AVANCÉES DES *Large Langage Models* POUR LES RÉSEAUX DE NEURONES

RECENT ADVANCES IN MACHINE LEARNING

Théo Lopès-Quintas

BPCE Payment Services,
Université Paris Dauphine

30 mai 2024

Comment bien modifier la valeur du learning rate ?

Comment bien modifier la valeur du learning rate ?

Il faut modifier la valeur du learning rate

On cherche à minimiser une fonction de perte $\mathcal{L}$ qui s'écrit très souvent en Machine Learning comme la somme d'une fonction sur chaque observations du dataset. On note $\hat{y}_i$ la prédiction d'un algorithme pour l'observation x_i avec $i \leqslant n$ l'index de l'observation i. Pour un problème de régression et de classification, on a par exemple :

$$\begin{aligned} \mathcal{L}(\theta) &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}\ell_i(\theta) \end{aligned}$$

$$\begin{aligned} \mathcal{L}(\theta) &= -\frac{1}{n}\sum_{i=1}^{n} y_i \ln(\hat{y}_i) + (1 - y_i)\ln(1 - \hat{y}_i) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\ell_i(\theta) \end{aligned}$$

Ainsi, pour une unique mise à jour, on doit appliquer n fonctions ℓ. La descente de gradient stochastique consiste à sélectionner aléatoirement un index i_t et mettre à jour les poids avec uniquement cette observations. La descente de gradient devient alors :

$$\theta_{t+1} = \theta_t - \eta_t \nabla \ell_{i_t}(\theta_t)$$

Comment bien modifier la valeur du learning rate ?

Il faut modifier la valeur du learning rate

Notons que la descente de gradient stochastique n'est pas vraiment une *descente* : nous ne pourrons garantir une descente qu'en espérance. On suppose que $\mathcal{L}$ est différentiable et β-smooth, qu'il existe un minimum pour cette fonction et que chaque ℓ_i soit différentiable continue.

Comment bien modifier la valeur du learning rate ?

Il faut modifier la valeur du learning rate

Notons que la descente de gradient stochastique n'est pas vraiment une *descente* : nous ne pourrons garantir une descente qu'en espérance. On suppose que $\mathcal{L}$ est différentiable et β-smooth, qu'il existe un minimum pour cette fonction et que chaque ℓ_i soit différentiable continue.
Alors, puisque $\mathcal{L}$ est β-smooth :

$$\mathcal{L}\left(\theta_{t+1}\right) \leqslant \mathcal{L}\left(\theta_t\right) + \left\langle \nabla \mathcal{L}\left(\theta_t\right), \theta_{t+1} - \theta_t \right\rangle + \frac{\beta}{2} \| \overbrace{\theta_{t+1} - \theta_t}^{\eta_t \nabla \ell_{i_t}(\theta_t)} \|^2$$

Comment bien modifier la valeur du learning rate ?

Il faut modifier la valeur du learning rate

Notons que la descente de gradient stochastique n'est pas vraiment une *descente* : nous ne pourrons garantir une descente qu'en espérance. On suppose que $\mathcal{L}$ est différentiable et β-smooth, qu'il existe un minimum pour cette fonction et que chaque ℓ_i soit différentiable continue.
Alors, puisque $\mathcal{L}$ est β-smooth :

$$\mathcal{L}(\theta_{t+1}) \leqslant \mathcal{L}(\theta_t) + \langle \nabla \mathcal{L}(\theta_t), \theta_{t+1} - \theta_t \rangle + \frac{\beta}{2} \| \overbrace{\theta_{t+1} - \theta_t}^{\eta_t \nabla \ell_{i_t}(\theta_t)} \|^2$$

Ainsi, en espérance sur le choix aléatoire de i_t, on a :

$$\mathbb{E}[\mathcal{L}(\theta_{t+1})] \leqslant \mathcal{L}(\theta_t) + \eta_t \langle \nabla \mathcal{L}(\theta_t), \mathbb{E}[\nabla \ell_{i_t}(\theta_t)] \rangle + \frac{\beta \eta_t^2}{2} \mathbb{E}\left[\|\nabla \ell_{i_t}(\theta_t)\|^2 \right]$$

Cela ne nous assure pas pour autant une garantie de descente en espérance. Nous devons faire des hypothèses supplémentaires.

Hypothèse 1

Un index i_t d'une mise à jour t est tiré selon :

- i_t *ne dépend pas des index* $i_0, \ldots, i_{t-1}$
- $\nabla \ell_{i_t}(\theta_t)$ *est un estimateur non biaisé de* $\nabla \mathcal{L}(\theta_t)$
- $\mathbb{E}\left[\|\nabla \ell_{i_t}(\theta_t)\|^2\right] \leqslant \sigma^2 + \|\mathcal{L}(\theta)\|^2$

Si les indices sont tirés uniformément alors les deux premières hypothèses sont vérifiées. Pour le troisième point, s'il existe $M > 0$ tel que pour tout itérations t on a $\|\nabla \ell_{i_t}(\theta_t)\| \leqslant M$, alors il est vérifié. Dans ce cas, on a une garantie de descente en espérance :

$$\mathbb{E}\left[\mathcal{L}\left(\theta_{t+1}\right)\right] \leqslant \mathcal{L}\left(\theta_t\right) - \left(\eta_t - \frac{\beta \eta_t^2}{2}\right) \|\nabla \mathcal{L}(\theta_t)\|^2 + \frac{\beta \eta_t^2}{2} \sigma^2 \tag{1}$$

En supposant que $\eta_t \leqslant \dfrac{1}{\beta}$.

