 = 0 \Rightarrow X \subset M^1_{I,J},\\
\forall X \in \{1,\cdots,n\} \setminus (I \cup J),\ &X \mod 2 = 1 \Rightarrow X \subset M^2_{I,J}.
\end{aligned}
$$

We have constructed $\binom{n}{2}$ models and by definition none of the models are equivalent. □

We provide an example that reaches the lower bound. Notably, the lower bound is achieved by the hypercube configuration, as illustrated in Figure 28.

Proof. Let us consider $C = \{0,1\}^k$ the set of natural integers between 0 and 2^{k-1} written in base 2 with k digits. For instance, when $k = 2$, $C = \{00, 01, 10, 11\}$. By definition of C, $n = 2^k$.

Let us define k models $(M_0, \cdots, M_{k-1})$ such that for $i \in \{0,\cdots,k-1\}$ the partition M^1_k contains the numbers with a k^{th} digit equal to 0 and M^2_k the number with a k^{th} digit equal to 1. We consider a pair $I, J \in C$, by construction of C, I and J differ from one bit at least at position k. Thus $I \perp_{M_k} J$ and we set $M_{I,J} = M_k$.
Hence $\mathcal{F}(C) = k = log_2(n)$ □

5.2.2 Hyperplanes

The preceding definitions have provided a broad framework for comprehending the challenge of class separation, yet they are completely agnostic of the geometric aspects of the classes. To render these definitions more pragmatic and suited to real-world situations, we propose a refinement.

Let us consider a Euclidean space $E = \mathbb{R}^d$. We define n classes as independent variable $\mathbf{C}_i$ within this space. Each class follows its own probability density function p_i, and we introduce a scalar value $\varepsilon > 0$. We now narrow our focus to models consisting of affine hyperplanes with an error of ε (as formally defined in Definition 5.2.2 below).

Definition 24: Affine hyperplane

An affine hyperplane h is a affine subset of $\mathbb{R}^n$ described with a single equation: $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ where $\mathbf{w}$ is a nonzero vector, and b a scalar.
This hyperplane defines a model H of E with the associated bipartion $M^1 = \{x \in E | \langle \mathbf{w}, \mathbf{x} \rangle + b \geq 0\}$ and $M^2 = \{x \in E | \langle \mathbf{w}, \mathbf{x} \rangle + b < 0\}$.

In (Boland & Urrutia, 1995), the authors presented an elegant proof demonstrating that in $\mathbb{R}^d$, it is possible to separate n points using at most $\lceil (n - 2^{\lceil \log(d) \rceil})/d + \lceil \log(d) \rceil \rceil$ hyperplanes. This result provides an upper bound on $\mathcal{F}(C)$ in scenarios where the classes consist of individual points, employing the same definitions as previously outlined.

Extending these findings to radius 1 balls, is straightforward. However, when dealing with non-convex geometries, the interfaces between sets can exhibit intricate complexities, making it challenging to establish meaningful bounds on the requisite number of hyperplanes, as observed in (van den Berg, 2016).

In the context of random variables, it becomes necessary to introduce new definitions for separability. In real-world scenarios, hyperplanes may not achieve perfect separation of classes.

Definition 25: Epsilon-Separability

Let us consider two random variables $\mathbf{C}_i, \mathbf{C}_j$, and a hyperplane H. Without loss of generality, let us assume that $\mathrm{P}\left[\langle \mathbf{w}, \mathbf{C}_i \rangle + b \geq 0\right] \geq 0.5$. We define the probability error of the hyperplane H as:

$$\begin{aligned} \mathrm{P}_{err}(H, \mathbf{C_i}, \mathbf{C}_i) = \frac{1}{2} \big(& \mathrm{P}_i \left[\langle \mathbf{w}, \mathbf{C}_i \rangle + b < 0\right] \\ & + \mathrm{P}_j \left[\langle \mathbf{w}, \mathbf{C}_j \rangle + b \geq 0\right] \big) \end{aligned}$$

H separates $\mathbf{C_i}, \mathbf{C}_j$ if: $\mathrm{P}_{err} < \varepsilon$ and we note $\mathbf{C}_i \perp_H \mathbf{C}_j$.

Let us apply this definition to two real-valued random variables following a normal distribution with unit variance: $\mathbf{C_i} \sim \mathcal{N}(\mu_i, 1),\ C_j \sim \mathcal{N}(\mu_j, 1)$. Assuming $\mu_j > \mu_i$, the probability error of a hyperplane h is:

$$\begin{aligned} & \mathrm{P}_{err}(h, i, j) = \mathrm{P}_i \left[C_i < -b/w\right] + 1 - \mathrm{P}_j \left[C_j \leq -b/w\right] \\ & = \frac{1}{4} \left(2 + \left(erf\left(-\frac{b/w + \mu_2}{\sqrt{2}} \right) - erf\left(-\frac{b/w + \mu_1}{\sqrt{2}} \right) \right) \right). \end{aligned}$$

Note that the error is minimal when: $b/w + \mu_1 = b/w - \mu_2$ i.e $b/w = (\mu_1 + \mu_2)/2$ (Bishop & Nasrabadi, 2006).

Definition 26: Empirical Probability of Error

In practical scenarios, exact probability density functions are often unavailable, and instead, they are approximated empirically based on a given set of data

points. Let us consider a hyperplane h, and two classes i and j. Without loss of generality, we assume that samples $\mathbf{x}^i$ belonging to class i are correctly classified if $h(\mathbf{x}^i) > 0$, and conversely, samples $\mathbf{x}^j$ belonging to class j are correctly classified if $h(\mathbf{x}^i) \leq 0$. Let $\mathbf{X_i} = \{\mathbf{x}_1^i, \cdots, \mathbf{x}_n^i\}$ and $\mathbf{X_j} = \{\mathbf{x}_1^i, \cdots, \mathrm{x}_n^i\}$, represent sets of n samples drawn of classes i and j respectively. The approximated probability error is defined as follows:

$$\hat{\mathrm{P}}_{err}(h, i, j) = \frac{1}{2n} \sum_{k=1}^{n} \left(\mathbf{1}_{h(x_k^i)<0} + \mathbf{1}_{h(x_k^j)\geq 0} \right)$$

5.3 Methodology

When dealing with a set of classes denoted as C, it is possible for a model to learn to distinguish more classes than those explicitly present in C. In this study, we examine the selection of subsets $C' \subset C$ where the model can learn to separate the largest number of new classes. To accomplish this, we leverage the theoretical framework previously introduced. In this section, we provide a detailed description of our methodology for identifying such subsets.

5.3.1 Settings

Let us introduce a scalar parameter $\varepsilon > 0$, a set of classes denoted as C, a set P encompassing all possible pairs of classes in C, and $\mathbf{X}$ representing a set of examples drawn from these classes. As outlined in the theoretical section, we learn a model for each pair of classes. In our experiments, we adopt linear hyperplanes, which can be typically the final layer of a neural network. However, instead of inferring these hyperplanes directly on the input data domain, we determine them based on features extracted from pretrained networks. This approach is favored as pretrained networks are more likely to capture high-level information.

In our experiments, we utilize two widely used pretrained models: a ResNet-50 (He et al., 2016) and a Vision Transformer DINO VIT-8 (Caron et al., 2021), both pretrained on the IMAGENET dataset (J. Deng et al., 2009). Vision Transformer models have gained large recognition for their proficiency in learning generic features through self-supervised learning, while ResNet-50 has served as a standard benchmark in transfer learning (Krizhevsky et al., 2017). Furthermore, works such as (Z. Liu et al., 2022; Wightman et al., 2021) have shown that residual networks remain competitive despite the prevalent use of transformers.

In line with common practice in classification tasks, we divide the network into two components: the feature extractor f and the heads h. The heads, denoted as $h_{I,J}$ for each pair $\{I, J\}$ according to Definition 9, play a key role in our framework. A pair $\{K, L\}$ is considered separated by a head $h_{I,J}$, if the empirical probability error of the head $h_{I,J}$ is less than ε. We refer to the sum of distinct pairs that the models separate as ***separability***.

5.3.2 Estimating the generalization potential of a subset

Initially, we employ a feature extractor to represent the data, and subsequently, we train one head per pair based on this representation. To evaluate the potential of a subset $C' \subset C$, we exclusively consider the heads that correspond to the pairs in P'. Here,

P' is the set of pairs derived from C'. Subsets that attain the highest separabilities are considered the most promising candidates.

Once such subsets are identified, we investigate two distinct approaches: fine-tuning and training from scratch. In the case of fine-tuning, we fine-tune the pretrained networks on the selected subsets of classes. The separabilities serve as relevant performance indicators in this scenario since the same network is employed. However, when training from scratch, the relationship between separabilities and performance becomes more uncertain, and it is challenging to predict whether separabilities will reliably indicate performance.

5.3.3 Training procedure

Upon selecting a subset C', we initiate training process on the pairs within C'. As previously explained, we employ one head per pair, assigning binary labels to each pair, where one class is labeled as 0 and the other as 1. For loss computation, we utilize binary cross-entropy for each head.

It's important to note that the fine-tuning process comprises two steps. In the first step, the feature space is frozen, and only the heads are trained. This step has already been completed during the assessment of subset potential. Consequently, instead of repeating this step, we retain the heads corresponding to the selected pairs. In the second fine-tuning step, the entire network is fine-tuned, including the feature extractor.

5.3.4 Evaluation

We evaluate the separability of the model across the entire set of pairs P of C. To determine whether pairs are separated, we follow a specific procedure. Consider a pair $\{I, J\}$. For one of the classes in this pair, say I, we calculate the average sign of the predictions provided by a head. Subsequently, we count the number of examples from the class I where the sign of the prediction aligns with the average sign. For the other class J, we count the number of examples where the sign of the prediction is opposite to the average sign. Finally, we compute the ratio of correctly predicted examples to the total number of examples and iterate over all the heads. If the error is less ε for at least one of the heads, we consider the pair to be separated.

5.4 Experiments

We conduct experiments using classification datasets, and for each dataset, we refer to its set of classes as C. Our primary goal is to investigate the identification of the most promising subsets of classes, denoted as C', that exhibit the highest potential for generalization in terms of transferability. It is important to note that C' can either be a subset of C or completely disjoint, like it is the case in a few-shot setting. The process of determining these subsets relies on the methodology outlined earlier.

We assess our approach in three distinct settings: Fine-tuning, where we employ a pretrained architecture and fine-tune it on specific subsets of classes; Training from scratch, where we train a new architecture exclusively on the subsets; and Few-Shot Learning, where we train an architecture solely on base classes and evaluate its ability to distinguish novel classes (additional explanations are provided later).

5.4.1 Finetuning

In this experiment, we employ pretrained architectures to identify promising subsets of classes using our methodology. We focus our experiments on the CIFAR10 dataset as it serves as a standard benchmark in computer vision and comprises $N = 10$ classes, allowing us to explore various subset sizes. Dealing with datasets with a larger number of classes would make it practically infeasible to compute all potential subsets. Moreover, conducting experiments on very large datasets, where a single training may take days, would make it impossible to compute enough runs for rigorous statistical analysis.

We generate all possible subsets composed of $n \in \{2, 4, 6, 8\}$ classes. We refer to CIFAR_n when the dataset is restricted to n classes. As described in the methodology section, for each subset of n classes, we compute the number of pairs of classes separated before and after finetuning. For $n = 4$ or 6, there are 210 possible subsets. Separability of these runs before and after finetuning are shown in Figure 31 and 32.

In Table 9, we provide the correlation between the two metrics, as well as the slope and intercept of the regression line for both VIT-8 and ResNet50. It is important to note that in both cases, the correlation is high, indicating that separability before finetuning is a reliable indicator of finetuning performance.

Additionally, we carry out an analysis of the classes having the most impact on finetuning performance. For this matter, we calculate a metric that we call class separability $sep(I)$. This metric is computed as follows. Let us consider a class I. We take all the runs where the class I appears in the selected subset of classes. For each run, we utilize the corresponding model and associated heads –these are all the heads h assigned to pairs that include class I. We then consider the smallest error given by the heads for each pair and check if it is smaller than ε.

