Conservatoire National des Arts et Métiers

APPRENTISSAGE STATISTIQUE :
MODÉLISATION DÉCISIONNELLE ET
APPRENTISSAGE PROFOND
Département d'Informatique

PROJET DE L'UE RCP209

DISTILLATION DE MODÈLES

Enseignants :
Arnaud BRELOY
Marin FERECATU
Clément RAMBOUR
Wafa AISSA

Etudiant: Arnaud FELDMANN

ANNÉE ACADÉMIQUE 2023 / 2024

Table des matières

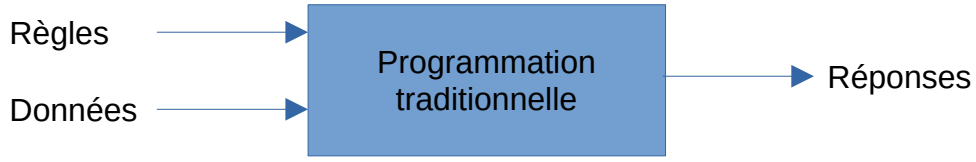
Introduction

Sujet

Distillation de modèle. Les réseaux de neurones de l'état de l'art sont généralement complexes et coûteux en temps de calcul. Il a été montré que des approches dites de "distillation" permettent de transférer les connaissances d'un réseau de neurones complexe à un réseau de neurones plus simple, et donc plus économe en calculs. On implémentera une méthode de distillation utilisant la régression des logits d'un réseau complexe (par ex. un ResNet-50) entraîné sur le jeu de données CIFAR-10 ou CIFAR-100 vers un réseau plus simple (par ex. un ResNet-18).

Vers des modèles légers ?

Un programme informatique peut être modélisé comme un ensemble de règles et une sortie qui est déduite de ces règles. Traditionnellement, le développeur et l'ingénieur convenaient, de manière rigide, de ces dernières. Ils les traduisaient sous la forme de structures syntaxiques spécifiques, ce qui permettait alors de déduire les sorties à partir des données.



Mais ce type de programmation ne passe pas à l'échelle dans tous les domaines. Est-il possible d'expliciter à l'ordinateur ce qu'est précisément une image de chien ? Un chien a quatre pattes ; à moins qu'il ne soit estropié. Si une patte avant est à gauche de l'écran, l'autre patte avant a de grandes chances d'y être aussi. Un chien est plutôt petit, à moins d'être un Terre Neuve. Un chien est souvent marron, mais pas toujours. On comprend que les limites des langages rendent difficile d'expliciter ce qu'est un chien de manière synthétique.

La philosophie de l'apprentissage supervisé renverse la manière d'écrire ces règles, devenue inductive. À partir d'une partie des données et des réponses, il s'agit de déduire automatiquement des règles sous-jacentes, elles-mêmes étant traduites sous la forme de code binaire, potentiellement volumineux, que personne n'a explicité à l'avance. Ces "règles" peuvent ensuite être appliquées à d'autres échantillons de données pour procéder à des inférences.

L'apprentissage supervisé donne de bons résultats. C'est une perspective extrêmement réjouissante, mais l'usage de règles déduites suppose l'ajustement d'un nombre considérable de paramètres, ce qui peut engendrer des modèles assez lourds.

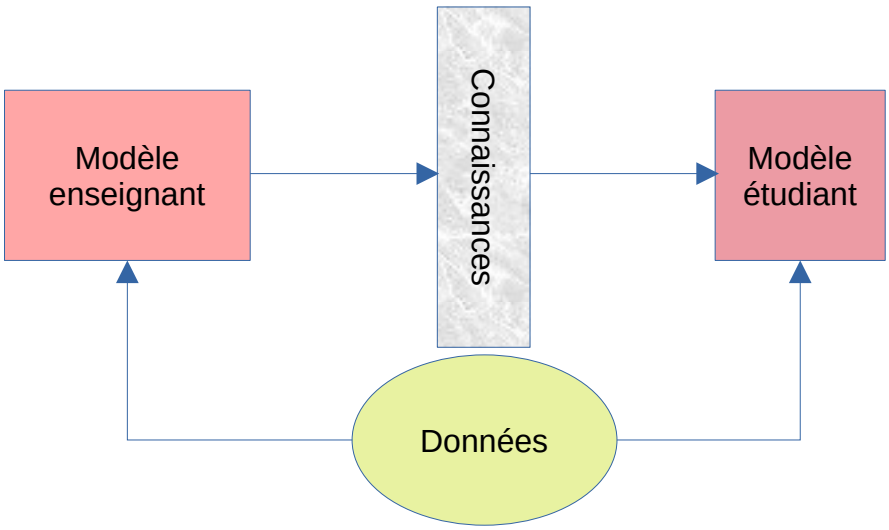
Cette importante taille est problématique à plusieurs égards. D'une part, un nombre important de paramètres induit un certain gâchis de ressources, tant en espace mémoire qu'en termes d'énergie. D'autre part, de nombreux cas d'usage actuels reposent sur l'informatique embarquée : "Alexa, rédige mon rapport sur la distillation de modèles pour RCP209 !". L'informatique embarquée est certes devenue plus performante qu'à l'accoutumée, ce qui permet des applications étonnantes[1]. Mais les cas d'usage en production, cependant, sont le plus souvent restrictifs. Par exemple, quand bien même les téléphones récents sont très puissants, ceux-ci sont matériellement restreints par leur autonomie. Aucun utilisateur n'accepterait que l'intégralité de l'autonomie de l'appareil soit consacrée à surveiller l'occurrence d'"OK GOOGLE"[2]. De même, l'univers des microcontrôleurs est fortement contraint ; il y faut consommer le moins possible, tant pour faire tenir les modèles en mémoire que pour les faire fonctionner sur des accumulateurs[3].