Comment bien modifier la valeur du learning rate ?

Il faut modifier la valeur du learning rate

On se place dans le cadre d'une optimisation d'une fonction de perte $\mathcal{L}$ non convexe, mais β-smooth. On conserve les hypothèses (1) précédentes. On note $\theta^* = \arg\min_{\theta \in \mathbb{R}^d} \mathcal{L}(\theta)$

Théorème 1 (Learning rate fixe)

On considère une descente de gradient stochastique avec $\eta_t = \eta$ tel que $\eta \in \left]0, \frac{1}{\beta}\right]$. Alors, pour tout $T \geqslant 1$:

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} \|\nabla \mathcal{L}(\theta_t)\|^2\right] \leqslant \eta\beta\sigma^2 + \frac{2(\mathcal{L}(\theta_0) - \mathcal{L}(\theta^*))}{\eta T}$$

On a donc que $\lim_{T\to+\infty} \mathbb{E}\left[\min_{0\leqslant t\leqslant T-1} \|\nabla \mathcal{L}(\theta_t)\|^2\right] \in [0, \eta\beta\sigma^2]$: le bruit nous empêche de converger vers un point de gradient nul. Ce n'est pas non plus une garantie de convergence vers le minimum global.

Comment bien modifier la valeur du learning rate ?

Il faut modifier la valeur du learning rate

On se place dans le cadre d'une optimisation d'une fonction de perte $\mathcal{L}$ non convexe, mais β-smooth. On conserve les hypothèses (1) précédentes.On note $\theta^* = \arg\min_{\theta \in \mathbb{R}^d} \mathcal{L}(\theta)$

Théorème 2 (Pas de descente décroissant)

On considère une descente de gradient stochastique avec η_t une suite décroissante telle que $\eta_t \in \left]0, \frac{1}{\beta}\right]$ et que :

$$\sum_{t=0}^{+\infty} \eta_t = +\infty \quad \textit{et} \quad \sum_{t=0}^{+\infty} \eta_t^2 < +\infty$$

Alors, pour tout $T \geqslant 1$:

$$\lim_{T \to +\infty} \mathbb{E}\left[\frac{1}{\sum_{t=0}^{T-1} \eta_t} \sum_{t=0}^{T-1} \eta_t \|\nabla \mathcal{L}(\theta_t)\|^2\right] = 0$$

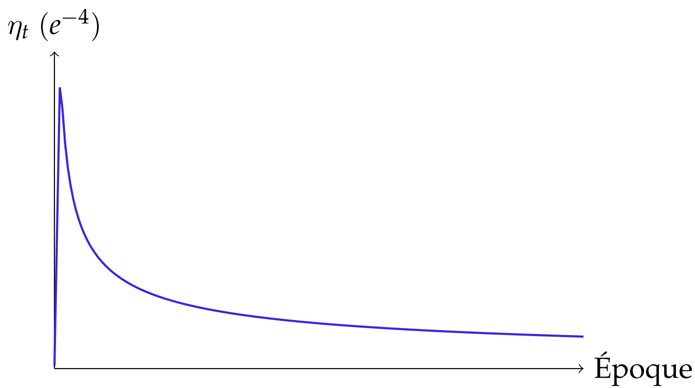
Nous n'avons plus cette fois une convergence dans un intervalle proche du minimum, mais une convergence vers un point de gradient nul. D'où la nécessité d'avoir un échéancier pour le choix du learning rate. De même pour Nesterov, nous avions besoin d'un échéancier pour le momentum.

Comment bien modifier la valeur du learning rate ?

Échéancier des Transformers originels

Attention is all you need [Vaswani et al., 2017] introduit l'architecture transformers et également un échéancier associé :

$$\eta_{\underbrace{t}_{\textbf{Époque}}} = \frac{1}{\sqrt{\underbrace{d}_{\textbf{Dimension du modèle = 512}}}} \min\left\{\frac{1}{\sqrt{t}}, t\, \overbrace{\tau}^{\textbf{Époques de warmup = 4000}}{}^{-1.5}\right\}$$



Comment bien modifier la valeur du learning rate ?

Échéanciers cycliques

[Smith, 2017] propose une nouvelle manière de définir un échéancier : il sera cyclique.