For instance, let K, L be a pair we are considering. The separability $\text{sep}_{K,L}(I)$ is determined as follows:

$$\text{sep}_{K,L}(I) = \begin{cases} 0 & \text{if } \min_{j\in\{1,\cdots,n\}} \hat{\mathbf{P}}_{err}(h_{I,K}, K, L) > \varepsilon \\ 1 & \text{otherwise.} \end{cases}$$

Finally, we sum the separabilites of all pairs so that: $\text{sep}(I) = \sum_{I=1}^{N} \sum_{J=I+1}^{N} \text{sep}_{K,L}(I)$.

The values of this metric are depicted in Figure 29 and 30, and we report the correlations in 10. Notably, we observe a high correlation between class separability before and after finetuning on CIFAR10. With the ResNet50 model, the correlation is 0.90 on CIFAR_4 and 0.86 on CIFAR_6, while with the VIT-8 model, it is 0.96 and 0.95, respectively.

Therefore, we can confidently identify the classes that are likely to yield the best finetuning performance. For $n = 6$, these classes are bird, frog, and horse for Resnet50 and truck, dog and frog for VIT-8. Notably, these classes are included in the subset of classes that consistently lead to the best fine-tuning performance within the 210 computed runs: ("bird," "cat," "deer," "dog," "horse," "ship") for Resnet50. Animal classes appear to be particularly conducive to achieving the best generalization, although the optimal subsets consistently include at least one non-animal class.

We highlight that subtle distinctions exist between the graphical representations produced by VIT-8 and Resnet50, as shown in Figure 29 and 30. These distinctions can be attributed to the dissimilarities in their training procedures. Typically, VIT-8 is a recent architecture trained with a contrastive loss function, coupled with various data augmentation techniques and contemporary regularization methods.

Furthermore, we have undertaken the computation of analogous metrics by considering pairs rather than classes. Let us consider a pair P, we proceed as follows to compute the metric. We consider all the runs where the pair P is included in the selected pairs of classes. For each run, we take the corresponding model and the associated head h assigned to the pair P. We then count the pairs that can be separated to obtain sep(P).

Once again, a noteworthy observation emerges concerning CIFAR_6. Detailed correlations can be found in Table 10. Interestingly, within both experimental settings, the pair that exhibits the lowest values of separability is "dog vs. cat." This observation implies that the learned hyperplane may excessively specialize in distinguishing this particular pair. Conversely, the "airplane/bird" pair proves to be the most effective.

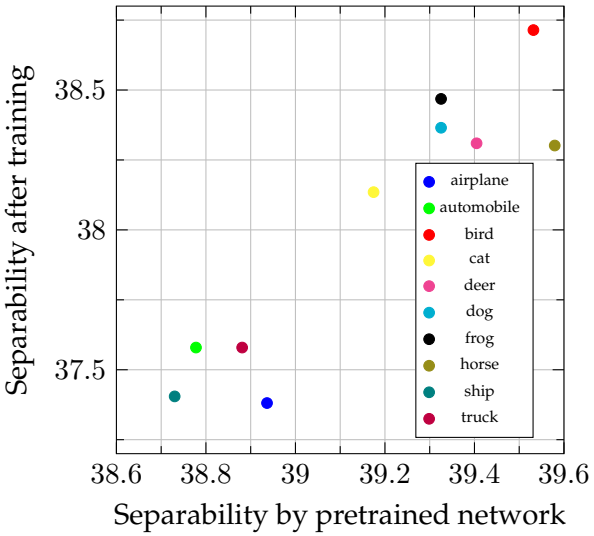
We conduct a similar experiment on the fine-grained CUB200 dataset, which consists of 200 distinct bird classes (accessible at: https://www.vision.caltech.edu/datasets/cub_200_2011/). Our approach includes finetuning the networks on subsets of 60 classes; we generate 300 distinct subsets. In Table 9, we report the correlation and slope coefficients of the regression analysis between the separability prior to and following the fine-tuning process for both the VIT-8 and Resnet50 networks on the CUB200 dataset. gain correlation are very high indicating that the separability before finetuning is relevant indicator of performances.

Table 9 – Correlation and Slope Coefficient of the regression between the separability before and after finetuning for the pretrained Resnet50.

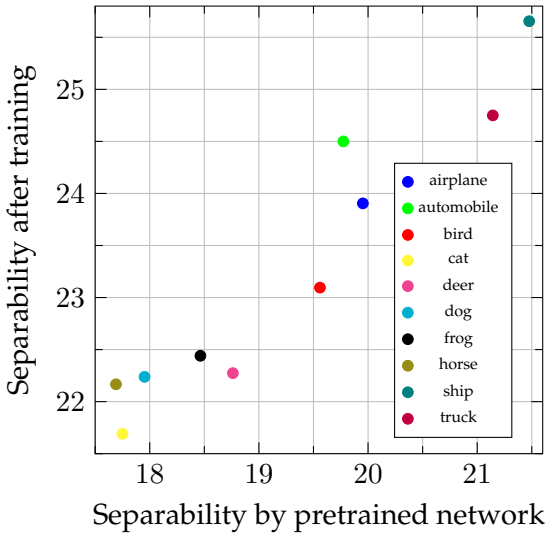
Network	Dataset	Corr.	Slope	Intercept
Resnet-50	CIFAR10_4	0.76	0.73	0.89
	CIFAR10_6	0.79	0.89	2.8
	CUB200_60	0.71	0.63	7766
VIT-8	CIFAR10_4	0.97	0.88	3.88
	CIFAR10_6	0.95	0.86	4.2
	CUB200_60	0.88	0.60	8090

Table 10 – Correlation between multiple metrics given by a pretrained network and separability when training from scratch.

Network	Metric	CIFAR10_4	CIFAR10_6
Resnet-50	Class Separability	0.95	0.92
	Pair separability	0.92	0.89
VIT-8	Class Separability	0.99	0.98
	Pair separability	0.98	0.98

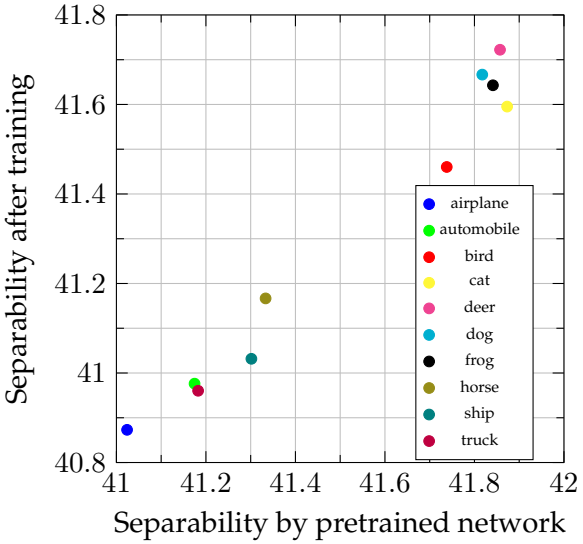


(a) Finetuning on 6 classes of CIFAR10. The analysis highlights the most promising classes, such as bird, frog, dog and horse.

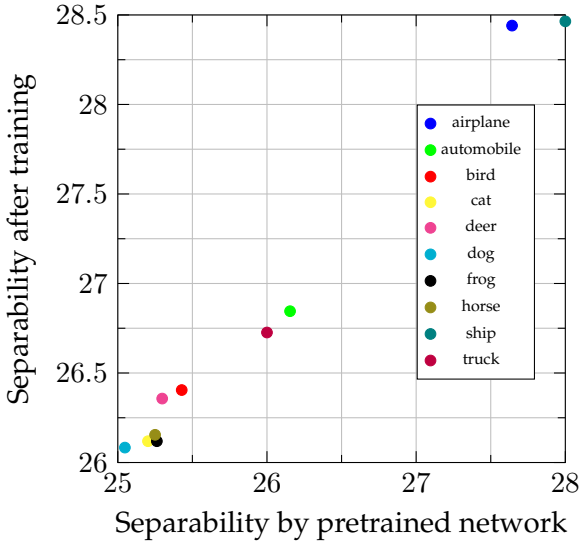


(b) Finetuning on 4 classes of CIFAR10. The analysis highlights the most promising classes, such as bird, frog, dog and horse.

Figure 29 – Separability given by a **Resnet-50** before and after **finetuning** on subsets of CIFAR10. The x-axis represents the results from the Pretrained **Resnet-50**, while the y-axis corresponds to the same network finetuned on subsets classes of CIFAR10.

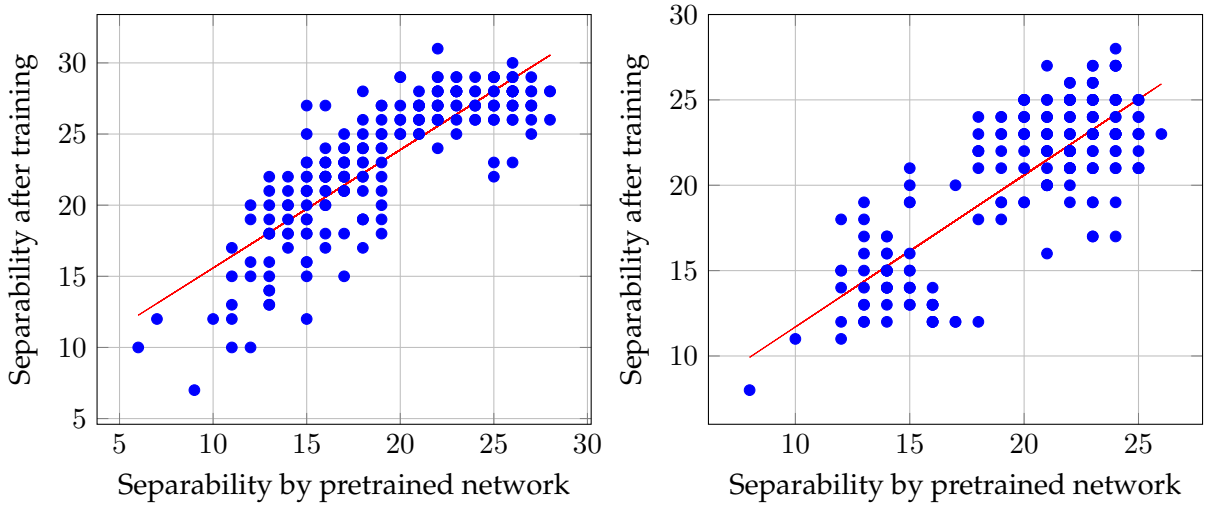


(a) Finetuning on 6 classes of CIFAR10. The analysis highlights the most promising classes, such as truck, dog, frog.



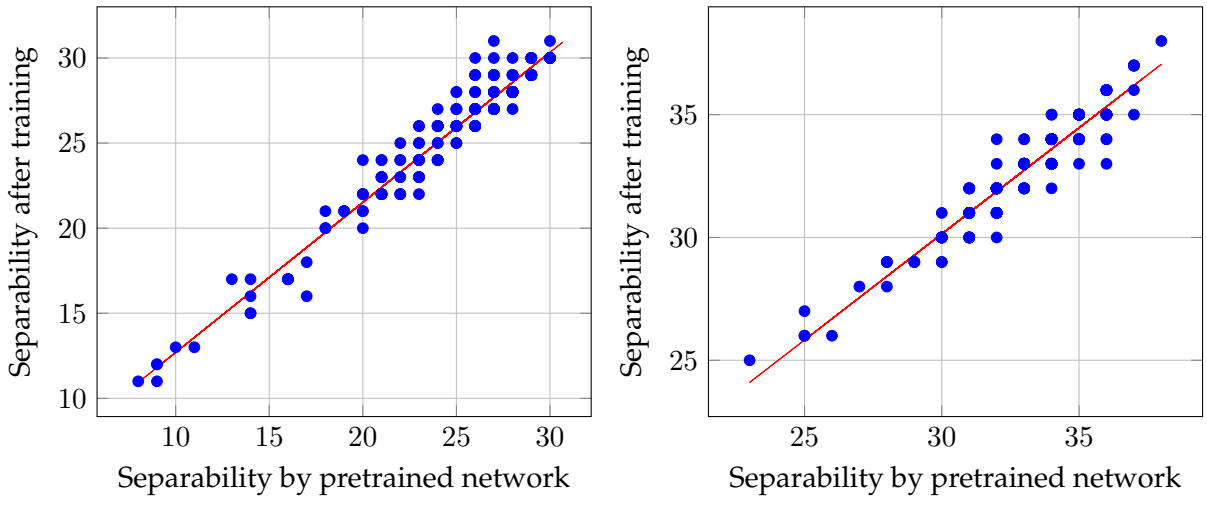
(b) Finetuning on 4 classes of CIFAR10. The analysis highlights the most promising classes, such as ship, airplane, automobile.