Plusieurs stratégies sont définies pour pouvoir répondre à ces problématiques de compacité et d'efficacité. La **quantification** (quantification) repose sur une discrétisation du signal et des coefficients. L'**élagage de poids** (pruning) consiste à essayer de supprimer les synapses superficiels superflus. Le **partage de poids** (weight clustering) consiste à regrouper les poids d'une couche en différents groupes, et à ne retenir que leurs centroïdes[4].

[1] Mes raspberry Pi 5 sont suffisantes pour faire tourner des modèles de langage cf https://ollama.afeldmann.fr/ même si comme je suis en ADSL il faut attendre pas mal au chargement.
[2] Les téléphones, en effet, ont leurs propres modèles de détection, qui est ajusté pour un fort rappel mais une précision légèrement moins importante : une fois que le téléphone a un doute, il peut alors y avoir un enchainement de modèles de plus en plus couteux en cascade. D'abord sur le téléphone, puis ensuite sur le serveur. Outre une plus grande efficacité énergétique, cela permet une parcimonie sur le partage des données individuelles.
[3] Parfois, les accumulateurs doivent alimenter le système pendant de très longues périodes d'autonomie https://www.edx.org/learn/tinyml/harvard-university-deploying-tinyml
[4] Ainsi, on diminue l'entropie des poids du réseau de neurones, ce qui facilite sa compression https://www.tensorflow.org/model_optimization/guide?hl=fr

Distillation de modèles

Cependant, ces astuces de compression de modèles sont incomplètes si l'on souhaite non seulement optimiser les modèles, mais également diminuer la complexité des modèles en elle-même. L'on peut toujours essayer des astuces, un baobab adulte ne rentrera pas dans un pot de 20 centimètres de diamètre. C'est là qu'intervient la stratégie de **distillation de modèle** : comment mobiliser les connaissances acquises par un modèle complexe et les transmettre, *par l'apprentissage*, à un modèle plus simple ?



L'idée derrière la distillation de modèles est de rapprocher, par l'apprentissage, des modèles le plus souvent de tailles différentes. Plaçons-nous, pour ce projet, dans le cas concret de l'apprentissage supervisé. **Qu'est-ce que le gros modèle a à transmettre de plus qu'un label ?** Après tout, dans le cas spécifique de l'apprentissage supervisé, la cible est déjà connue ; qu'est-ce que le modèle enseignant nous apprendra de plus ?

1. Théorie

1.1. Différents types de connaissances

1.1.1. Les réponses

L'idée sous-jacente derrière le rapprochement des réponses est de profiter d'une information supplémentaire acquise par le modèle enseignant : les probabilités des différentes classes. La fonction de perte alors utilisée est alors le plus souvent dérivée de la divergence Kullback-Leibler[5]. Pour une observation unique, on a :

$$L_{distillation} = L\left(P_{enseignant}, P_{étudiant}\right)$$

Où les P correspondent aux distributions respectives données par les différents modèles, pour cette observation.

Dans le cas où l'on travaille avec des softmax de température T, l'on utilise une entropie croisée mise à l'échelle en divisant par le carré de la température, ceci afin de préserver l'échelle des gradients lorsque l'on fait varier les températures. On a donc finalement[6] :

$$L_{distillation} = -\alpha \log p^{étu}_{i*,1} + (1-\alpha) T^2 \sum_i p^{ens}_{i,T} \log \frac{p^{ens}_{i,T}}{p^{étu}_{i,T}}$$

avec :

- i correspond aux différentes classes
- i* correspond à l'index de la cible dure
- $p_{i,T} = \frac{e^{s_i/T}}{\sum_i e^{s_i/T}}$ correspond aux probabilités des cibles dites douces
- s_i correspond au logit de la classe i

Il est possible d'utiliser d'autres fonctions de perte, notamment les moindres carrés[7]. L'essentiel à comprendre est que, par rapport à un label one-hot binaire et exclusif, une probabilité estimée, même imparfaite, peut apporter de l'information utile. Toutes les probabilités, de toutes les classes, servent alors dans l'optimisation.

Hinton souligne que ce qui convient le mieux à la distillation sont des températures importantes. Avec des températures importantes, on donne en effet plus d'importance aux "secondes classes possibles" ; c'est-à-dire qu'on force le modèle à rapprocher le camion de la voiture pour l'éloigner de la carotte. Cependant, pousser les modèles à

[5] Knowledge Distillation: A Survey, de Jianping Gou et al. https://arxiv.org/pdf/2006.05525.pdf
[6] https://keras.io/examples/vision/knowledge_distillation/
[7] Comparing Kullback-Leibler Divergence and Mean Squared Error Loss in Knowledge Distillation, de Taehyon Kim et al. https://arxiv.org/pdf/2105.08919.pdf

l'incertitude envers leur premier choix, c'est faire confiance aux deuxièmes. Dans le cas de datasets erratiques, avec des labels incorrects, les températures élevées sont peu recommandables.

1.1.2. Les caractéristiques (features)

Au delà du rapprochement des logits, il est également possible de rapprocher les caractéristiques *internes* des deux modèles, les features. Dans ce cas, la fonction de coût dérive de deux fonctions de transformation et d'une comparaison.

$$L_{distillation} = L\left(\Phi_1\left(c_{enseignant}\right), \Phi_2\left(c_{étudiant}\right)\right)$$

Comme pour le rapprochement des réponses, différentes fonctions de perte sont possibles. L peut être dérivé de L1, de L2, de l'entropie croisée... L'usage des distances dérivées des normes est d'autant plus naturel ici qu'à l'intérieur des réseaux de neurones, les caractéristiques représentent moins souvent des probabilités qu'en couche de sortie.

1.1.3. Les relations entre caractéristiques

Les relations entre les différentes caractéristiques données par un modèle peuvent aussi être considérées comme une connaissance acquise. Dans ce cas là, l'on va pénaliser les différences de similarités.

$$L_{distillation} = L\left(\Phi_1\left(c_{enseignant,1}, c_{enseignant,2}\right), \Phi_2\left(c_{étudiant,1}, c_{étudiant,2}\right)\right)$$

L'idée sous-jacente est de dire que, par exemple, deux données doivent être corrélées de manière similaire dans les deux modèles.