> *The essence of this learning rate policy comes from the observation that increasing the learning rate might have a short term negative effect and yet achieve a longer term beneficial effect*
>
> — Leslie Smith (2015)

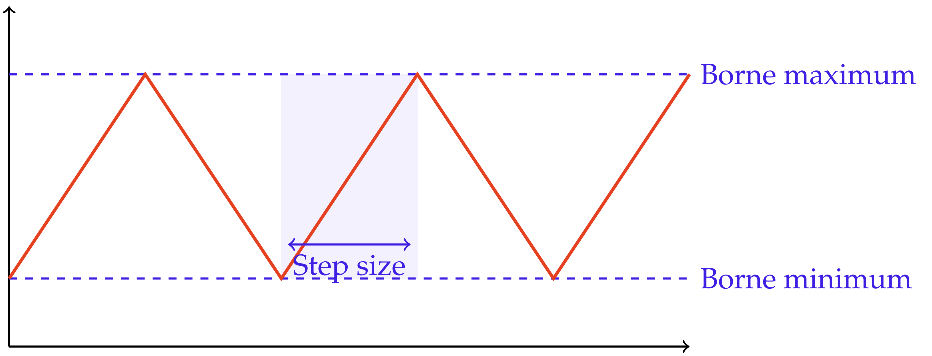


Figure – Échéancier triangulaire pour le learning rate

Comment bien modifier la valeur du learning rate ?
Échéanciers cycliques

Nous pouvons imaginer beaucoup de manières de réaliser ces cycles et on peut rajouter des bornes qui décroissent dans le temps également. Une manière est devenue standard dans tout les modèles fondamentaux : l'échéancier cosinus [Loshchilov and Hutter, 2016].

Nombre d'itérations réalisées dans le cycle i

$$\eta_t = \eta^i_{\min} + \frac{1}{2}\left(\eta^i_{\max} - \eta^i_{\min}\right)\left(1 + \cos\left(\frac{\boxed{T_{\text{cur}}}}{\boxed{T_i}}\pi\right)\right)$$

Nombre d'itérations à réaliser dans le cycle i

Comment bien modifier la valeur du learning rate ?

Échéanciers cosinus

L'exploitation optimale de l'échéancier cosinus a été étudié dans plusieurs articles, citons-en deux :

- [Hoffmann et al., 2022] montre qu'un cycle durant une époque (la totalité de l'entraînement d'un LLM) et réduisant de 10% la valeur maximale semble optimal
- [Popel and Bojar, 2018] montre que la période de warmup est critique : la longueur et le maximum sont très important pour la suite de l'entraînement

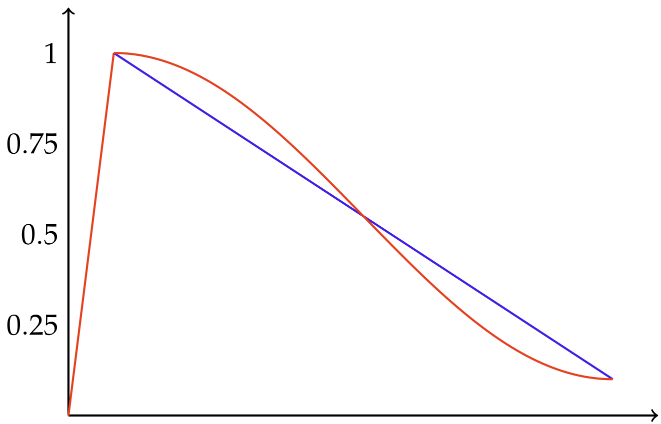


Figure – Rapport maximum LR / LR pour l'échéancier décroissement linéaire et cosinus avec période de chauffe

Quelles sont les bonnes non-linéarité dans un réseau de neurones ?

Quelles sont les bonnes non-linéarité dans un réseau de neurones ?

GELU

[Hendrycks and Gimpel, 2016] introduit une nouvelle fonction d'activation : GELU. La volonté est de combiner les comportements de :

- **ReLU** : multiplier par 0 ou 1 l'input
- **Dropout** : multiplier par 0 ou 1 aléatoirement

Pour le faire, on propose de multiplier l'input par $m \sim \text{Bernoulli}(\phi(x))$ avec $\phi(x) = \mathbb{P}(X \leqslant x)$ et $X \sim \mathcal{N}(0,1)$. Mais ce comportement doit être déterministe pour la phase d'inférence, donc pour approcher au mieux le comportement de cette régularisation :

$$\text{GELU}(x) = x\mathbb{P}(X \leqslant x) = x\phi(x)$$

Notons quelques propriétés :

- GELU est $\mathcal{C}^1$ sur $\mathbb{R}$, en particulier au voisinage de 0 contrairement à ReLU
- GELU n'est pas monotone

QUELLES SONT LES BONNES NON-LINÉARITÉ DANS UN RÉSEAU DE NEURONES ?