Figure 30 – Separability given by a **VIT-8** before and after **finetuning** on subsets of CIFAR10. The x-axis represents the results from the Pretrained VIT-8, while the y-axis corresponds to the same network finetuned on subsets of classes of CIFAR10.



(a) Subsets of 4 classes of CIFAR10. **(b)** Subsets of 6 classes of CIFAR10.

Figure 31 – Resnet50: Number of Pairs Separated before vs after **finetuning** on subsets of CIFAR10.



(a) Subsets of 4 classes of CIFAR10. **(b)** Subsets of 6 classes of CIFAR10.

Figure 32 – VIT8: Number of Pairs Separated before vs after **finetuning** on subsets of CIFAR10.

5.4.2 Training From Scratch

We delve into the possibility of identifying the subset that yields optimal generalization in a transfer learning scenario when training a network from scratch.

We opt to utilize a Resnet18 (He et al., 2016), a standard and competitive architecture within the field of Computer Vision (Wightman et al., 2021). On CIFAR10, we conduct training on subsets comprising both $n = 4$ and $n = 6$ classes. Once again, we leverage either the pre-trained Resnet50 or VIT-8 to derive a data representation, which subsequently facilitated the assessment of separability within the subsets prior to training. It is crucial to note that, unlike fine-tuning, the separability indicators furnished by the pre-trained networks may not necessarily align with the post-training separability. This discrepancy arises from variations in dataset used for training, architectural differences, and disparate training procedures.

Nevertheless, as detailed in Table 11, the pre-trained networks still serve as relevant performance indicators, exhibiting robust correlation coefficients, especially concerning class separability. We depict as well the separability of each run on Figure 37 and 38.

Similar to our approach during fine-tuning, we have graphically depicted the mean separability for each class in Figure 29 and 30, offering valuable insights for the identification of the most promising classes. Intriguingly, in comparison to the fine-tuning experiments, the optimal subsets reveal only marginal disparities. For instance, with $n = 6$, the optimal subset is (automobile/bird/cat/deer/horse/truck) for Resnet-50. As illustrated in Figure 29, most of these classes exhibit substantial potential.

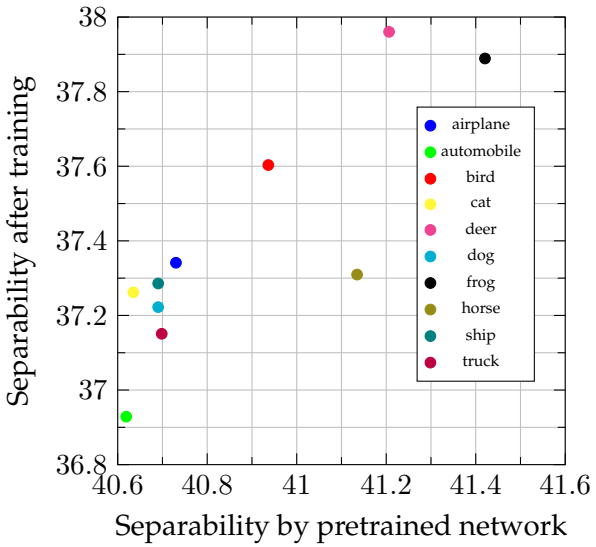
We have also conducted a similar experiment on FASHION-MNIST (Xiao et al., 2017) and MNIST (L. Deng, 2012), encompassing 10 distinct classes. For FASHION-MNIST, the "Dress" class emerges as the most promising, which aligns logically with its dissimilarity from the other clothing categories. It is noteworthy that, across both datasets, none of the models produced by this optimal subset of classes are equivalent when visualized via the Hasse Diagram.

Table 11 – Correlation between multiple metrics given by a pretrained network and separability when training from scratch.

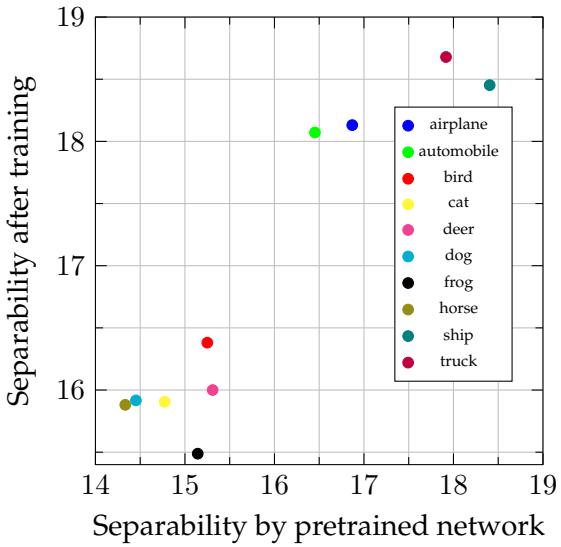
	Metric	CIFAR10_4	CIFAR10_6	FASHION	MNIST
Resnet-50	Separability	0.68	0.47	0.86	0.51
	Class Separability	0.93	0.76	0.92	0.67
	Pair separability	0.90	0.74	0.95	0.67
VIT-8	Separability	0.78	0.44	0.88	0.58
	Class Separability	0.94	0.97	0.96	0.73
	Pair separability	0.92	0.95	0.94	0.68

5.4.3 Few-Shot Learning

We carry out experiments in few-shot learning, a common scenario for assessing a model's ability to adapt to new classes. Specifically, our focus lies on inductive few-shot learning, where predictions hinge solely upon a limited set of labeled examples. To assess performance in transfer, we employ a nearest mean classifier (NCM) approach, a well-established method in the realm of inductive few-shot learning (Y. Wang et al., 2019).

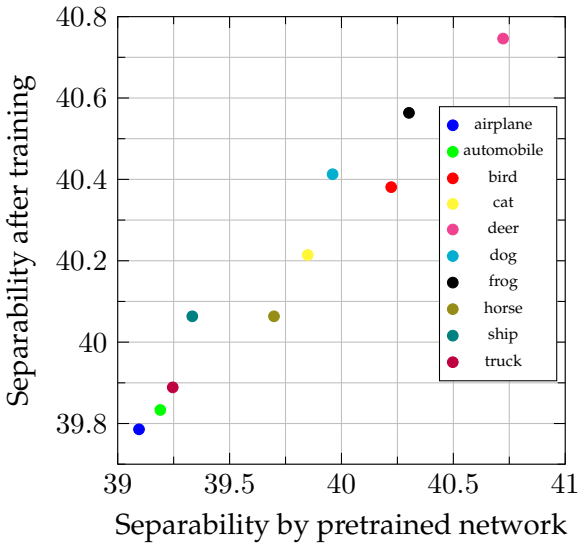


(a) Training from scratch on 6 classes of CIFAR10. The analysis highlights the most promising classes, such as deer, frog, horse, and bird.

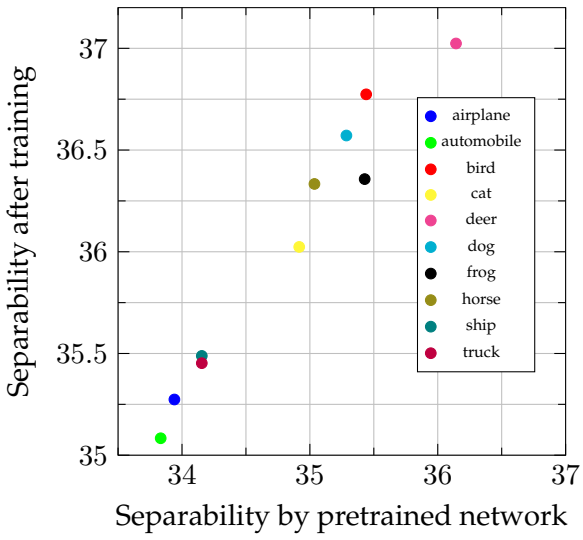


(b) Training from scratch on 4 classes of CIFAR10. The analysis highlights the most promising classes, such as truck, ship, airplane, and automobile.

Figure 33 – Training from scratch on subsets of classes of CIFAR10. The x-axis represents the results from the **Pretrained Resnet-50,** while the y-axis corresponds to the Resnet18 trained from scratch on subsets of classes of CIFAR10.

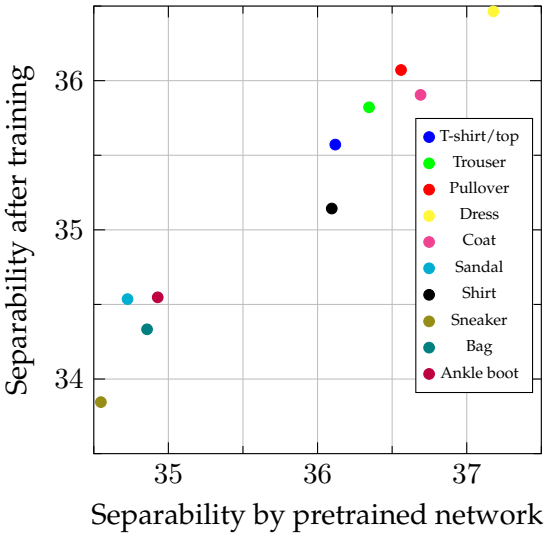


(a) Trained from scratch on 6 classes of CIFAR10. The analysis highlights the most promising classes, such as deer, bird, dog and frog.

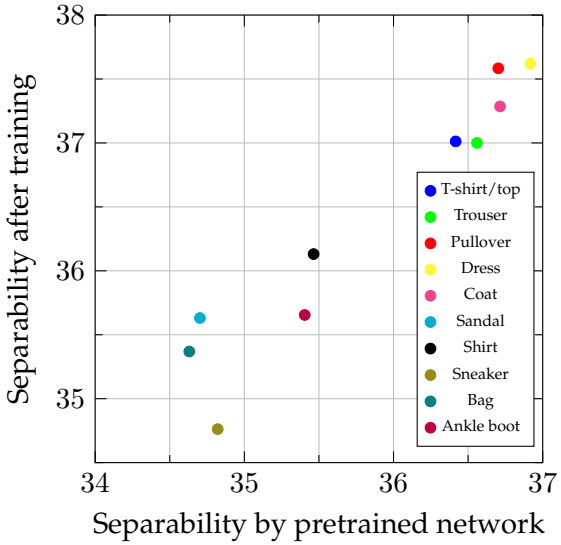


(b) Trained from scratch on 4 classes of CIFAR10. The analysis highlights the most promising classes, such as deer, frog, horse, and bird.

Figure 34 – Training from scratch on subsets of classes of CIFAR10. The x-axis represents the results from the **Pretrained VIT-8**, while the y-axis corresponds to the Resnet18 trained from scratch on subsets of classes of CIFAR10.

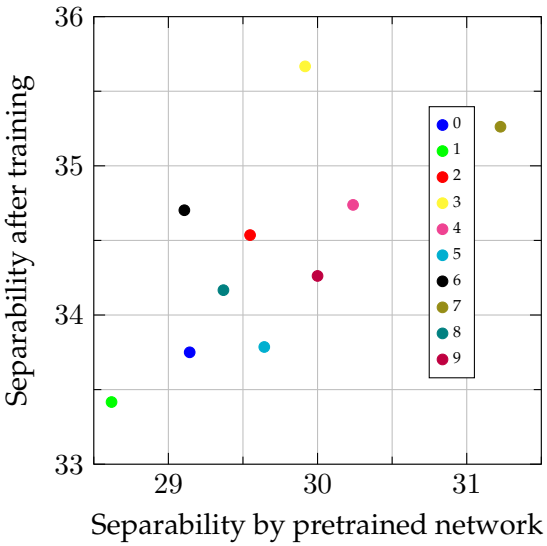


(a) Trained from scratch on 4 classes of FASHION-MNIST. The analysis highlights the most promising classes, such as Dress, Pullover, Coat.