1.2. Différents types de distillation

La **distillation hors-ligne** évalue le modèle enseignant, puis optimise le modèle étudiant selon les métriques préalablement définies. Dans la **distillation en ligne**, les deux modèles sont entraînés en même temps. Ce type de fonctionnement peut par exemple être adapté aux flux de données dont la distribution change avec le temps. L'**auto-distillation** elle utilise les connaissances déduites d'un modèle sur lui-même. Par exemple dans le cas des architectures par ResBlocks, les prédictions des différents ResBlocks peuvent être liées entre elles[8].

Des algorithmes spécifiques existent dans la distillation : la distillation adversaire dans laquelle un modèle génère des observations et l'autre essaye de les discriminer, la distillation avec plusieurs enseignants... Dans le cadre de ce projet, nous nous concentrerons uniquement sur la distillation simple telle que décrite par Hinton.

[8] Be Your Own Teacher: Improve the Performance of Convolutional Neural Networks via Self Distillation, de Linfeng Zhang et al. https://arxiv.org/abs/1905.08094

2. Application

L'application proposée dans ce projet[9] est, comme le stipule le sujet, d'utiliser ResNet50 pour entraîner ResNet18, avec l'aide de CIFAR-100, en rapprochant les logits du modèle étudiant de celui du modèle enseignant.

Notre objectif est d'utiliser le modèle enseignant pour obtenir un **modèle étudiant compact qui soit un bon compromis entre respect du modèle enseignant et des cibles dures**.

2.1. Éléments de contexte

2.1.1. ResNet50 et ResNet18

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 1: Architecture des ResNet

ResNet50 est un réseau de neurones convolutifs profond. Il est constitué de 50 couches (hors couches secondaires comme le padding, les activations...). Ces couches sont organisées par blocs résiduels, dont le principe est d'être court-circuités en sortie par une addition avec leur entrée. Cette architecture permet la bonne diffusion des gradients parmi les différentes couches d'une suite de blocs, ce qui permet de diminuer le problème de disparition du gradient (vanishing gradient problem). À la fin d'un ResNet50, une piscine globale permet de réduire la dimensionnalité et de ne sortir ainsi que 2048 caractéristiques.

ResNet50 est disponible avec des poids issus d'un pré-entrainement sur le dataset ImageNet. C'est ce modèle pré-entrainé que l'on prendra ici comme modèle enseignant,

[9] Les sources complètes sont disponibles sur
https://github.com/arnaud-feldmann/projet_distillation_cnam/

après transfert et réglage fin. ResNet18 est une version moins complexe de ResNet mais dispose d'une architecture assez proche (cf tableau d'architecture des ResNet).

2.1.2. CIFAR-100 et augmentation de données

CIFAR-100 est un dataset d'images étiquetées avec 100 classes. Chacune des classes fait théoriquement partie d'une des 20 superclasses, mais celles-ci sont ignorées dans le cadre de ce projet, l'objet n'étant pas tant la prédiction la plus pertinente de CIFAR-100 que l'étude de la distillation. Les dimensions des images de CIFAR-100 sont de 32*32*3, mais nous les redimensionnons en 224*224*3, pour deux raisons :

- Nous nous conformons à la dimensionnalité d'ImageNet pour pouvoir en exploiter les poids par transfert.
- L'architecture des ResNet est prévue pour des images dont la dimensionnalité est voisine de celle d'ImageNet. La première couche des ResNet, par exemple, révèle directement par convolution 64 caractéristiques avec un filtre 7*7 et un pas de 2. Si on a des images 32*32, cela ferait en sortie, sans padding, directement des images 13*13. On comprend que l'architecture d'un réseau convolutif dépend de la dimensionnalité attendue.

En plus de cet agrandissement des images, nous utilisons, sur le dataset d'entraînement seulement, des fonctions d'augmentation de données. Cela permet d'ajouter un peu de bruit, et donc de diversité, sur les données, ce qui permet une plus confortable généralisation. Contrairement au redimensionnement appliqué une bonne fois pour toutes, l'augmentation est dynamique : à chaque époque, les transformations sont différentes. Cela permet de limiter le surajustement.