SILU

Dans le même article [Hendrycks and Gimpel, 2016] est introduit SiLU comme une variante de GELU où l'on choisit la distribution logistique au lieu de gaussienne :

$$\mathrm{SiLU}(x) = x\sigma(x) \quad \text{avec} \quad \sigma(x) = \frac{1}{1+e^{-x}}$$

On conserve les deux particularité de GELU par rapport à ReLU : non monotonie et $\mathcal{C}^1$ partout. Cela permet d'éviter le phénomène de mort de neurones et une meilleure transmission des informations au voisinage de zéro.

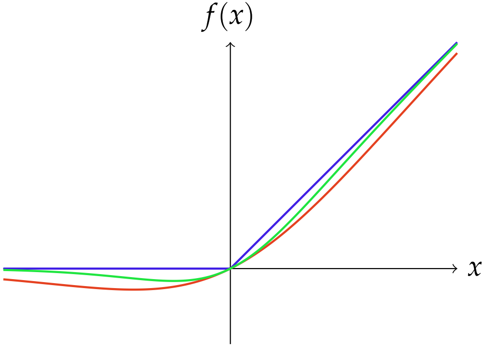


Figure – Fonctions ReLU, SiLU et GELU

QUELLES SONT LES BONNES NON-LINÉARITÉ DANS UN RÉSEAU DE NEURONES ?

CELLULE LSTM

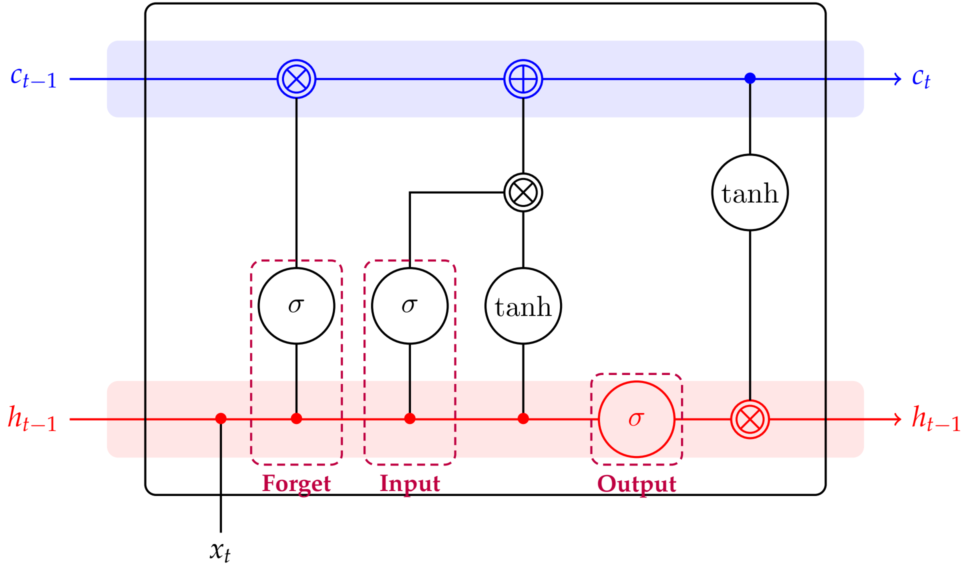


Figure – Schéma d'un LSTM avec état de cellule $(c_t)_{t\in\mathbb{N}}$ et état caché $(h_t)_{t\in\mathbb{N}}$

Quelles sont les bonnes non-linéarité dans un réseau de neurones ?

Gated Linear Unit

[Dauphin et al., 2016] reprend l'idée de LSTM de contrôler à plusieurs niveau la quantité d'informations que l'on transmet en introduisant une nouvelle manière de calculées les couches cachées h_l avec $l \leqslant L$ la l-ième couche cachée du réseau de neurones à L couches :

$$\forall l \leqslant L, \quad h_l(\overset{\text{Matrice input}}{X}) = (X \underbrace{W + b}_{\text{Poids couche 1}}) \otimes \sigma(X \underbrace{V + c}_{\text{Poids couche 2}})$$

On a noté $\otimes$ le produit terme à terme et σ la fonction sigmoid.

Quelles sont les bonnes non-linéarité dans un réseau de neurones ?

Gated Linear Unit

[Shazeer, 2020] pousse la réflexion plus loin et propose de remplacer la fonction σ par d'autres fonctions d'activation classiques. En pratique, les résultats ne sont pas vraiment des fonctions d'activation mais plutôt de nouveaux blocs d'architecture. Parmi celles-ci, notons :

$$\begin{aligned} \mathrm{GEGLU}(X) &= \mathrm{GELU}(XW+b)\otimes(XV+c) \\ \mathrm{SwiGLU}(X) &= \mathrm{SiLU}(XW+b)\otimes(XV+c) \end{aligned}$$

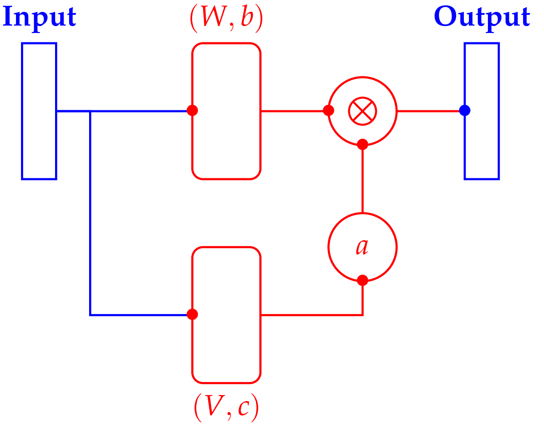


Figure – Architecture GLU avec une fonction d'activation a

Quelles sont les bonnes non-linéarité dans un réseau de neurones ?