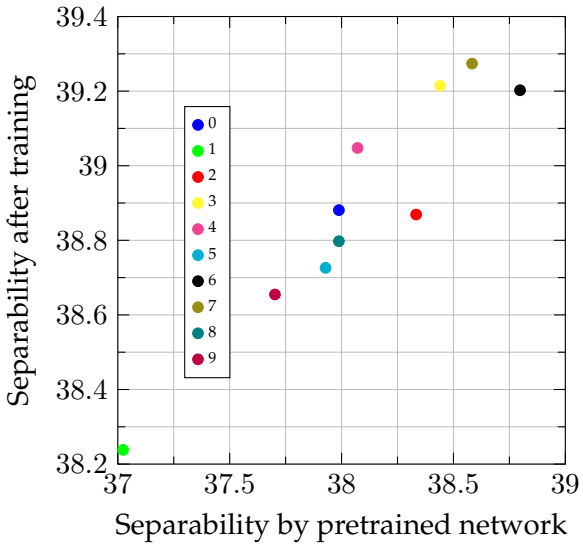


(b) Trained from scratch on 4 classes of FASHION-MNIST. The analysis highlights the most promising classes, such as Dress, Pullover, Coat, Trouser and T-shirt.

Figure 35 – Training from scratch on subsets of classes of **FASHION-MNIST**. The x-axis represents the results from the Pretrained **Resnet-50,** while the y-axis corresponds to the Resnet18 trained from scratch on subsets of classes of CIFAR10.

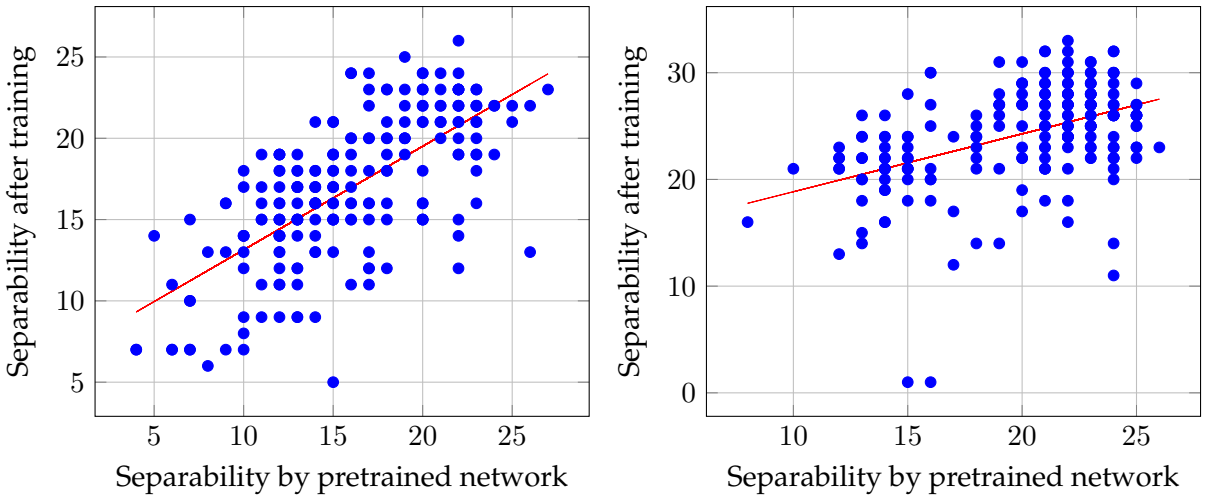


(a) Trained from scratch on 4 classes of MNIST with a **Resnet-50**. The analysis highlights the most promising classes, such as 7 and 4



(b) Training from scratch on 4 classes of MNIST. The analysis highlights the most promising classes, such 6, 7 and 3

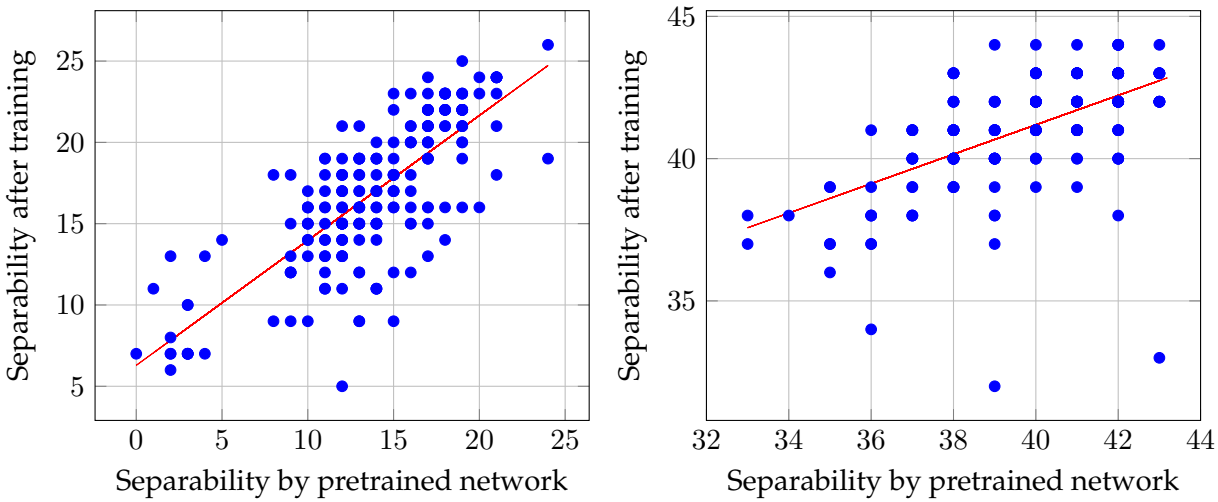
Figure 36 – Training from scratch on subsets of classes of **MNIST**. The x-axis represents the results from the Pretrained Resnet-50, while the y-axis corresponds to the Resnet18 trained from scratch on subsets of classes of CIFAR10.



(a) Separability by pretrained network vs after training from scratch on subsets of 4 classes of CIFAR10.

(b) Separability by pretrained network vs after training from scratch on subsets of 6 classes of CIFAR10.

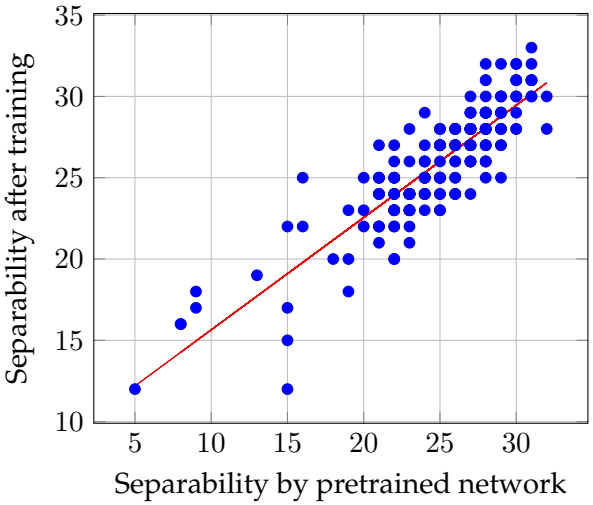
Figure 37 – Training from Scratch on subsets of CIFAR10, **Resnet-50** employed to identify promising subsets.



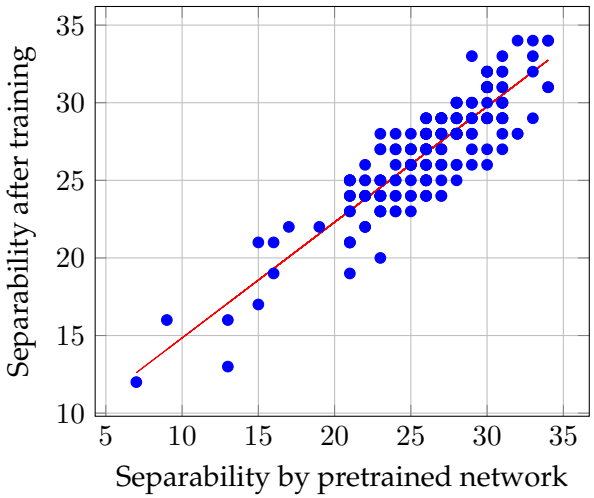
(a) Separability by pretrained network vs after training from scratch on subsets of 4 classes of CIFAR10. X-axis corresponds to the number of pairs separated by the Pretrained VIT-8.

(b) Separability by pretrained network vs after training from scratch on subsets of 6 classes of CIFAR10. X-axis corresponds to the number of pairs separated by the Pretrained VIT-8.

Figure 38 – Training from Scratch on subsets of CIFAR10, **VIT-8** employed to identify promising subsets.

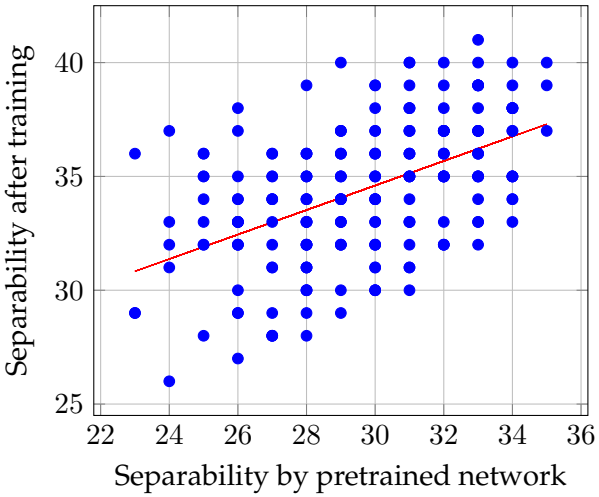


(a) Separability by pretrained network vs after training from scratch. X-axis corresponds to the number of pairs separated by the **Pretrained Resnet50**.

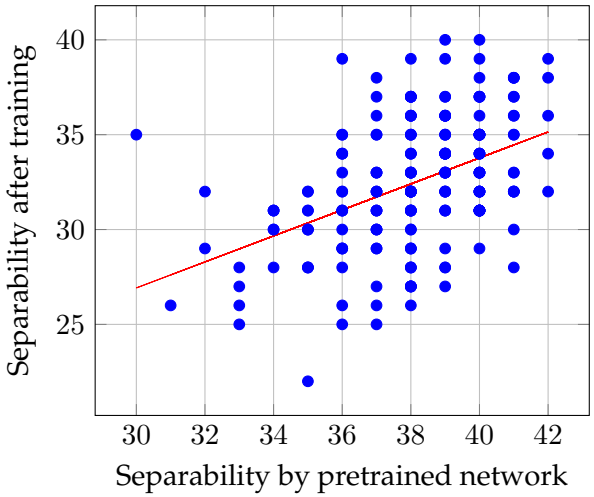


(b) Separability by pretrained network vs after training from scratch. X-axis corresponds to the number of pairs separated by the **Pretrained VIT-8**.

Figure 39 – Training from Scratch on subsets composed of 4 classes of **FASHION-MNIST**.



(a) Separability by pretrained network vs after training from scratch. X-axis corresponds to the number of pairs separated by the **Pretrained Resnet50**.



(b) Separability by pretrained network vs after training from scratch. X-axis corresponds to the number of pairs separated by the **Pretrained VIT-8**.

Figure 40 – Training from Scratch on subsets composed of 4 classes of **MNIST**.

Aligned with our methodology, we evaluate the separability of **novel classes** by employing representations derived from a pre-trained network, namely Resnet-50. However, in this instance, our separability assessment is derived **only on models inferred from pairs of the base classes**, as few-shot learning exclusively trains on these classes. For each dataset, we identify potential sets of pairs that exhibit the best and worst separability.

Once again, it remains uncertain whether the separability offered by the pre-trained network serves as a reliable predictor for few-shot learning. The networks are trained exclusively on base classes, and evaluation differs from standard classification.