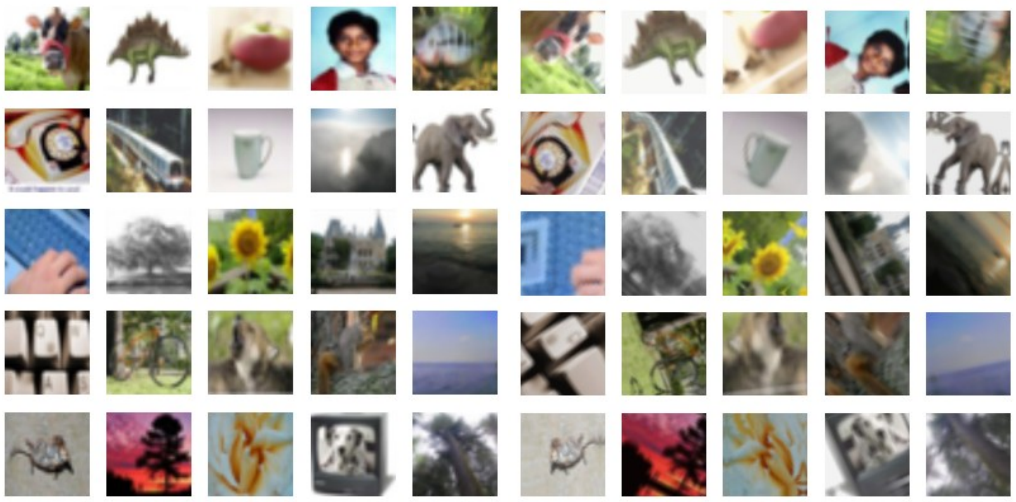


Figure 2 : Les premières images du dataset d'entrainement

Figure 3 : Les mêmes images après une augmentation

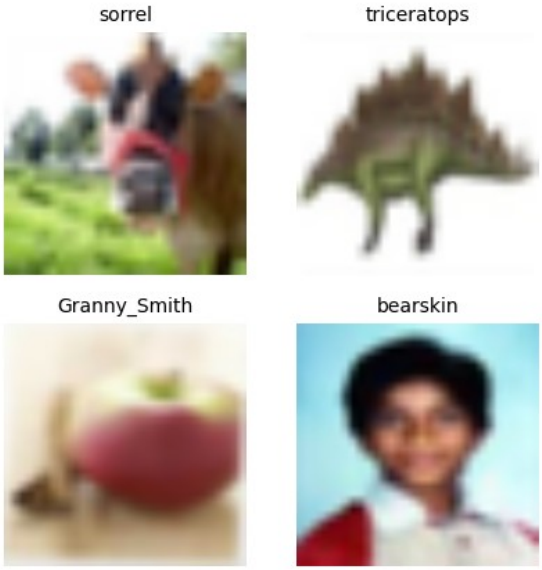


Figure 4: Les 4 premières images munies de leurs prédictions via le prétraitement et ResNet50

A posteriori de ces couches de génération de données, nous avons ajouté une couche finale de prétraitement de sorte à ce que le format des données corresponde exactement à celui du dataset ImageNet. Un petit test de prédiction sans entraînement supplémentaire montre que le pré-traitement fonctionne correctement (cf figure 4).

2.2. Un modèle de distillation de réponse hors-ligne, exporté avec élagage et quantification

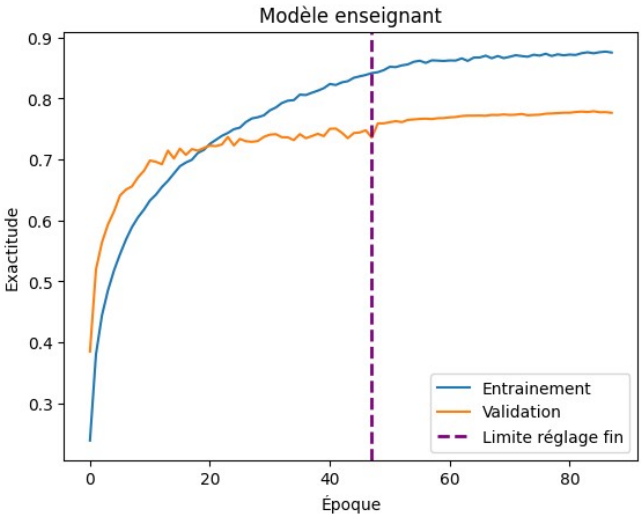
2.2.1. Entraînement du modèle enseignant

La première chose à faire pour obtenir un modèle de distillation de réponse hors-ligne, c'est d'entraîner un modèle enseignant. Comme prévu, on entraîne donc le modèle ResNet50 sur le dataset CIFAR-100. L'architecture complète utilisée figure ci-dessous.

Celle-ci contient une régularisation L1 sur la couche cachée entre le réseau convolutif et la sortie. L1 permet de forcer la sélection de caractéristiques, ce qui est bien ce que l'on veut faire. À contrario, la régularisation appliquée sur la dernière couche est plutôt L2. On veut éviter le surajustement, mais, vu que le rôle de sélection de caractéristiques est déjà accompli la régularisation précédente, il ne semble pas nécessaire de pousser ici vers des zéros stricts.

En ce qui concerne la couche cachée, on utilise des sigmoïdes plutôt que des activations Unitaires Linéaires Rectifiées. C'est un choix purement arbitraire guidé par l'idée que, dans la perspective de tester la distillation, l'on préfère a priori un joli modèle à la pure efficacité mathématique.

```
model = Sequential([
    ResNet50V2(include_top=False,
weights='imagenet', pooling="avg"),
    Dropout(0.25),
    Dense(256, activation="sigmoid",
kernel_regularizer =
tf.keras.regularizers.L1(0.001)),
    Dropout(0.5),
    Dense(num_classes, activation="softmax",
kernel_regularizer =
tf.keras.regularizers.L2(0.001))
])
```



Lors de la première phase d'apprentissage, l'apprentissage de transfert, en plus des couches denses que nous avons rajoutées, seuls les éléments dont le nom correspond à l'expression régulière **"^.*(_3_conv|_bn)$"** sont entraînés. Les _bn correspondent aux normalisations de batch, les _3_conv correspondent aux convolutions à la fin de chaque bloc, juste avant les additions. Libérer sélectivement des couches tout le long du réseau tire partie de l'architecture des blocs résiduels ; leur architecture permet mieux la diffusion du signal que sur un réseau de neurones dont on n'aurait libéré les *dernières* couches. Après entraînement, nous obtenons 76,5 % sur l'échantillon de test.

2.2.2. Choix et entraînement du modèle étudiant

Le modèle étudiant est entraîné sur le même modèle, à ceci que :

- Le ResNet est maintenant un ResNet18
- Le coût est maintenant tel que décrit en 1.1.1
- L'optimiseur Adam est remplacé par un Stochastic Gradient Descent. Curieusement, Adam, qui était efficace et rapide sur le modèle ResNet50, se révèle cette fois assez inefficace sur le modèle ResNet18. La perte d'exactitude observée est considérable, autour de 8 points sur le modèle témoin. Une explication possible, non investiguée dans le cadre de ce projet, pourrait être que Adam introduit une régularisation implicite en limitant la croissance des variables hautement contributives.

Pour comparer les deux hyperparamètres, α et T, nous utilisons trois valeurs différentes de T (1, 3 et 8) et quatre de α (0, 0.1, 0.25 et 0.5). En plus de ces 12 couples valeurs est rajouté un modèle témoin d'hyperparamètres T=1 et α=1, qui correspond à l'absence complète de distillation.