Gated Linear Unit

Dans l'architecture Transformers les blocs *position-wise feed-forward networks* (FFN) sont les blocs impactés par cette modification. A noter que de plus en plus de LLM suppriment les vecteurs de biais, pour atteindre les blocs FFN suivants :

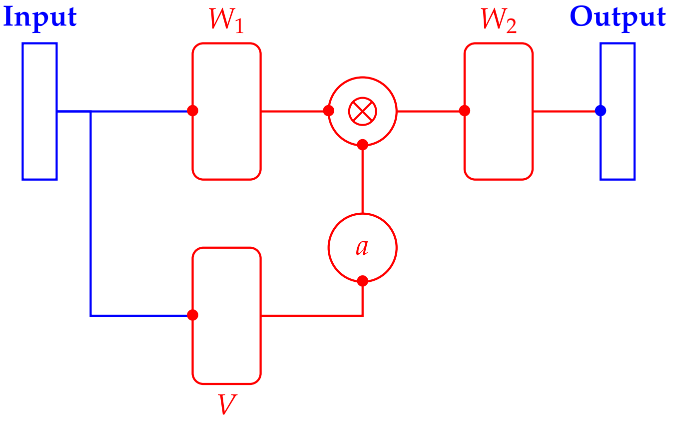


Figure – Architecture FFN avec une fonction d'activation GLU a

Pour conserver un ordre de grandeur similaire en termes de paramètres, il faut réduire le nombre de neurones que l'on place sur chaque couche.

Comment appliquer les avancées du NLP pour la vision ?

Depuis son introduction, l'utilisation des réseaux convolutionnel [LeCun et al., 1998] fait consensus dans tous les domaines traitant avec des images. La structuration autour des datasets commun MNIST et FashionMNIST [LeCun et al., 2010, Xiao et al., 2017] ou la compétition ImageNet qui a été source d'architecture et briques élémentaires des réseaux de neurones, on peut citer :

Depuis son introduction, l'utilisation des réseaux convolutionnel [LeCun et al., 1998] fait consensus dans tous les domaines traitant avec des images. La structuration autour des datasets commun MNIST et FashionMNIST [LeCun et al., 2010, Xiao et al., 2017] ou la compétition ImageNet qui a été source d'architecture et briques élémentaires des réseaux de neurones, on peut citer :

- **AlexNet** [Krizhevsky et al., 2012] : première utilisation des GPU pour l'entraînement, et du Dropout
- **Inception v1** [Szegedy et al., 2015] : introduction du module Inception : il consiste à travailler à plusieurs échelles en parallèle pour avoir une vision locale et plus global en même temps
- **VGG** [Simonyan and Zisserman, 2014] : plutôt que de faire varier la taille des convolutions, il suffit d'en placer plus de petite taille pour réduire la taille du modèle sans changer la capacité de représentation
- **ResNet** [He et al., 2016] : introduction des ResBlock

Comment appliquer les avancées du NLP pour la vision ?

Vision Transformers

Suite à la proposition du mécanisme d'attention, des travaux avec les réseaux convolutionnels ont été mené. On peut citer :

- [Bello et al., 2019] adresse le problème de localité des réseaux convolutionnel en augmentant les features avec le mécanisme d'attention
- [Wu et al., 2020] propose le *Visual Transformer* (VT). Une image est d'abord traité par des couches de convolutions, et ce résultat est placé dans 16 catégories sémantiques, traité par la suite par un transformer

[Dosovitskiy et al., 2020] propose en octobre 2020 les Vision Transformers (ViT) qui s'appuie uniquement sur l'architecture transformers, sans couches de convolution. Cependant, une image n'est pas une séquence de vecteur. Ainsi, l'article propose de brutalement de transformer une image en patch qui ensemble formeront une séquence, donc une entrée pour un transformer.

Exercice 1 (Image en patch)

On considère une image de taille (H, W) pixels. On souhaite obtenir des patchs de taille (P, P).

1. *On note N le nombre de patch. Que vaut N dans notre cas ?*
2. *Chaque patch est aplati (flattened). Quelle est la taille de la séquence total pour l'image de départ ?*
3. *Quelle est la taille de la séquence totale pour une image de taille (H, W, C) où C représente le nombre de canal ?*
4. *Faire l'application numérique pour une image de taille (6, 8, 3) avec $P = 2$*

L'objectif d'aplatir l'image via des patchs est de conserver une idée de *localité* : si l'on aplati l'image brutalement on perd la structure.