To conduct our experiments, we use the networks provided by (Bendou et al., 2022). Our experiments are carried out on standard few-shot learning datasets, that includes MINIIMAGENET (Vinyals et al., 2016), FC100 (Oreshkin et al., 2018), TIEREDIMAGENET (M. Ren et al., 2018) and CIFAR-FS (Y. Liu et al., n.d.), all of which are widely acknowledged benchmarks in the field. We evaluate performance using the Nearest Class Mean (NCM) approach on both the best and worst class pair sets under the 1-shot learning scenario, where each novel class has only a single labeled example. Additionally, we establish a baseline by reporting performance across all class pairs. To ensure robust results, we conduct 10,000 runs for each scenario, calculating average accuracy and a 95% confidence interval across these runs.

As reported in Table 12, the worst sets consistently produce lower performance compared to both the baseline and the best sets, in most datasets, with the exception of TIEREDIMAGENET. Notably, the best sets consistently outperform the others across all datasets. Specifically, on MINIIMAGENET, the most challenging pairs include crab vs ants, lion vs bus, and Siberian Husky vs Dalmatian. Visual examples of these challenging pairs are provided in 41. A closer examination of these visual examples reveals that “crab" and “ants" share similar colors, textures, and structural patterns. Similarly, “lion" and “bus" are depicted against a comparable grassy background. Additionally, Dalmatian and Husky both belong to the category of dog breeds and exhibit similar color patterns, making them harder to distinguish.

Table 12 – Accuracy averaged over 10000 runs in Inductive Few Shot Learning (2-Ways, 1-Shot 15-Queries).

Dataset	Best	Worst	Base
MINI-IMAGENET	0.94 ± 0.12	0.86 ± 0.12	0.87 ± 0.11
CIFARFS	0.87 ± 0.11	0.83 ± 0.15	0.85 ± 0.14
FC100	0.74 ± 0.1	0.68 ± 0.14	0.70 ± 0.15
TIERED-IMAGENET	0.95 ± 0.07	0.94 ± 0.07	0.95 ± 0.07

(a) Crab

(b) Ants

(c) Lion

(d) Bus

(e) Husky

(f) Dalmatian

Figure 41 – Depiction of the most challenging pairs on TIEREDIMAGENET: Crab vs Ants; Lion vs Bus; Husky vs Dalmatian.

5.5 Limitations and Perspectives

5.5.1 Limitations

Our proposed framework comes with limitations. We focus on models based on pairs, and in practice, we potentially overlook interactions that may emerge when learning is done on subsets of multiple pairs. However, it is worth noting that our theoretical framework could be easily extended to include subsets of several pairs, instead of singleton of pairs, offering a potential solution to overcome this limitation. Furthermore, our assertion about identifying promising subsets during training from scratch might be overly ambitious and constrained, given that the effectiveness of subset identification relies on pretrained networks.

5.5.2 Perspectives

We believe that further research can build upon the presented framework to investigate several intriguing questions and explore various aspects of transfer learning. In Wightman et al., 2021, the authors highlight the need to adapt training to the considered deep architecture, but finding the optimal training process remains a complex task. Our framework could be leveraged to examine the impact of different training procedures on the inferred representations and transferability.

By systematically varying the training settings, such as data augmentation techniques, loss functions, and optimization algorithms, researchers can gain deeper insights into how these factors affect the generalization capabilities of pretrained models. Understanding the nuances of training procedures could lead to the development of more robust and transferable models. Moreover, the framework's potential extends to investigating the transferability of representations across different domains and tasks. Researchers can explore how pretrained models generalize across different datasets, domains.

Additionally, the formalism introduced in this work opens up possibilities for constructing challenging few-shot learning datasets. By carefully designing subsets of classes with specific characteristics, such as similarity, diversity, or hierarchical relationships, researchers can create benchmarks that thoroughly assess the generalization capabilities of few-shot learning algorithms. These datasets can aid in evaluating the effectiveness of different few-shot learning methods and shed light on the strategies that work best.

5.6 Conclusion

In this chapter, we addressed the concept of transfer learning in classification tasks by introducing a novel theoretical model. We provide an order relationship of learning, allowing us to precisely define transferability between different sets of classes. By conducting experiments on multiple datasets, we demonstrated that the separability achieved by classes before fine-tuning is a strong indicator of the transferability potential during the fine-tuning process. This finding opens up new possibilities for predicting model performance and optimizing transfer learning practices.

Additionally, we explored training from scratch and few-shot learning scenarios, finding that the separability provided by pretrained networks remains relevant for assessing future performance. Overall, this work contributes to a deeper understanding of

transfer learning mechanisms and paves the way for enhanced model generalization and transferability in various real-world applications.

The proposed framework opens new research possibilities. It enables the study of the impact of different training procedures on representations and transferability. Moreover, it helps creating challenging few-shot learning datasets and exploring transferability across domains. This framework paves the way for advancing transfer learning research.

Chapter 6

General Conclusion

Contents

6.1 Conclusion

Originally, generalization in machine learning mainly focused on how well models performed on new data that was closely related to their training tasks. However, the advent of deep learning has brought about significant changes, revealing that deep learning architectures can actually generalize beyond their original training tasks. In this manuscript, we have delved into various situations where this extended generalization ability becomes apparent and proves to be advantageous.

To begin, let us provide a concise summary of each contribution and outline how they could be further enhanced with additional dedicated time. Subsequently, we will contextualize this manuscript within a broader research framework.

6.1.1 Task-Related Invariant Operators

We have seen that deep learning architectures can provide versatile data representations, but their ability to generalize well, whether on new data or tasks, depends largely on how the architecture is designed. The layers or operators used in building the architecture can make it either invariant or equivariant to geometric transformations. For instance, deep convolutional neural networks achieve invariance to Euclidean translations, which can be harnessed to create highly effective feature representations for natural images.

But, determining the specific invariances or equivariances that should be achieved, especially in the case of irregular domains like graphs, can be a challenging task. Nonetheless, our research has demonstrated the feasibility of training models to learn these layers as invariant operators tailored to a specific training task. In doing so, we have successfully uncovered standard operations such as dilation, compression, and translation in the context of natural images. Importantly, our findings extend beyond regular domains, showing that these operators can also be learned effectively for irregular and abstract graph structures.

However, with additional time and further research efforts, our work stands to gain valuable insights from complementary investigations. Several noteworthy directions for future exploration include:

- **Fixing learned operators**: Building upon (d'Ascoli et al., 2021), an intriguing avenue of research involves investigating whether improved generalization can be achieved by initially training the operators and subsequently fixing them to redefine the architecture's layers.
- **Investigating others tasks and losses**: While our current work primarily focuses on classification tasks using cross-entropy, it would be enlightening to explore alternative criteria, such as unsupervised and self-supervised losses. Analyzing the potential differences between the inferred operators and discussing their implications. Similarly, assessing how data augmentation influences the learned operator represents an intriguing area to understand their impact over the training process.
- **Invariances in Transfer Learning**: In the context of transfer learning, it would be particularly insightful to study which invariances prove advantageous or detrimental. Some of these invariances may depend on the specific target tasks. For example, invariance to colors may be beneficial in digit classification but suboptimal for a birds dataset, where variations in colors is leveraged to distinguish species.

- **Graphs and Transformers**: Our methodology employs graph to embed data geometry, for instance treating an image as a grid graph. Considering the architecture of transformers (Dosovitskiy et al., 2020) and their self-attention layers, it reveals a connection between graph neural networks and transformers. Indeed, self-attention layers, originally designed for sequences, can be adapted to images by viewing an image as a patchwork of smaller images. Each element in an image (e.g., pixel) possesses spatial relationships with its neighbors, similar to nodes in a graph connected to neighboring nodes. This relationship opens doors to further exploration into how attention mechanisms can learn equivariant or invariant operators (d'Ascoli et al., 2021), contributing to enhanced generalization capabilities.

6.1.2 Artificial Tasks for Improved Generalization

In the context of deep learning, the challenge of overfitting has led to the development of various regularization techniques. Interestingly, some of these techniques can be viewed as methods for artificially augmenting the complexity of the training task. For instance, Mixup achieves this augmentation by generating virtual samples through linear interpolation. Unfortunately, in certain cases, these virtual samples may deviate from the underlying data geometry.

To address this issue, we proposed an improvement to Mixup by incorporating locality-based constraints. This adjustment led to enhanced generalization while mitigating the regularization impact of Mixup, and we denoted this approach as Local Mixup. The implementation of these constraints involves the use of a weight matrix that allocates weights to interpolations based on the distances between the data points engaged in the interpolation. Interestingly, this weight matrix corresponds directly to the adjacency matrix of a graph. We conducted an exploration of several techniques for constructing this graph, encompassing threshold graphs, K-nearest neighbor (KNN) graphs, and decreasing exponential graphs.

Our theoretical analysis revealed that in low-dimensional spaces, Mixup exerted an averaging effect that impacted the model's bias and variance. In high-dimensional spaces, Mixup influenced the Lipschitz constant of the model. Experimentally, we demonstrated that Local Mixup outperformed Mixup in performance on well established datasets such as SVHN, CIFAR-10, and FASHIONMNIST.

However, there remains ample opportunity for further investigation and enhancement of our work. Some notable directions for future exploration include:

- **Extending the Theoretical Framework to Higher Dimensions**: We could examine cases where a function f^* achieves a zero loss on the training data, while considering neural networks composed of two hidden layers. This analysis could shed light on properties of such functions, such as how Mixup encourages a form of convexity, as the Mixup criterion expects that $f(\lambda x_1 + (1-\lambda)x_2) = \lambda f(x_1) + (1-\lambda)f(x_2)$. This exploration could also encompass convex interpolation on simplicial complexes.
- **Improving Mixup with geometry**: It would be intriguing to investigate the implementation to explore the use of Local Mixup in a latent space, as opposed to the direct space, where the geometry is more likely to be Euclidean. One could also use more relevant distance metrics and interpolation functions to generate new samples, including the possibility of using a generative neural network like a normalizing flow for interpolation in the latent space and subsequent reconstruction of natural images.

- **Further Exploring the Regularization Effect of Mixup**: An engaging avenue of research involves a comprehensive examination of the interpretation of Mixup as a method for generating anisotropic noise along a specific direction determined by the interpolation coefficient. Additionally, one could investigate Mixup as a means to rebalance the gradient contributions of samples during training, especially those that tend to be disregarded as the training progresses. In particular, Mixup can potentially reintroduce importance to these samples by interpolating them with more challenging examples.

These unexplored avenues hold the promise of advancing our understanding of regularization techniques like Mixup and their impact on deep learning generalization.

6.1.3 Generalization Beyond the Training Task: Coarse to Refined Labels

Deep learning architectures have showcased their remarkable ability to extend their generalization capabilities beyond their original training tasks, particularly in the domain of transfer learning (Krizhevsky et al., 2017; Radford et al., 2015; Touvron et al., 2021; Zhuang et al., 2020a). This achievement is often attained through training on large datasets using self-supervised losses. In our investigation, we sought to determine whether it is possible to retrieve information from refined labels when training on approximated, coarse, labels. Intriguingly, we identified two distinct phases in the learning process, akin to what was observed by a prior study Shwartz-Ziv and Tishby, 2017. These phases can not be observed when solely considering the accuracy on the training task but can be seen when monitoring the entropy of the feature space, as we have demonstrated, or by assessing the mutual information between the input and the feature space.

Moreover, we demonstrated that it is feasible to apply an entropic regularization in the feature space to recover information related to the refined labels. This approach also yields improvements in the model's transferability in scenarios like Few Shot Learning. We see the following avenues for further exploration:

- **Hierarchical Datasets**: Investigate our approach using hierarchical datasets such as CIFAR100 or INaturalist. Additionally, examine existing metaclasses on Imagenet and assess the emergence of hierarchical structures and subclasses. Evaluate whether these structures align with the dataset's inherent hierarchy or differ from it. It is worth noting that hierarchical datasets may not necessarily conform to the latent space's underlying geometry. For example, two birds from different species may be classified by an expert as belonging to the same genus, while latent spaces may indicate alternative metaclasses. Many researchers are exploring the application of hyperbolic networks to solve hierarchical datasets (Peng et al., 2021). Specifically, they are leveraging hyperbolic geometry's unique properties to better represent complex hierarchical relationships.
- **Evolution of Feature Space Geometry**: Most self-supervised learning research typically focuses on achieving high accuracy on the test set. However, as indicated, this metric often inadequately reflects the model's transferability. Our study demonstrated that entropy can reveal distinct phases during learning that are not apparent when monitoring accuracy alone, potentially linked to collapses in the feature space. Nonetheless, this metric provides a global perspective and may not capture variations between individual classes. To comprehensively analyze the feature space's geometry, consider leveraging tools such as graphs and derived metrics. This approach could lead to the development of new geometric criteria for evaluating deep learning models.