L'implémentation de la distillation ici utilisée a été écrite en Tensorflow, à l'aide de GradientTape. L'usage direct des fonctions de Tensorflow permet de garantir la pleine reproductibilité comparative en diminuant l'aspect boîte noire de Keras. En outre, en utilisant Keras, l'accès aux mécaniques internes de l'apprentissage est moins libre, ce qui complique par exemple la production de métriques tenant compte des cibles multiples. Keras est plus simple, mais son API favorise un apprentissage supervisé classique, avec un *y_true* et un *y_pred*.

Les modèles sont testés sur 50 époques, avec un taux d'apprentissage de 0.001 pendant 40 époques puis de 0.0001 pendant 10 époques. Cette distribution rigide des taux d'apprentissage est convenue pour améliorer la comparabilité des tests. Deux exactitudes différentes sont mesurées. L'une, par rapport aux étiquettes dures, mesure la justesse du modèle étudiant dans l'absolu. L'autre, par rapport aux étiquettes douces,

mesure la propension de la distillation à rapprocher le modèle enseignant du modèle étudiant.

Les résultats sont résumés dans les deux tableaux ci-dessous.

Résultats d'exactitude de la distillation sur les étiquettes dures du dataset de validation

(meilleur modèle atteint à la 10ème | 20ème | 50ème époque)

T	**α = 0**	**α = 0.1**	**α = 0.25**	**α = 0.5**	**α = 1 (témoin)**
1	0.58 | 0.63 | 0.72	0.58 | 0.65 | 0.73	0.59 | 0.64 | 0.73	0.58 | 0.65 | 0.73	0.56 | 0.62 | 0.71
3	0.56 | 0.63 | 0.69	0.59 | 0.64 | 0.72	0.59 | 0.65 | 0.73	0.57 | 0.64 | 0.74	
8	0.43 | 0.54 | 0.62	0.47 | 0.58 | 0.68	0.54 | 0.63 | 0.69	0.57 | 0.65 | 0.73	

Note de lecture : Au bout de 50 époques, avec les hyperparamètres α = 0.1 et T = 3, l'exactitude maximale atteinte par le modèle est de 72 % sur les étiquettes dures, contre 71 % pour le modèle témoin.

Résultats d'exactitude de la distillation sur les étiquettes douces du dataset de validation

(meilleur modèle atteint à la 10ème | 20ème | 50ème époque)

T	**α = 0**	**α = 0.1**	**α = 0.25**	**α = 0.5**	**α = 1 (témoin)**
1	0.59 | 0.64 | 0.73	0.59 | 0.66 | 0.74	0.60 | 0.66 | 0.74	0.59 | 0.65 | 0.74	0.58 | 0.63 | 0.72
3	0.56 | 0.63 | 0.70	0.59 | 0.66 | 0.73	0.60 | 0.66 | 0.74	0.58 | 0.65 | 0.75	
8	0.44 | 0.56 | 0.63	0.48 | 0.59 | 0.69	0.55 | 0.64 | 0.70	0.58 | 0.66 | 0.74	

Note de lecture : Au bout de 50 époques, avec les hyperparamètres α = 0.1 et T = 3, l'exactitude du meilleur modèle atteint 73 % par rapport aux étiquettes douces, contre 72 % pour le modèle témoin.

Le meilleur modèle est T=3 et α = 0.5. Son exactitude face aux étiquettes dures est de 73,5 %, contre 71,4 % pour le modèle témoin. Celle face aux étiquettes douces est de 74,6 %, contre 72,4 % pour le modèle témoin. Il faut minorer cet écart d'un biais : le modèle témoin n'est lui lancé qu'une seule fois, et est comparé à 12 essais de distillation. La lecture de *l'ensemble* des résultats semble cependant montrer que l'impact de la distillation est supérieur à la variance entre les essais, puisque le tableau a un message assez régulier.

Ces résultats illustrent, dans ce contexte hors-ligne précis, une performance légère de la distillation du modèle enseignant pour améliorer les exactitudes face aux étiquettes douces comme dures. À hyperparamètre α faible, la température a un effet néfaste, ce

qui peut surprendre car ce n'est pas conforme avec les températures hautes que recommande Hinton. Le meilleur modèle demeure de température 3.

L'une des explications possibles à la faiblesse du gain d'exactitude face aux étiquettes dures est que les performances d'apprentissage du modèle témoin ResNet18 sont déjà sont proches, dans le contexte de ce dataset, des performances du modèle ResNet50. ResNet18 est *déjà* un modèle facilement optimisable.

La distillation n'améliore pas beaucoup l'exactitude face aux étiquettes douces non plus. Un tel paradoxe avait déjà été constaté dans la publication *Does Knowledge Distillation Really Work?*. Quels que soient les hyperparamètres, l'on peut constater que les exactitudes face aux étiquettes douces sont toujours proches de celles face aux étiquettes dures.

Devant la grande homogénéité des exactitudes des modèles, l'on pourrait émettre l'hypothèse que certaines observations sont difficiles à classer *dans l'absolu* et que les différents processus d'apprentissage buttent sur les mêmes. C'est le cas pour la plupart des observations selon la table de Burt ci-dessous. La probabilité pour une observation mal classée par le modèle enseignant d'être mal classée par le modèle étudiant témoin est de deux tiers, contre un cinquième en ce qui concerne les observations correctement prédites par le modèle maître. Cependant, l'on peut faire une contre-observation surprenante : un quart des observations mal apprises par le modèle enseignant le sont correctement par le modèle étudiant totalement distillé. Ce qui, dans le contexte d'un dataset à 100 classes, est très supérieur au hasard uniforme.