Comment appliquer les avancées du NLP pour la vision ?

Vision Transformers

Les blocks Transformers exploitent la fonction d'activation GELU et la couche de Layer Normalization Il y a également plusieurs tailles de modèles qui sont disponible pour pouvoir traiter différents usages. Il est important de préciser que pour obtenir des performances compétitive avec ces réseaux, il semble nécessaire empiriquement, d'après les deux articles, de traiter avec de très large volume de données tant en nombre qu'en classe à prédire.

Modèle	Année	Paramètres (M)	Accuracy (%)
ResNet-101	2015-12	45	78.2
ViT-B/16	2020-10	87	85.4
ViT-L/16	2020-10	305	86.8
Swin-B	2021-08	88	86.4
Swin-L	2021-08	197	87.3

Table – Comparaison des performances d'accuracy sur ImageNet-1K, pré-entraîné sur ImageNet-22K pour des images de taille 384^2

Comment appliquer les avancées du NLP pour la vision ?
ConvNeXt

[Liu et al., 2022] déroule un plan de travail pour moderniser les réseaux convolutionnels en utilisant uniquement des couches de convolutions, mais en se nourrissant des avancées des Transformers. Son point de départ est le ResNet 50 [He et al., 2016]. Les premiers changement, avant de toucher à l'architecture, portent sur les méthodes d'entraînement avec entre autres :

- Entraînement plus long (de 90 époques à 300 époques) avec une data augmentation plus complète
- L'optimiser AdamW est exploité avec un batch size de 4096 au lieu de 256
- L'échéancier du learning rate suit un échéancier cosinus

On obtient un gain de performance de 2.7 points (selon l'article) en *mettant à jour* la procédure d'entraînement.

Comment appliquer les avancées du NLP pour la vision ?

ConvNeXt

Notons quelques-un des autres changement sont présentés :

- **Plus grande taille de filtre** : la taille 3×3 est remplacé par une taille 7×7
- **Moins de fonction d'activation** : une fonction d'activation par bloc permet de gagner 0.7 point
- **Moins de couche de normalisation** : de deux couches de Batch-Normalization à une seule couche de Layer-Normalization : on gagne deux fois 0.1 point de performance

Modèle	Année	Paramètres (M)	Accuracy (%)
ResNet-101	2015-12	45	78.2
ViT-B/16	2020-10	87	85.4
ViT-L/16	2020-10	305	86.8
Swin-B	2021-08	88	86.4
Swin-L	2021-08	197	87.3
ConvNeXt-T	2022-03	29	82.9
ConvNeXt-S	2022-03	50	85.8
ConvNeXt-B	2022-03	89	86.8
ConvNeXt-L	2022-03	198	87.5
ConvNeXt-XL	2022-03	350	87.8

Table – Comparaison des performances d'accuracy sur ImageNet-1K, pré-entraîné sur ImageNet-22K pour des images de taille 384^2

BIBLIOGRAPHIE I

Bello, I., Zoph, B., Vaswani, A., Shlens, J., and Le, Q. V. (2019).
Attention augmented convolutional networks.
In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3286–3295.

Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. (2016).
Language modeling with gated convolutional networks.
In *International conference on machine learning*, pages 933–941. PMLR.

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.

He, K., Zhang, X., Ren, S., and Sun, J. (2016).
Deep residual learning for image recognition.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Hendrycks, D. and Gimpel, K. (2016).
Gaussian error linear units (gelus).
arXiv preprint arXiv :1606.08415.

Bibliographie II

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. (2022).
Training compute-optimal large language models.
arXiv preprint arXiv :2203.15556.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012).
Imagenet classification with deep convolutional neural networks.
Advances in neural information processing systems.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998).
Gradient-based learning applied to document recognition.
Proceedings of the IEEE.

LeCun, Y., Cortes, C., and Burges, C. (2010).
Mnist handwritten digit database.
ATT Labs [Online]. Available : http ://yann.lecun.com/exdb/mnist, 2.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. (2022).
A convnet for the 2020s.
In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986.

Bibliographie III

Loshchilov, I. and Hutter, F. (2016).
Sgdr : Stochastic gradient descent with warm restarts.
arXiv preprint arXiv :1608.03983.

Popel, M. and Bojar, O. (2018).
Training tips for the transformer model.
arXiv preprint arXiv :1804.00247.

Shazeer, N. (2020).
Glu variants improve transformer.
arXiv preprint arXiv :2002.05202.

Simonyan, K. and 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.
In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE.

Bibliographie IV

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015).
Going deeper with convolutions.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017).
Attention is all you need.
Advances in neural information processing systems, 30.

Wu, B., Xu, C., Dai, X., Wan, A., Zhang, P., Yan, Z., Tomizuka, M., Gonzalez, J., Keutzer, K., and Vajda, P. (2020).
Visual transformers : Token-based image representation and processing for computer vision.
arXiv preprint arXiv :2006.03677.