6.1.4 A Theoretical Framework for Generalization in Classification and Transfer Learning

In chapter IV, we investigated the generalization from coarse to refined labels: a specific scenario where training and target labels were related, and the input domain remained unchanged. Therefore, we then extended our exploration to a more generalized context of transfer learning that occurs in classification settings. We considered the case where a model is trained on a set of classes C and demonstrates its ability to generalize to another set of classes C′.

We proposed a theoretical framework for accurately defining such scenarios. We revisit a general definition of separability: a set of classes C is deemed separable if any pair of classes of C is separated by at least one model. We proceed to define models as bi-partation, such as hyperplanes that split the space into two parts. We characterize models learned on pairs of classes and the set of pairs that they can effectively separate. This framework allows us to introduce an ordering relationship among models, revealing which pairs of classes are more useful when learning. Ideally, the concept attempts to address questions, such as, determining the minimal size of a set of classes C to learn to separate a set of classes C′.

Experimentally, we tested our approach on multiple scenarios, such as identifying optimal subsets of classes for fine-tuning or training a neural network; identifying classes that may pose difficulty in few-shot learning. While this framework retains simplicity, it serves as a valuable tool for identifying promising subsets of classes. Moreover, we observed that the choice of the deep learning architecture employed in building the models exhibits different models and ordering, suggesting differences in their underlying geometry. Future research could be conducted in the following directions:

- **Extension to Subsets of Classes**: While the current framework primarily focuses on pairs of classes, it could be extended to subsets of classes. This extension will offer the advantage of defining relationships of learning at different cardinalities. One application could involve generating, or decomposing a dataset into, classification datasets with varying degrees of complexity and investigating incremental learning strategies, for example.
- **Incorporation of Data Geometry**: The current framework provides limited insights into the geometry of the data. To enhance its effectiveness, theoretical results might be obtained by considering simple statistical hypotheses, such as Gaussian distributions to represent classes. Considering the work of (Fang et al., 2022) which proposed a "probably approximately correct" (PAC) approach for out-of-distribution scenarios, one could extend this work by seeing it as an extension in the context of transfer learning.
- **Investigating Architectural Differences**: Utilizing the framework to investigate the impact of different architectures and training processes on learning. In recent years, multiple architectures and training processes have been proposed by the community. This exploration could reveal how architectural choices lead to varying representations.
- **Characterizing Minimal Class Requirements**: Determining the minimum number of classes necessary to learn to distinguish all classes in a data set composed of N classes. While in worst-case scenarios, this number would be N, it is conceivable that very large datasets like ImageNet may be solvable with a smaller number of classes, paving the way for efficient transfer learning strategies.

6.1.5 Global conclusion

In summary, our research has delved into multiple scenarios where deep learning architectures demonstrate the ability to generalize beyond their initial training task. This ability has been extensively leveraged in the field of transfer learning. The ultimate goal is to achieve strong or general artificial intelligence capable of improvising and learning independently, addressing new and unseen tasks. As Demis Hassabis stated, ***transfer learning*** *is the key to general intelligence. [...] The key to successful transfer learning lies in the acquisition of* ***conceptual knowledge abstracted*** *from the perceptual details of its source*.

Consequently, over the years, the field of deep learning has witnessed a proliferation of new architectures, criteria, and training process introduced by the community to deliver increasingly abstract data representations. There has been a noticeable shift in research practices, marked by a growing tendency to utilize the same large pretrained models as a starting point and fine-tune it for specific target tasks. In the realm of computer vision, the ultimate aspiration is to develop a universal feature extractor capable of representing natural images that could solve any classification problem.

Yet, there remains room for investigating transfer learning and its underlying mechanisms. While self-supervised and unsupervised learning methods have become the dominant approaches in the field, the optimal training processes leading to the highest transferability are still ambiguous. This ambiguity is exacerbated by the fact that each architecture comes with its own training process (Wightman et al., 2021). As a result, fundamental questions persist, including:

- Which components of the architecture should be fine-tuned, or frozen?
- What type of information is extracted at each layer of the model?
- What geometric properties (invariance, equivariance) should be targeted by different parts of the model?
- Can we design an architecture with sufficient generality to avoid overfitting to the training task? What does it mean being overspecialized for a task, and in which scenarios is it problematic?

The last question might be unsolvable with current architectures. Let us consider our experience with learning from coarse-grained data to predict fine-grained labels. Our regularization technique enabled us to extract information about the fine-grained labels. However, achieving the right balance between the supervised learning criterion and the regularization was crucial. If we applied too much regularization, the problem remained unsolvable because no useful information was learned from the labeled data. On the other hand, insufficient regularization made the representation too specific to the coarse-grained labels. **The optimal balance between these two criteria depends on the specific target task**. A different target task would likely necessitate a different balance.

In this thesis, our exploration primarily centered around scenarios within the domain of classification. We examined transfer learning from one classification task to another, learned invariant operators with respect to classification, generated artificial tasks to enhance generalization on classification dataset. We delved into transfer learning within the realm of **pattern recognition**, highlighting how learning from specific data can yield general patterns applicable to other datasets.

However, as noted by (Lake et al., 2017), deep learning models, despite their prowess in pattern recognition and impressive performance across various tasks, still lack cer-

tain essential traits of human-like general intelligence. True general intelligence encompasses more than mere pattern recognition; it encompasses the ability to reason, learn from minimal examples (few-shot learning), and comprehend causal relationships between objects and events in the world.

Deep learning models have achieved remarkable success in tasks like image recognition, natural language processing, and playing complex games. However, they do not possess the same level of abstract reasoning and common-sense understanding that humans do. Deep learning has its strengths, but additional techniques and paradigms may be necessary to achieve the kind of versatile, human-like intelligence that many envision for the future of AI.

Bibliography

Abouelnaga, Y., Ali, O. S., Rady, H., & Moustafa, M. (2016). Cifar-10: knn-based ensemble of classifiers. *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, 1192–1195.

Amari, S.-i. (1993). Backpropagation and stochastic gradient descent method. *Neurocomputing*, *5*(4-5), 185–196.

Atanov, A., Filatov, A., Yeo, T., Sohmshetty, A., & Zamir, A. (2022). Task discovery: finding the tasks that neural networks generalize on. *arXiv preprint arXiv:2212.00261*.

Bachmann, G., Bécigneul, G., & Ganea, O. (2020). Constant curvature graph convolutional networks. *International Conference on Machine Learning*, 486–496.

Backstrom, L., & Leskovec, J. (2011). Supervised random walks: predicting and recommending links in social networks. *Proceedings of the fourth ACM international conference on Web search and data mining*, 635–644.

Baena, R., Drumetz, L., & Gripon, V. (2021). Inferring graph signal translations as invariant transformations for classification tasks. *2021 29th European Signal Processing Conference (EUSIPCO)*, 2169–2173.

Baena, R., Drumetz, L., & Gripon, V. (2022a). Preserving fine-grain feature information in classification via entropic regularization. *arXiv preprint arXiv:2208.03684*.

Baena, R., Drumetz, L., & Gripon, V. (2022b). Preventing manifold intrusion with locality: local mixup. *arXiv preprint arXiv:2201.04368*.

Baena, R., Drumetz, L., & Gripon, V. (2022c). Preventing manifold intrusion with locality: local mixup. *arXiv preprint arXiv:2201.04368*.

Bagherinezhad, H., Horton, M., Rastegari, M., & Farhadi, A. (2018). Label refinery: improving imagenet classification through label progression. *arXiv preprint arXiv:1805.02641*.

Bakhtin, A., Brown, N., Dinan, E., Farina, G., Flaherty, C., Fried, D., Goff, A., Gray, J., Hu, H., et al. (2022). Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, *378*(6624), 1067–1074.

Balestriero, R., Pesenti, J., & LeCun, Y. (2021). Learning in high dimension always amounts to extrapolation.

Bau, D., Zhou, B., Khosla, A., Oliva, A., & Torralba, A. (2017). Network dissection: quantifying interpretability of deep visual representations. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 6541–6549.

Bendou, Y., Hu, Y., Lafargue, R., Lioi, G., Pasdeloup, B., Pateux, S., & Gripon, V. (2022). Easy—ensemble augmented-shot-y-shaped learning: state-of-the-art few-shot classification with simple components. *Journal of Imaging*, *8*(7), 179.

Bengio, Y. (2012). Deep learning of representations for unsupervised and transfer learning. *Proceedings of ICML workshop on unsupervised and transfer learning*, 17–36.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: a review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798–1828.

Benson, A. R., Gleich, D. F., & Leskovec, J. (2016). Higher-order organization of complex networks. *Science*, *353*(6295), 163–166.

Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning* (Vol. 4). Springer.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Boland, R. P., & Urrutia, J. (1995). Separating collections of points in euclidean spaces. *Information Processing Letters*.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2021). On the opportunities and risks of foundation models. *arXiv:2108.07258*.

Bontonou, M., Lassance, C., Hacene, G. B., Gripon, V., Tang, J., & Ortega, A. (2019). Introducing graph smoothness loss for training deep learning architectures. *2019 IEEE Data Science Workshop (DSW)*, 160–164.

Bordes, F., Balestriero, R., Garrido, Q., Bardes, A., & Vincent, P. (2022). Guillotine regularization: improving deep networks generalization by removing their head. *arXiv preprint arXiv:2206.13378*.

Bottou, L., & Cun, Y. (2003). Large scale online learning. *Advances in neural information processing systems*, *16*.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., & Vandergheynst, P. (2017). Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, *34*(4), 18–42.

Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., & Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, *33*, 9912–9924.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., & Joulin, A. (2021). Emerging properties in self-supervised vision transformers. *CVPR*.

Carratino, L., Cissé, M., Jenatton, R., & Vert, J.-P. (2020). On mixup regularization. *arXiv preprint arXiv:2006.06049*.

Caruana, R. (1997). Multitask learning. *Machine learning*, *28*, 41–75.

Chan, W., Jaitly, N., Le, Q. V., & Vinyals, O. (2015). Listen, attend and spell. https://doi.org/10.48550/ARXIV.1508.01211

Chen, J., Sinha, S., & Kyrillidis, A. (2020). Stackmix: a complementary mix algorithm. *arXiv preprint arXiv:2011.12618*.

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. *International conference on machine learning*, 1597–1607.

Chidambaram, M., Wang, X., Hu, Y., Wu, C., & Ge, R. (2021). Towards understanding the data dependency of mixup-style training. *arXiv preprint arXiv:2110.07647*.

Chou, H.-P., Chang, S.-C., Pan, J.-Y., Wei, W., & Juan, D.-C. (2020). Remix: rebalanced mixup. *European Conference on Computer Vision*, 95–110.

Chung, F. R. (1997). *Spectral graph theory* (Vol. 92). American Mathematical Soc.

Cohen, T., & Welling, M. (2016). Group equivariant convolutional networks. *International conference on machine learning*, 2990–2999.

Cohen, T. S., Geiger, M., Köhler, J., & Welling, M. (2018). Spherical cnns. *arXiv preprint arXiv:1801.10130*.

Cosentino, R., Shekkizhar, S., Soltanolkotabi, M., Avestimehr, S., & Ortega, A. (2022). The geometry of self-supervised learning models and its impact on transfer learning. *arXiv preprint arXiv:2209.08622*.

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., & Bengio, Y. (2016). Binarized neural networks: training deep neural networks with weights and activations constrained to +1 or -1.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., & Le, Q. V. (2018). Autoaugment: learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*.