Table de Burt comparant les classements
des différents modèles sur l'échantillon de test

		Modèle enseignant		**Modèle étudiant α = 1 (témoin)**		**Modèle étudiant α = 0 et T=1**	
		Bon	**Faux**	**Bon**	**Faux**	**Bon**	**Faux**
Modèle enseignant	**Bon**	7651	0	6309	1342	6331	1320
	Faux	0	2349	765	1584	671	1678
Modèle étudiant α = 1 (témoin)	**Bon**	6309	765	7074	0	6092	982
	Faux	1342	1584	0	2926	910	2096
Modèle étudiant α = 0 et T=1	**Bon**	6331	671	6092	910	7002	0
	Faux	1320	1678	982	2016	0	2998

Note de lecture : Sur les 7651 observations correctement étiquetées par le modèle enseignant dans l'échantillon de test, 6309 le sont par le modèle étudiant témoin et 6331 le sont par le modèle étudiant totalement distillé.

Dans le détail, comme le montre le tableau de probabilités ci-dessous, les modèles ResNet18 utilisés sont beaucoup moins sûrs d'eux que ne l'est le ResNet50. Leur probabilité estimée est très loin de la majorité absolue, y compris lorsque leur prédiction est correcte. C'est sans doute cette relative incertitude qui les rend capables, parfois, de contredire l'enseignant avec justesse. Leurs premiers choix sont plutôt corrects, qui plus est.

Probabilité estimées moyennes des classes choisies
par chaque modèle et dans chaque cas
(enseignant | étudiant témoin | étudiant distillé)

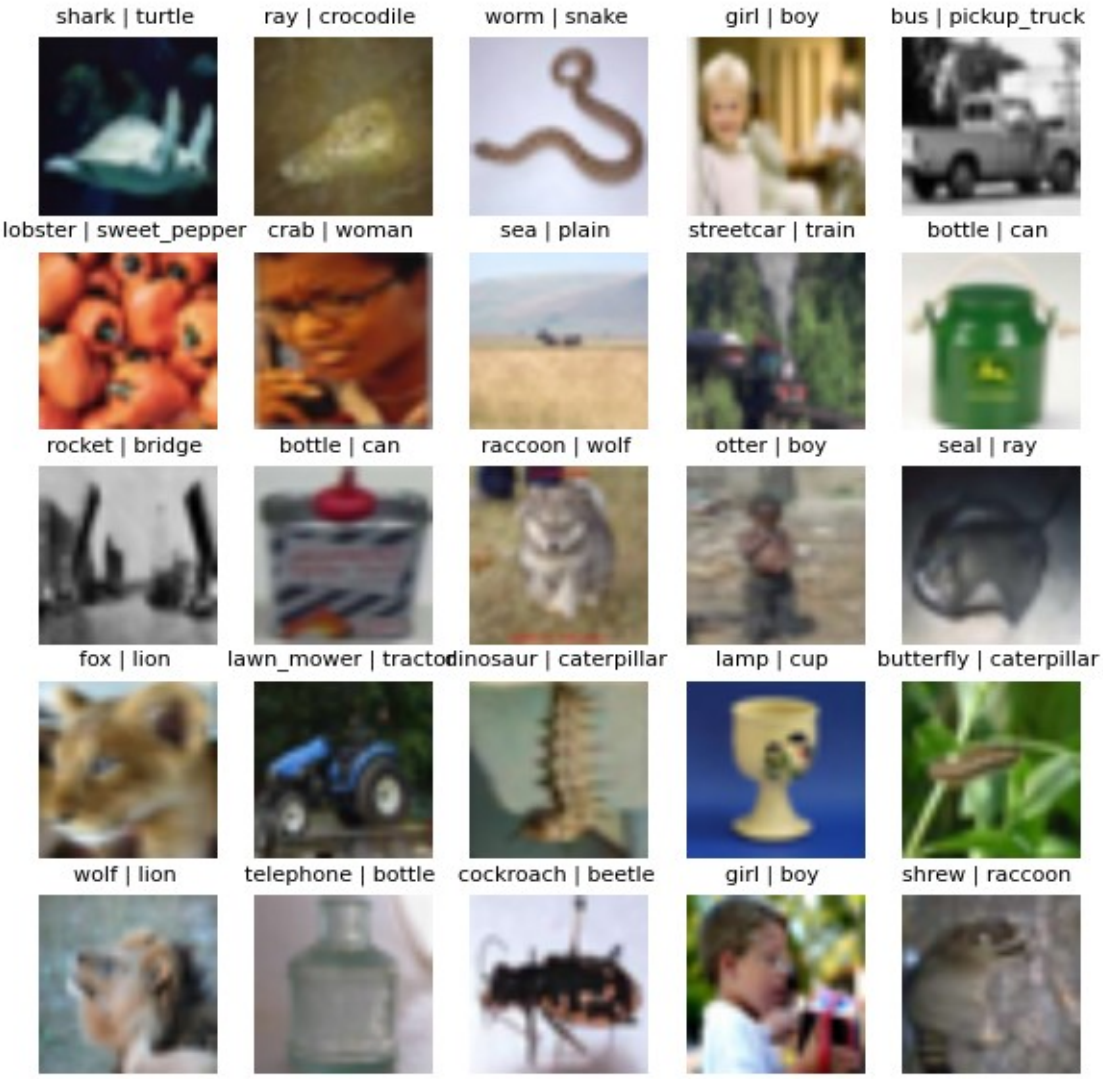
		Modèle enseignant		**Modèle étudiant α = 1 (témoin)**		**Modèle étudiant α = 0 et T=1**	
		Bon	**Faux**	**Bon**	**Faux**	**Bon**	**Faux**
Modèle enseignant	**Bon**	88 | 27 | 23		92 | 31 | 26	74 | 10 | 10	92 | 30 | 26	72 | 12 | 9
	Faux		57 | 12 | 10	58 | 14 | 11	56 | 10 | 9	**57 | 14 | 11**	57 | 11 | 9
Modèle étudiant α = 1 (témoin)	**Bon**	92 | 31 | 26	58 | 14 | 11	88 | 29 | 24		91 | 32 | 27	71 | 13 | 9
	Faux	74 | 10 | 10	56 | 10 | 9		64 | 10 | 9	73 | 10 | 11	60 | 10 | 9
Modèle étudiant α = 0 et T=1	**Bon**	92 | 30 | 26	57 | 14 | 11	91 | 32 | 27	73 | 10 | 11	88 | 29 | 25	
	Faux	72 | 12 | 9	57 | 11 | 9	71 | 13 | 9	60 | 10 | 9		64 | 11 | 9

Note de lecture : Pour les observations mal étiquetées par le modèle enseignant mais étiquetées correctement par le modèle étudiant distillé (valeurs en gras), la probabilité moyenne de la classe estimée par le modèle enseignant est de 57 %, celle du modèle témoin est de 14 %, et celle du modèle étudiant distillé n'est que de 11 %.