Xiao, H., Rasul, K., and Vollgraf, R. (2017).
Fashion-mnist : a novel image dataset for benchmarking machine learning algorithms.
CoRR.

Annexe : SGD par mini-batch

Afin d'obtenir des convergences plus proche du minimum, nous pouvons considérer n_B observations puis mettre à jour. Si $n_B = 1$ on obtient la descente de gradient stochastique et si $n_B = n$ on retrouve la descente de gradient classique. La descente de gradient par mini-batch est donc un compromis entre les deux descentes présentées.
On note $\mathcal{B}_t$ l'ensemble des index choisit aléatoirement tel que $|\mathcal{B}_t| = n_B$, on a :

$$\theta_{t+1} = \theta_t - \eta_t \left(\frac{1}{n_B} \sum_{i \in \mathcal{B}_t} \nabla \ell_i(\theta_t) \right)$$

(a) Calcul des $-\nabla \ell_i(\theta_t)$ pour un batch $\mathcal{B}_t$

(b) Mise à jour des poids

On conserve les hypothèses (1) que l'on a faite sur la fonction de perte et l'aléatoire de tirage des index dont les deux dernières conditions sont :

- $\nabla \ell_{i_t}(\theta_t)$ est un estimateur non biaisé de $\nabla \mathcal{L}(\theta_t)$
- $\mathbb{E}\left[\|\nabla \ell_{i_t}(\theta_t)\|^2\right] \leqslant \sigma^2 + \|\mathcal{L}(\theta)\|^2$

On obtient le résultat suivant :

Proposition 1 (Variance)

La variance d'une estimation par descente de gradient stochastique par mini-batch utilisant n_B échantillons avec remise vérifie :

$$\mathbb{E}\left[\left\|\frac{1}{n_B}\sum_{i\in\mathcal{B}_t}\nabla \ell_i(\theta_t)\right\|^2\right] \leqslant \frac{\sigma^2}{n_B} + \|\nabla \mathcal{L}(\theta_t)\|^2$$

A l'aide de la proposition 1 nous pouvons déduire un résultat similaire au théorème 1 qui le généralise.

Théorème 3 (Learning rate fixe pour SGD par mini-batch)

On considère une descente de gradient stochastique par mini-batch avec remise de taille n_B avec $\eta_t = \eta$ tel que $\eta \in \left]0, \frac{1}{\beta}\right]$. Alors, pour tout $T \geqslant 1$:

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} \|\nabla \mathcal{L}(\theta_t)\|^2\right] \leqslant \frac{\eta\beta\sigma^2}{n_B} + \frac{2(\mathcal{L}(\theta_0) - \mathcal{L}(\theta^*))}{\eta T}$$

On obtient alors que $\lim\limits_{T\to+\infty} \mathbb{E}\left[\min\limits_{0\leqslant t\leqslant T-1} \|\nabla \mathcal{L}(\theta_t)\|^2\right] \in \left[0, \frac{\eta\beta\sigma^2}{n_B}\right]$.

Il est important de noter que l'on ne traite que du cas où le mini-batch est construit avec remise et que l'on a supposé des indépendances, distributions identiques et des estimations non biaisées. Il est difficile de s'en assurer en pratique.

Annexe : Preuves

Annexe : Preuves

Théorème 1 : Convergence pour SGD avec learning rate fixe

Soit $T \leqslant 1$ et $t \leqslant T$ une étape. L'inéquation 1 se réécrit comme :

$$\begin{aligned}
\mathbb{E}\left[\mathcal{L}\left(\theta_{t+1}\right)-\mathcal{L}\left(\theta_t\right)\right] & \leqslant -\left(\eta-\frac{\eta^2 \beta}{2}\right)\left\|\nabla \mathcal{L}\left(\theta_t\right)\right\|^2+\frac{\eta^2 \beta}{2} \sigma^2 \\
& \leqslant -\frac{\eta}{2}\left\|\nabla \mathcal{L}\left(\theta_t\right)\right\|^2+\frac{\eta^2 \beta}{2} \sigma^2 \quad \text{car } \eta \in \left]0, \frac{1}{\beta}\right]
\end{aligned}$$

Puisque $\mathcal{L}\left(\theta_t\right) \geqslant \mathcal{L}\left(\theta^*\right)$ pour toute étape t, en sommant sur toute les étapes, on a en espérance :

$$\begin{aligned}
\mathcal{L}\left(\theta^*\right)-\mathcal{L}\left(\theta_0\right) \leqslant \mathbb{E}\left[\mathcal{L}\left(\theta_T\right)-\mathcal{L}\left(\theta_0\right)\right] & \leqslant -\frac{\eta}{2} \sum_{t=0}^{T-1} \mathbb{E}\left[\left\|\nabla \mathcal{L}\left(\theta_t\right)\right\|^2\right]+T \frac{\eta^2 \beta}{2} \sigma^2 \\
\mathbb{E}\left[\frac{1}{T} \sum_{t=0}^{T-1}\left\|\nabla \mathcal{L}\left(\theta_t\right)\right\|^2\right] & \leqslant \eta \beta \sigma^2+\frac{2\left(\mathcal{L}\left(\theta_0\right)-\mathcal{L}\left(\theta^*\right)\right)}{\eta T}
\end{aligned}$$