Cubuk, E. D., Zoph, B., Shlens, J., & Le, Q. V. (2020). Randaugment: practical automated data augmentation with a reduced search space. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 702–703.

d'Ascoli, S., Touvron, H., Leavitt, M. L., Morcos, A. S., Biroli, G., & Sagun, L. (2021). Convit: improving vision transformers with soft convolutional inductive biases. *International Conference on Machine Learning*, 2286–2296.

Davis, J., & Domingos, P. (2009). Deep transfer via second-order markov logic. *ICML*, 217–224.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: a large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition*, 248–255.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, *29*(6), 141–142.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

DeVries, T., & Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.

Dhillon, G. S., Chaudhari, P., Ravichandran, A., & Soatto, S. (2019). A baseline for few-shot image classification. *arXiv preprint arXiv:1909.02729*.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2014). Decaf: a deep convolutional activation feature for generic visual recognition. *International conference on machine learning*, 647–655.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Dubey, A., Gupta, O., Raskar, R., & Naik, N. (2018). Maximum-entropy fine grained classification. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/0c74b7f78409a4022a2c4c5a5ca3ee19-Paper.pdf

Dubey, A., Gupta, O., Raskar, R., Rahwan, I., & Naik, N. (2017). Regularizing prediction entropy enhances deep learning with limited data. *Proceedings of the Neural Information Processing Systems (NIPS)*.

Eshratifar, A. E., Eigen, D., Gormish, M., & Pedram, M. (2021). Coarse2fine: a two-stage training method for fine-grained visual classification. *Machine Vision and Applications*, *32*(2), 49.

Fang, Z., Li, Y., Lu, J., Dong, J., Han, B., & Liu, F. (2022). Is out-of-distribution detection learnable? *arXiv:2210.14707*.

Fei-Fei, L., Fergus, R., & Perona, P. (2004). Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. *CVPR workshop*, 178–178.

Fei-Fei, L., Fergus, R., & Perona, P. (2006). One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, *28*(4), 594–611.

Finn, C., Xu, K., & Levine, S. (2018). Probabilistic model-agnostic meta-learning. *Advances in neural information processing systems*.

Garrido, Q., Balestriero, R., Najman, L., & Lecun, Y. (2023). Rankme: assessing the downstream performance of pretrained self-supervised representations by their rank. *International Conference on Machine Learning*, 10929–10974.

Ghamisi, P., Yokoya, N., Li, J., Liao, W., Liu, S., Plaza, J., Rasti, B., & Plaza, A. (2017). Advances in hyperspectral image and signal processing: a comprehensive overview of the state of the art. *IEEE Geoscience and Remote Sensing Magazine*, *5*(4), 37–78.

Gidaris, S., Singh, P., & Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*.

Girault, B., Gonçalves, P., & Fleury, E. (2015). Translation on graphs: an isometric shift operator. *IEEE Signal Processing Letters*, *22*(12), 2416–2420.

Girault, B., Ortega, A., & Narayanan, S. S. (2018). Irregularity-aware graph fourier transforms. *IEEE Transactions on Signal Processing*, *66*(21), 5746–5761.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 580–587.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 249–256.

Gonzalez-Garcia, A., Modolo, D., & Ferrari, V. (2018). Do semantic parts emerge in convolutional neural networks? *International Journal of Computer Vision*, *126*, 476–494.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139–144.

Grandvalet, Y., & Bengio, Y. (2004). Semi-supervised learning by entropy minimization. *Advances in neural information processing systems*, *17*.

Greenewald, K., Gu, A., Yurochkin, M., Solomon, J., & Chien, E. (2021). K-mixup regularization for deep learning via optimal transport. *arXiv preprint arXiv:2106.02933*.

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. *International Conference on Machine Learning*, 1321–1330.

Guo, H., Mao, Y., & Zhang, R. (2019). Mixup as locally linear out-of-manifold regularization. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 3714–3722.

Guo, Y., Liu, Y., Bakker, E. M., Guo, Y., & Lew, M. S. (2018). Cnn-rnn: a large-scale hierarchical image classification framework. *Multimedia tools and applications*, *77*(8), 10251–10271.

Guo, Y., Zhang, Y., Chen, Y., & Yu, S. X. (2023). Unsupervised feature learning with emergent data-driven prototypicality. *arXiv preprint arXiv:2307.01421*.

Gyawali, P. K., Ghimire, S., & Wang, L. (2020). Enhancing mixup-based semi-supervised learning with explicit lipschitz regularization. *2020 IEEE International Conference on Data Mining (ICDM)*, 1046–1051.

Hacene, G. B., Lassance, C., Gripon, V., Courbariaux, M., & Bengio, Y. (2019). Attention based pruning for shift networks.

Hammond, D. K., Vandergheynst, P., & Gribonval, R. (2011). Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, *30*(2), 129–150. https://hal.inria.fr/inria-00541855

He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9729–9738.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., & Lakshminarayanan, B. (2019). Augmix: a simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*.

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Horn, G. V., Aodha, O. M., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., & Belongie, S. (2018). The inaturalist species classification and detection dataset.

Hu, Y., Gripon, V., & Pateux, S. (2021). Graph-based interpolation of feature vectors for accurate few-shot classification. *2020 25th International Conference on Pattern Recognition (ICPR)*, 8164–8171.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

Huh, M., Agrawal, P., & Efros, A. A. (2016). What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*.

Jang, E., Gu, S., & Poole, B. (2017). Categorical reparameterization with gumbel-softmax. *International Conference on Learning Representations (ICLR)*.

Janocha, K., & Czarnecki, W. M. (2017a). On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*.

Janocha, K., & Czarnecki, W. M. (2017b). On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*.

Jia, Y., Zhang, Y., Weiss, R., Wang, Q., Shen, J., Ren, F., Nguyen, P., Pang, R., Lopez Moreno, I., Wu, Y., et al. (2018). Transfer learning from speaker verification to multispeaker text-to-speech synthesis. *Advances in neural information processing systems*, *31*.

Jiang, J., & Zhai, C. (2007). Instance weighting for domain adaptation in nlp.

Jing, L., Vincent, P., LeCun, Y., & Tian, Y. (2021). Understanding dimensional collapse in contrastive self-supervised learning. *arXiv preprint arXiv:2110.09348*.

Jing, L., & Tian, Y. (2020). Self-supervised visual feature learning with deep neural networks: a survey. *IEEE transactions on pattern analysis and machine intelligence*, *43*(11), 4037–4058.

Jost, J., & Jost, J. (2008). *Riemannian geometry and geometric analysis* (Vol. 42005). Springer.

Kalatzis, D., Eklund, D., Arvanitidis, G., & Hauberg, S. (2020). Variational autoencoders with riemannian brownian motion priors. *arXiv preprint arXiv:2002.05227*.

Keskar, N. S., & Socher, R. (2017). Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*.

Kim, J.-H., Choo, W., & Song, H. O. (2020). Puzzle mix: exploiting saliency and local statistics for optimal mixup. *International Conference on Machine Learning*, 5275–5285.

Kingma, D. P., & Ba, J. (2014). Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks.

Kornblith, S., Chen, T., Lee, H., & Norouzi, M. (2021). Why do better loss functions lead to less transferable features? *Advances in Neural Information Processing Systems*, *34*.

Kraft, L. G. (1949). *A device for quantizing, grouping, and coding amplitude-modulated pulses*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90.

Krogh, A., & Hertz, J. (1991). A simple weight decay can improve generalization. *Advances in neural information processing systems*, *4*.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and brain sciences*, *40*, e253.

LeCun, Y. (2015a). Yoshua bengio, and geoffrey hinton. *Deep learning. nature*, *521*(7553), 436–444.

LeCun, Y. (2015b). Yoshua bengio, and geoffrey hinton. *Deep learning. nature*, *521*(7553), 436–444.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lenc, K., & Vedaldi, A. (2015). Understanding image representations by measuring their equivariance and equivalence. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 991–999.

Li, J., Socher, R., & Hoi, S. C. (2020). Dividemix: learning with noisy labels as semi-supervised learning. *arXiv preprint arXiv:2002.07394*.

Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2017). Diffusion convolutional recurrent neural network: data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*.

Liu, X., Li, S., Kan, M., Zhang, J., Wu, S., Liu, W., Han, H., Shan, S., & Chen, X. (2015). Agenet: deeply learned regressor and classifier for robust apparent age estimation. *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 16–24.

Liu, Y., Liu, L., Zhang, H., Rezatofighi, H., Yan, Q., & Reid, I. (n.d.). Meta learning with differentiable closed-form solver for fast video object segmentation. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 8439–8446.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022). A convnet for the 2020s. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 11976–11986.

Liu, Z., Li, S., Wu, D., Chen, Z., Wu, L., Guo, J., & Li, S. Z. (2021). Automix: unveiling the power of mixup. *arXiv preprint arXiv:2103.13027*.

Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Maddison, C. J., Mnih, A., & Teh, Y. W. (2017). The concrete distribution: a continuous relaxation of discrete random variables. *International Conference on Learning Representations (ICLR)*.

Magoulas, G. D., & Prentza, A. (1999). Machine learning in medical applications. *Advanced course on artificial intelligence*, 300–307.

Mallat, S. (2016). Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *374*(2065), 20150203.

Mangla, P., Kumari, N., Sinha, A., Singh, M., Krishnamurthy, B., & Balasubramanian, V. N. (2020). Charting the right manifold: manifold mixup for few-shot learning.

Proceedings of the IEEE/CVF winter conference on applications of computer vision, 2218–2227.

Masci, J., Boscaini, D., Bronstein, M., & Vandergheynst, P. (2015a). Geodesic convolutional neural networks on riemannian manifolds. *Proceedings of the IEEE international conference on computer vision workshops*, 37–45.

Masci, J., Boscaini, D., Bronstein, M., & Vandergheynst, P. (2015b). *Shapenet: convolutional neural networks on non-euclidean manifolds* (tech. rep.).

Meister, C., Salesky, E., & Cotterell, R. (2020). Generalized entropy regularization or: there's nothing special about label smoothing. *arXiv preprint arXiv:2005.00820*.

Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., & Terzopoulos, D. (2021). Image segmentation using deep learning: a survey. *IEEE transactions on pattern analysis and machine intelligence*, *44*(7), 3523–3542.

Monti, F., Otness, K., & Bronstein, M. M. (2018). Motifnet: a motif-based graph convolutional network for directed graphs. *2018 IEEE Data Science Workshop (DSW)*, 225–228.

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., & Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model cnns. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5115–5124.

Müller, R., Kornblith, S., & Hinton, G. E. (2019). When does label smoothing help? *Advances in neural information processing systems*, *32*.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., & Sutskever, I. (2021). Deep double descent: where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, *2021*(12), 124003.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning.

Oh, J., & Yun, C. (2023). Provable benefit of mixup for finding optimal decision boundaries. *arXiv preprint arXiv:2306.00267*.

Oord, A. v. d., Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Oppenheim, A. V. (1999). *Discrete-time signal processing*. Pearson Education India.

Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1717–1724.

Oreshkin, B., Rodriguez Lopez, P., & Lacoste, A. (2018). Tadam: task dependent adaptive metric for improved few-shot learning. *Advances in neural information processing systems*.

Ortega, A., Frossard, P., Kovačević, J., Moura, J. M. F., & Vandergheynst, P. (2018). Graph signal processing: overview, challenges, and applications. *Proceedings of the IEEE*, *106*(5), 808–828.

Ortega, A., Frossard, P., Kovačević, J., Moura, J. M., & Vandergheynst, P. (2018). Graph signal processing: overview, challenges, and applications. *Proceedings of the IEEE*, *106*(5), 808–828.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*.

Papernot, N., & McDaniel, P. (2018). Deep k-nearest neighbors: towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*.

Park, C., Yun, S., & Chun, S. (2022). A unified analysis of mixed sample data augmentation: a loss function perspective. *arXiv preprint arXiv:2208.09913*.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *International conference on machine learning*, 1310–1318.

Pasdeloup, B., Gripon, V., Vialatte, J.-C., Grelier, N., & Pastor, D. (2018). A neighborhood-preserving translation operator on graphs.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., & Yang, B. (2020). Geom-gcn: geometric graph convolutional networks.