Cette analyse mixte est confirmée par la visualisation de quelques différences de prédictions dans le cas étrange où l'étudiant totalement distillé a raison contre l'enseignant (voir ci-après). Dans beaucoup de cas, il y a des ambigüités claires (shark/turtle, worm/snake, girl/boy...) et les images sont dures à classifier. Dans quelques cas, l'enseignant se trompe complètement sans que les images ne semblent particulièrement difficiles à l'humain (crab/woman, otter/boy, ...)[10]. Dans quelques derniers cas, il y a une imprécision de l'enseignant sans qu'il n'y ait ambigüité humaine sur le bon choix (bus / pickup_truck, wolf/lion, ...). Les raisons de contredire l'enseignant avec justesse sont donc plurielles.

[10] On remarquera que ces cas d'échec complet interviennent ici sur des personnes racisées, ce qui doit interpeller et inciter les concepteurs de cas d'usage à toujours utiliser les données les plus diverses, en particulier si les cas d'usage sont sensibles.

Prévisions des deux modèles quand le modèle
étudiant distillé a raison contre l'enseignant
(enseignant | étudiant distillé)



2.2.3. Distillation, élagage et quantification

Après passage du ResNet50 au ResNet18, le modèle est déjà réduit d'une taille d'archive zippée de 132 Mo à 42 Mo. Peut-on faire encore plus compact ? Pour ce faire, nous testons un élagage à 60 % suivi d'une quantification. Le modèle que l'on sélectionne pour cet export est T=3 et α = 0.5, car il obtient le meilleur score d'exactitude sur les étiquettes douces et dures.

Une phase de distillation supplémentaire a été testée pendant l'élagage[11], mais obtenait de moins bons résultats que la méthodologie simplifiée suivante. Pour parachever l'entrainement du modèle distillé, on a simplement fait une moyenne des cibles douces et dures et entrainé le modèle pendant 10 époques supplémentaires en même temps que l'élagage.

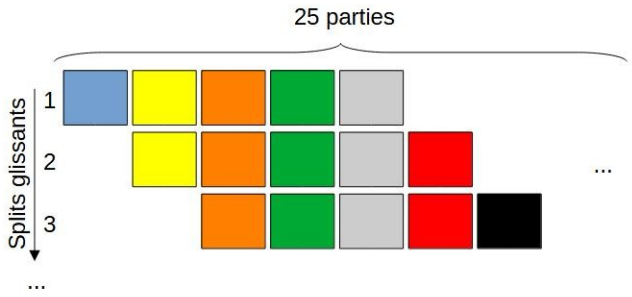
Les comparaisons de tailles se font après archive zip pour ne pas favoriser les formats avec compression, et pour mettre en avant l'impact de l'élagage.

Modèle	Taille zippée
Enseignant ResNet50	132 Mo
Etudiant ResNet18 *(T=3 et a = 0.5)*	42 Mo
Etudiant ResNet18 *(T=3 et a = 0.5)* après élagage 60 %	18 Mo
Etudiant ResNet18 *(T=3 et a = 0.5)* après élagage 60 % et quantification	6 Mo

Le modèle a très bien réagi à l'élagage : nous obtenons une **exactitude sur le modèle finale de 77 % sur les étiquettes dures, supérieure à celle du modèle enseignant** ! L'exactitude face aux étiquettes douces n'est pas en reste non plus, avec 77 %. Nous avons donc réussi à obtenir un modèle léger proche du modèle initial.

2.3. Distillation en ligne

Pour compléter l'expérience de distillation hors-ligne, une distillation en ligne est cette fois tentée, selon des modalités voisines. Il s'agit toujours d'utiliser ResNet50 pour entraîner ResNet18, mPour compléter l'expérience de distillation hors-ligne, une distillation en ligne est cette fois tentée, selon des modalités voisines. Il s'agit toujours d'utiliser ResNet50 pour entraîner ResNet18, mais les deux modèles sont ajustées en même temps, selon une logique de flux. Pour ce faire, les 45 000 observations d'entrainement, après mise à l'écart des 5 000 données de validation, ont été réunies en 25 parties de 1800 observations. Ces dernières sont ensuite regroupées en 21 splits glissants, un split glissant contenant les 5 dernières parties.



[11] cf le commit fc528a7 par exemple.