D'où $\lim\limits_{T \rightarrow+\infty} \mathbb{E}\left[\min\limits_{0 \leqslant t \leqslant T-1}\|\nabla \mathcal{L}(\theta_t)\|^2\right] \in\left[0, \eta \beta \sigma^2\right]$

Annexe : Preuves

Proposition 1 : Réduction de variance pour SGD avec mini-batch

On rappelle que l'on a fait les hypothèses suivante sur la constitution du mini-batch pour $i \in \mathcal{B}_t$ à une époque t :

- $\nabla \ell_i(\theta_t)$ est un estimateur non biaisé de $\nabla \mathcal{L}(\theta_t)$
- $\mathbb{E}\left[\|\nabla \ell_i(\theta_t)\|^2\right] \leqslant \sigma^2 + \|\mathcal{L}(\theta)\|^2$

On a supposé de plus qu'il y a indépendance et identique distribution pour les $\nabla \ell_i$. Ainsi, on a :

$$\begin{aligned}
\mathbb{E}\left[\left\|\frac{1}{n_B}\sum_{i\in\mathcal{B}} \nabla \ell_i(\theta_t)\right\|^2\right] - \left\|\mathbb{E}\left[\frac{1}{n_B}\sum_{i\in\mathcal{B}} \nabla \ell_i(\theta_t)\right]\right\|^2 &= \frac{1}{n_B}\left(\mathbb{E}\left[\|\nabla \ell_i(\theta_t)\|^2\right] - \|\mathcal{L}\left(\theta_t\right)\|^2\right) \\
&\leqslant \frac{\sigma^2}{n_B} \quad \text{avec le second point}
\end{aligned}$$

Le premier point nous indique que : $\mathbb{E}\left[\frac{1}{n_B}\sum_{i\in\mathcal{B}} \nabla \ell_i(\theta_t)\right] = \frac{1}{n_B}\sum_{i\in\mathcal{B}} \mathbb{E}\left[\nabla \ell_i(\theta_t)\right] = \nabla \mathcal{L}\left(\theta_t\right)$

On a donc finalement que :

$$\mathbb{E}\left[\left\|\frac{1}{n_B}\sum_{i\in\mathcal{B}} \nabla \ell_i(\theta_t)\right\|^2\right] \leqslant \frac{\sigma^2}{n_B} + \|\mathcal{L}\left(\theta_t\right)\|^2$$

Annexe : Preuves

Théorème 3 : Convergence pour SGD par mini-batch avec learning rate fixe

La démonstration du théorème 3 est similaire à celle du théorème 1 mais il faut prendre en compte le mini-batch. Puisque $\mathcal{L}$ est β-smooth :

$$\begin{aligned}
\mathcal{L}\left(\theta_{t+1}\right) &\leqslant \mathcal{L}\left(\theta_t\right) + \langle \nabla \mathcal{L}\left(\theta_t\right), \theta_{t+1} - \theta_t \rangle + \frac{\beta}{2} \|\theta_{t+1} - \theta_t\|^2 \\
&\leqslant \mathcal{L}\left(\theta_t\right) - \eta_t \langle \nabla \mathcal{L}\left(\theta_t\right), \frac{1}{n_B} \sum_{i \in \mathcal{B}} \nabla \ell_i(\theta_t) \rangle + \frac{\beta \eta^2}{2} \left\| \frac{1}{n_B} \sum_{i \in \mathcal{B}} \nabla \ell_i(\theta_t) \right\|^2
\end{aligned}$$

Ainsi, en espérance sur le choix aléatoire de $\mathcal{B}_t$:

$$\mathbb{E}\left[\mathcal{L}\left(\theta_{t+1}\right)\right] \leqslant \mathcal{L}\left(\theta_t\right) - \eta_t \|\nabla \mathcal{L}\left(\theta_t\right)\|^2 + \frac{\beta \eta_t^2}{2} \mathbb{E}\left[\| \frac{1}{n_B} \sum_{i \in \mathcal{B}_t} \nabla \ell_i(\theta_t) \|^2 \right]$$

A l'aide de la proposition 1 on obtient finalement :

$$\mathbb{E}\left[\mathcal{L}\left(\theta_{t+1}\right)\right] \leqslant \mathcal{L}\left(\theta_t\right) - \left(\eta_t - \frac{\beta \eta^2}{2}\right) \|\nabla \mathcal{L}\left(\theta_t\right)\|^2 + \frac{\beta \eta^2}{2 n_B} \sigma^2 \tag{2}$$

On utilise l'inéquation 2 comme point de départ à la place de l'inéquation 1, puis *mutatis mutandis* dans la démonstration du théorème 3.