Peng, W., Varanka, T., Mostafa, A., Shi, H., & Zhao, G. (2021). Hyperbolic deep neural networks: a survey. *IEEE Transactions on pattern analysis and machine intelligence*, *44*(12), 10023–10044.

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł., & Hinton, G. (2017). Regularizing neural networks by penalizing confident output distributions. *International Conference on Learning Representations (ICLR)*.

Peyré, G., & Cuturi, M. (2019). Computational optimal transport: with applications to data science. *Foundations and Trends® in Machine Learning*, *11*(5-6), 355–607.

Pratt, L. Y. (1992). Discriminability-based transfer between neural networks. *Advances in neural information processing systems*, *5*.

Qin, L., & Zhu, X. (2013). Promoting diversity in recommendation by entropy regularizer. *Twenty-Third International Joint Conference on Artificial Intelligence*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning transferable visual models from natural language supervision.

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434*.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. Y. (2007). Self-taught learning: transfer learning from unlabeled data. *ICML*, 759–766.

Rame, A., Sun, R., & Cord, M. (2021). Mixmo: mixing multiple inputs for multiple outputs via deep subnetworks. *arXiv preprint arXiv:2103.06132*.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779–788.

Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H., & Zemel, R. S. (2018). Meta-learning for semi-supervised few-shot classification. *arXiv:1803.00676*.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, *28*.

Ristin, M., Gall, J., Guillaumin, M., & Van Gool, L. (2015). From categories to subcategories: large-scale image classification with partial class label refinement. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 231–239.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, 400–407.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10684–10695.

Rosenstein, M. T., Marx, Z., Kaelbling, L. P., & Dietterich, T. G. (2005). To transfer or not to transfer. *NIPS 2005 workshop on transfer learning*, *898*(3).

Rothe, R., Timofte, R., & Van Gool, L. (2015). Dex: deep expectation of apparent age from a single image. *Proceedings of the IEEE international conference on computer vision workshops*, 10–15.

Rothe, R., Timofte, R., & Van Gool, L. (2018). Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision*, *126*(2-4), 144–157.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*, 211–252.

Sandryhaila, A., & Moura, J. M. F. (2013). Discrete signal processing on graphs. *IEEE Transactions on Signal Processing*, *61*(7), 1644–1656.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, *20*(1), 61–80.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: a unified embedding for face recognition and clustering. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 815–823.

Shen, Z., Liu, Z., Xu, D., Chen, Z., Cheng, K.-T., & Savvides, M. (2021). Is label smoothing truly incompatible with knowledge distillation: an empirical study. *arXiv preprint arXiv:2104.00676*.

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, *6*(1), 1–48.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., & Vandergheynst, P. (2013). The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, *30*(3), 83–98.

Shuman, D. I., Ricaud, B., & Vandergheynst, P. (2012a). A windowed graph fourier transform. *2012 IEEE Statistical Signal Processing Workshop (SSP)*, 133–136.

Shuman, D. I., Ricaud, B., & Vandergheynst, P. (2012b). A windowed graph fourier transform. *2012 IEEE Statistical Signal Processing Workshop (SSP)*, 133–136.

Shwartz-Ziv, R., & Tishby, N. (2017). Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, *529*(7587), 484–489.

Simard, P., Lecun, Y., Denker, J., & Victorri, B. (2001). Transformation invariance in pattern recognition – tangent distance and tangent propagation. *International Journal of Imaging Systems and Technology*, *11*.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Smith, L. N. (2017). Cyclical learning rates for training neural networks. *2017 IEEE winter conference on applications of computer vision (WACV)*, 464–472.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Stewart, L., Bach, F., Berthet, Q., & Vert, J.-P. (2023). Regression as classification: influence of task formulation on neural network features. *International Conference on Artificial Intelligence and Statistics*, 11563–11582.

Sukhbaatar, S., Bruna, J., Paluri, M., Bourdev, L., & Fergus, R. (2014). Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*.

Sun, C., Huang, L., & Qiu, X. (2019). Utilizing bert for aspect-based sentiment analysis via constructing auxiliary sentence. *arXiv preprint arXiv:1903.09588*.

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. *International conference on machine learning*, 1139–1147.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818–2826.

Taherkhani, F., Kazemi, H., Dabouei, A., Dawson, J., & Nasrabadi, N. M. (2019). A weakly supervised fine label classifier enhanced by coarse supervision. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6459–6468.

Takahashi, R., Matsubara, T., & Uehara, K. (2019). Data augmentation using random image cropping and patching for deep cnns. *IEEE Transactions on Circuits and Systems for Video Technology*, *30*(9), 2917–2931.

Tan, M., & Le, Q. (2019). Efficientnet: rethinking model scaling for convolutional neural networks. *International conference on machine learning*, 6105–6114.

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., & Isola, P. (2020a). What makes for good views for contrastive learning? *Advances in neural information processing systems*, *33*, 6827–6839.

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., & Isola, P. (2020b). What makes for good views for contrastive learning? *Advances in neural information processing systems*, *33*, 6827–6839.

Touvron, H., Sablayrolles, A., Douze, M., Cord, M., & Jégou, H. (2021). Grafit: learning fine-grained image representations with coarse labels. *Proceedings of the IEEE/CVF international conference on computer vision*, 874–884.

Tripuraneni, N., Jordan, M., & Jin, C. (2020). On the theory of transfer learning: the importance of task diversity. *Advances in neural information processing systems*.

Tschannen, M., Djolonga, J., Rubenstein, P. K., Gelly, S., & Lucic, M. (2019). On mutual information maximization for representation learning. *arXiv preprint arXiv:1907.13625*.

Van Den Oord, A., Vinyals, O., et al. (2017). Neural discrete representation learning. *Advances in neural information processing systems*, *30*.

van den Berg, E. (2016). Some insights into the geometry and training of neural networks. *arXiv:1605.00329*.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(11).

Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., & Bengio, Y. (2019). Manifold mixup: better representations by interpolating hidden states. *International Conference on Machine Learning*, 6438–6447.

Vialatte, J.-C., Gripon, V., & Coppin, G. (2017). Learning local receptive fields and their weight sharing scheme on graphs.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. (2016). Matching networks for one shot learning. *Advances in neural information processing systems*, *29*.

Voulodimos, A., Doulamis, N., Doulamis, A., Protopapadakis, E., et al. (2018). Deep learning for computer vision: a brief review. *Computational intelligence and neuroscience*, *2018*.

Wah, C., Branson, S., Welinder, P., Perona, P., & Belongie, S. (2011). The caltech-ucsd birds-200-2011 dataset.

Wang, M., & Deng, W. (2018). Deep visual domain adaptation: a survey. *Neurocomputing*, *312*, 135–153.

Wang, Y., Chao, W., Weinberger, K., & van der Maaten, L. S. (2019). Revisiting nearest-neighbor classification for few-shot learning. *arXiv:1911.04623*.

Wang, Y., Yao, Q., Kwok, J. T., & Ni, L. M. (2020). Generalizing from a few examples: a survey on few-shot learning. *ACM computing surveys (csur)*, *53*(3), 1–34.

Wightman, R., Touvron, H., & Jégou, H. (2021). Resnet strikes back: an improved training procedure in timm. *arXiv preprint arXiv:2110.00476*.

Wu, C., Tygert, M., & LeCun, Y. (2017). A hierarchical loss and its problems when classifying non-hierarchically. *arXiv preprint arXiv:1709.01062*.

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xu, Y., Ding, J., Zhang, L., & Zhou, S. (2021). Dp-ssl: towards robust semi-supervised learning with a few labeled samples. *Advances in Neural Information Processing Systems*, *34*.

Yang, L., Hanneke, S., & Carbonell, J. (2013). A theory of transfer learning with applications to active learning. *Machine learning*, *90*, 161–189.

Yeh, R. A., Hu, Y.-T., Hasegawa-Johnson, M., & Schwing, A. (2022a). Equivariance discovery by learned parameter-sharing. *International Conference on Artificial Intelligence and Statistics*, 1527–1545.

Yeh, R. A., Hu, Y.-T., Hasegawa-Johnson, M., & Schwing, A. (2022b). Equivariance discovery by learned parameter-sharing. *International Conference on Artificial Intelligence and Statistics*, 1527–1545.

Yin, W., Wang, H., Qu, J., & Xiong, C. (2021). Batchmixup: improving training by interpolating hidden states of the entire mini-batch.

Ying, W., Zhang, Y., Huang, J., & Yang, Q. (2018). Transfer learning via learning to transfer. *International conference on machine learning*, 5085–5094.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in neural information processing systems*, *27*.

Yu, Y., Jiang, H., Bahri, D., Mobahi, H., Kim, S., Rawat, A. S., Veit, A., & Ma, Y. (2021). An empirical study of pre-trained vision models on out-of-distribution generalization. *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*.

Yuan, L., Tay, F. E., Li, G., Wang, T., & Feng, J. (2019). Revisit knowledge distillation: a teacher-free framework.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., & Yoo, Y. (2019). Cutmix: regularization strategy to train strong classifiers with localizable features. *Proceedings of the IEEE/CVF international conference on computer vision*, 6023–6032.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, 818–833.

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, *64*(3), 107–115.

Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). Mixup: beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Zhang, R., Isola, P., & Efros, A. A. (2016). Colorful image colorization. *European conference on computer vision*, 649–666.

Zhang, Z., & Sabuncu, M. (2020). Self-distillation as instance-specific label smoothing. *Advances in Neural Information Processing Systems*, *33*, 2184–2195.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2921–2929.

Zhou, C., Loy, C. C., & Dai, B. (2021). Denseclip: extract free dense labels from clip. *arXiv:2112.01071*.

Zhou, K., Yang, J., Loy, C. C., & Liu, Z. (2022). Learning to prompt for vision-language models. *International Journal of Computer Vision*, *130*(9), 2337–2348.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2020a). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, *109*(1), 43–76.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2020b). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, *109*(1), 43–76.

Index

Titre : Au-delà de la tâche d'entraînement en classification : un regard sur des extensions de la notion de généralisation

Mots clés : Deep Learning, Classification, Apprentissage Supervisé, Transfer Learning, Généralisation

Résumé : La thèse s'intéresse à la notion de généralisation, en particulier dans le cadre de la classification en apprentissage automatique de manière supervisée. Cette approche consiste à apprendre à résoudre une tâche (classification) à partir de données d'entrainement étiquetées. La généralisation est définie comme la capacité à réaliser des prédictions correctes sur des données non observées pendant l'entraînement. Cette notion est généralement restreinte à des données qui correspondent au même domaine que celui de la tâche d'entrainement.

Cependant, une littérature récente met en exergue la capacité des architectures d'apprentissage profond à généraliser en dehors de leur tâche d'entrainement. Ainsi, un modèle entrainé sur une tâche particulière peut être réutilisée en partie sur d'autres tâches.

Ainsi, nous explorons différentes extensions possibles de la généralisation : apprentissage sur un ensemble de classe et l'habilité à généraliser sur un ensemble de classes plus grands, l'apprentissage sur des labels grossiers pour prédire des labels plus complexes, l'apprentissage sur une tâche artificiellement complexe pour améliorer la généralisation ou encore l'apprentissage d'opérateurs invariants pour une tâche spécifique.

Title : Beyond the training task in classification : looking at extension of the notion of generalization

Keywords : Deep Learning, Classification, Supervised Learning, Transfer Learning, Generalization

Abstract : The thesis focuses on the concept of generalization, particularly in the context of supervised machine learning classification. This approach involves learning to solve a task (classification) based on labeled training data. Generalization is defined as the ability to make accurate predictions on unseen data during training. Traditionally, generalization is limited to data that belongs to the same domain as the training task.

However, recent literature highlights the capability of deep learning architectures to generalize beyond their training task. Thus, a model trained on a specific task can be partially reused for other tasks.

The thesis explores various possible extensions of generalization, including learning on a set of classes and the ability to generalize to a larger set of classes, learning on coarse labels to predict more complex labels, learning on artificially complex tasks to improve generalization, and learning invariant operators for a specific task.