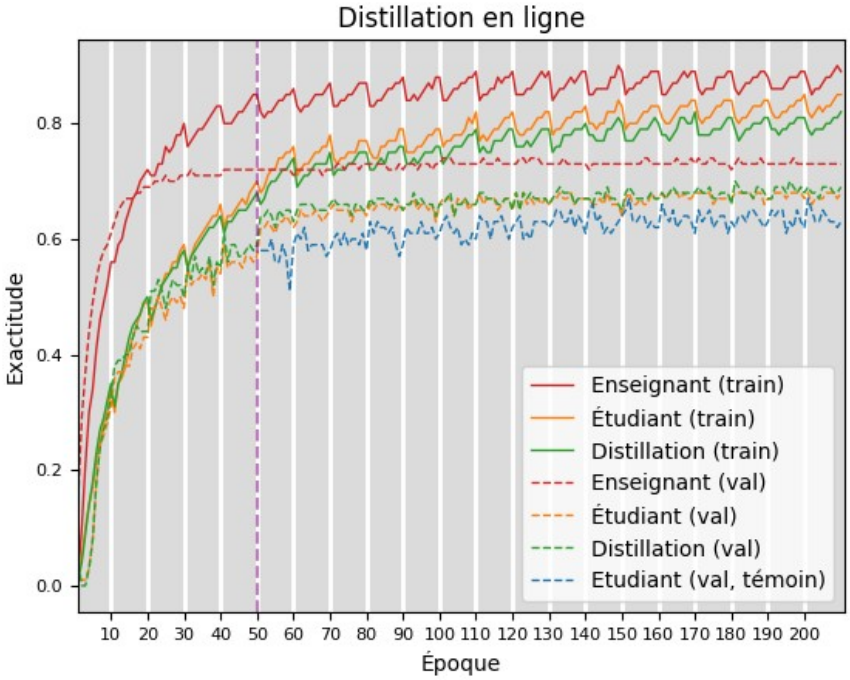
Une telle méthodologie est établie pour simuler une arrivée des données par séries de 1800 images dans un buffer de taille 9000. Ce buffer représente un compromis entre priorisation des données récentes et mémorisation du passé. Si l'on ne mémorise pas du tout le passé, on risque en effet l'oubli catastrophique (*catastrophic forgetting*).

Le choix des paramètres d'apprentissage (nombre d'époques, taux d'apprentissage) est assez sensible. Il faut qu'ils soient suffisamment forts pour permettre d'ingérer les nouvelles données parmi les coefficients ainsi que de rapprocher, en continu, l'étudiant de l'évolution de l'enseignant. Il faut cependant éviter les excès de surajustement sur les données récentes. Il y a donc un arbitrage à faire qui dépend du cas d'usage : peut-être qu'en production on acceptera un très léger surapprentissage si l'on constate que des données voisines dans le flux sont corrélées, à défaut d'introduire un réseau récurrent.

Concrètement, l'optimiseur de l'étudiant est une descente stochastique de gradient avec un taux d'apprentissage de 0,001 pendant 9 époques sur chaque split, puis de 0,0001 pendant 1 dernière époque. L'optimiseur de l'enseignant est lui un Adam constant à 0,001. Le modèle enseignant n'est jamais complètement libéré : on n'entraine que les couches supérieures.

Les hyperparamètres de distillation testés, choisis par analogie avec l'expérience de distillation hors-ligne, sont $T=3$ et $\alpha = 0.5$. La distillation n'est elle-même enclenchée qu'à partir du sixième split, pour laisser à l'enseignant le temps de converger vers quelque chose de raisonnable. Si on ne procède pas de la sorte, la distillation d'un modèle non-convergé semble abîmer durablement les couches profondes durant les premiers splits. En outre, déclencher ponctuellement la distillation permet d'apprécier son efficacité.

Par suite, un réentrainement témoin est accompli en recommençant à partir des coefficients du sixième split. De cette manière, l'on peut considérer tout ce qui se passe à partir du sixième split est un embranchement avec et sans distillation.



Lors du déclenchement de la distillation, celle-ci fait d'un seul coup gagner cinq points à l'étudiant en exactitude de validation, tant face aux étiquettes douces que dures. Le gain est donc, dans ce cas d'usage, supérieur à celui constaté pendant l'expérience hors-ligne. Ce surplus d'exactitude est par la suite conservé, quoi que légèrement attenué, lors de la convergence des différents modèles : l'usage de distillation réduit de moitié l'écart entre les plateaux respectifs du modèle étudiant et du modèle enseignant.

Il est à noter que l'expérience de distillation hors ligne réduisait déjà la différence entre l'exactitude de validation du modèle enseignant et celle du modèle étudiant. Cependant, cette réduction est ici bien plus perceptible.

L'on aurait pu s'attendre à profiter d'une rapidité d'exécution accrue par la conjonction des deux entraînements et par les possibilités de parallélisation induites. Les deux descentes de gradient sont dans une même boucle et leur optimisation est indépendante. Il n'y a pas eu de gain de rapidité[12] malgré l'usage d'une tf.function qui permet cette parallélisation des calculs. Peut-être que les ressources de la carte graphique étaient déjà suffisamment exploitées pour ne pas pouvoir profiter d'une marge de parallélisation supplémentaire.

[12] L'expérience n'est pas complètement comparable car des batchs de 100 ont été utilisés pour l'entrainement initial du modèle enseignant, contre des batchs de 64 tant pour la distillation hors-ligne que celle ligne. Toutefois, la durée d'une époque avec ces valeurs de batchs est environ de 280s pour l'entrapinement du modèle enseignant, 340s pour la distillation hors-ligne, et 990s pour la distillation en ligne. Il y a donc fort à parier que, tel que cela a été appliqué dans le cadre de ce projet du moins, la distillation en ligne n'était pas vectrice d'une plus grande vitesse.

Conclusion

On a exploré dans ce rapport la distillation au sens de Hinton. Il s'agit d'une méthode de rapprochement des logits produits par deux modèles via la minimisation de leur entropie croisée. Cette *distillation* peut potentiellement servir à de nombreux cas d'usage. Par exemple, il peut s'agir de réinvestir les connaissances acquises par un modèle volumineux dans un contexte physiquement contraint, tel que celui de l'informatique embarquée. La distillation permet de réduire l'écart d'exactitudes constaté entre deux modèles.

Dans le cas hors-ligne, l’architecture du ResNet18 est suffisante pour s’ajuster convenablement sur CIFAR-100. Il n’est pas certain que le temps consacré à la distillation soit un choix pertinent en ce contexte, même s’il pourrait être intéressant d'étudier des formes de distillation plus lourdes et rigides.

Dans le cas de l’apprentissage en ligne, le gain semble bien plus direct. Le ResNet50 en transfert-learning se comporte mieux que le ResNet18 face au flux de données, et ces connaissances améliorent nettement les résultats du ResNet18.