ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

ADAM Optimisation

Lisette, Marie, Abigail

Projet d'Optimisation (MA174) - 2021

Résumé

Il est important de optimaliser le network construit dans le domaine de deep learning. Donc le but de ce sujet est de réaliser un algorithme d'optimisation et de vérifier la convergence.Le résultat nous indique que l'ADAM améliore considérablement la performance du CNN

Table des matières

I - Introduction

Ce projet comprend deux parties : partie mathématique et partie informatique. Dans la partie mathématique, nous avons étudié la convergence et la complexité de l'algorithme d'optimisation Adam. Dans la partie informatique, nous avons construit un modèle CNN pour reconnaître l'écriture manuscrite dans les données de MNIST. Nous avons ensuite appliqué l'algorithme ADAM pour diminuer la perte et optimiser la prédiction.

Déclaration de la contribution
Dans ce projet, Marie et Abigaïl sont responsables de la démonstration de la convergence de l'algorithme ADAM et de la rédaction correspondante ; Lisette est responsable de l'étude de la complexité de l'algorithme ADAM, de toute la partie informatique et de la rédaction correspondante. Mais chacun d'entre nous a participé à toutes les deux parties dans une certaine mesure en nous entraidant. Et avec Overleaf, chacun d'entre nous a contribué à la mise en page de cette article.

II - Partie Mathématique

II.A - L'introduction de l'ADAM

La recherche dans le domaine de l'apprentissage profond se compose de deux grandes branches principales.

La première est la conception de modèles, dans le but de réaliser des percées en matière de performances dans un certain scénario d'application, typiquement représenté par Resnet et BERT, qui ont réalisé des percées importantes dans les domaines de la vision par ordinateur et du traitement du langage naturel, respectivement.

Le deuxième est l'algorithme d'optimisation, le but est de calculer les paramètres optimaux du réseau pour toute structure de réseau donnée, le représentant typique est l'algorithme d'Adam, d'après l'expérience de la pratique de l'ingénierie, la vitesse de convergence d'Adam est rapide et peut être généralement applicable à divers réseaux.

Le processus de formation de modèles pour l'apprentissage profond consiste essentiellement à résoudre un problème d'optimisation. Dans nos études, il s'agit de l'algorithme de l'ADAM.

II.A.1 - L'idée sur algorithme de l'optimisation

Le processus de formation d'un modèle pour l'apprentissage profond consiste essentiellement à résoudre un problème d'optimisation :

$$\min_{\boldsymbol{\theta}\in\Theta} \quad f(\boldsymbol{\theta})$$

où f est la fonction de perte à optimiser et donc à minimiser, θ est les paramètres du réseau (à ce stade, la structure du réseau a été déterminée), et est le domaine de définition (ou domaine de restriction) des paramètres du réseau. Dans le scénario spécifique de l'apprentissage profond, nous pouvons étoffer le problème d'optimisation comme suit.

Le domaine de définition Θ n'a généralement pas besoin d'être spécifié, il peut être simplement omis, ou spécifié comme $\mathbb{R}^n$, c'est-à-dire le nombre réel entier dans l'espace dimensionnel de n, où n est le nombre total de paramètres.

f est une fonction de perte basée sur un certain (grand nombre d') ensemble d'échantillons, que nous désignons par $\{\mathbf{x}_k, y_k\}_{k=1}^K$, $\mathbf{x}_k$ est l'entrée de la structure du réseau (qui peut être interprétée comme des caractéristiques), y_k est l'étiquette de cet échantillon (qui peut être un nombre réel, ou une valeur binaire 0-1, une valeur de multi-classification 0-1-2-3, etc.).

Donc la fonction de la perte

$$f = \sum_{k=1}^{K} L\left(g\left(\mathbf{x}_k; \boldsymbol{\theta}\right), y_k\right)$$

est la somme des fonctions de perte individuelles de l'échantillon.

La fonction de perte de l'échantillon calcule la valeur de perte entre la valeur prédite $g\left(\mathbf{x}_k; \boldsymbol{\theta}\right)$ et la valeur réelle y_k. Et $g\left(\mathbf{x}_k; \boldsymbol{\theta}\right)$ dépend des caractéristiques d'entrée $\mathbf{x}_k$,de la structure du réseau (résolution des fonctions) et du paramètre du réseau $\boldsymbol{\theta}$.

Ainsi, le problème d'optimisation de la formation du modèle d'apprentissage profond est spécifié comme suit

$$\min_{\boldsymbol{\theta}} \sum_{k=1}^{K} L\left(g\left(\mathbf{x}_k; \boldsymbol{\theta}\right), y_k\right)$$

Lorsque nous effectuons une certaine classe de tâches d'apprentissage automatique avec une certaine structure de réseau, il suffit de spécifier les expressions pour L et g.

Par exemple, lorsque nous effectuons la tâche de régression logistique de la classification binaire, on a :

$$L(x, y) = -y \log x - (1 - y) \log(1 - x) \text{ et } g\left(\mathbf{x}_k; \boldsymbol{\theta}\right) = \frac{1}{1+\exp(-\langle \mathbf{x}_k, \boldsymbol{\theta} \rangle)} \text{ ...}$$

Ici,$\langle \cdot, \cdot \rangle$ désigne l'opération de produit interne. Dans la régression logistique, le nombre de paramètres est le même que le nombre de caractéristiques, tandis que dans les réseaux neuronaux profonds, le nombre de paramètres est beaucoup plus grand que le nombre de caractéristiques, et l'expressivité du modèle est donc grandement améliorée.

La plupart des recherches actuelles dans le domaine de l'apprentissage profond se sont concentrées sur la conception de modèles, c'est-à-dire la conception de $L(x, y)$ et de $g\left(\mathbf{x}_k; \boldsymbol{\theta}\right)$, la plupart des travaux étant consacrés à l'étude des $g\left(\mathbf{x}_k; \boldsymbol{\theta}\right)$ et relativement peu à l'amélioration des $L(x, y)$. Dans cet article, nous n'aborderons pas la conception du modèle pour le moment, et nous nous concentrerons sur la branche des algorithmes d'optimisation.

II.A.2 - Définition des variables

Ici, on a introduit la quantité de mouvement (momentum) : $\mathbf{g}_t - flche\mathbf{m}^{(t)}$ avec $\mathbf{g}_t = \nabla f_t\left(\boldsymbol{theta}^{(t)}\right)$. Pour le contenant de cet algorithme on a :

$$\begin{array}{ll} \mathbf{m}^{(t)} = \beta_1 \mathbf{m}^{(t-1)} + (1-\beta_1)\,\mathbf{g}_t & \hat{\mathbf{m}}^{(t)} = \mathbf{m}^{(t)} / \left(1-\beta_1^t\right) \\ \mathbf{v}^{(t)} = \beta_2 \mathbf{v}^{(t-1)} + (1-\beta_2)\,\mathbf{g}_t^2 & \hat{\mathbf{v}}^{(t)} = \mathbf{v}^{(t)} / \left(1-\beta_2^t\right) \\ \boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha_t \hat{\mathbf{m}}^{(t)} / \sqrt{\hat{\mathbf{v}}^{(t)}} & \end{array}$$

Avec la condition initiale : $\mathbf{m}^{(0)} = 0$, $\mathbf{v}^{(0)} = \beta_1$ et $^{(0)}$ aléatoire.

II.B - Démonstration de la convergence de l'ADAM

Pour analyser la convergence de ce algorithme, il faut des conditions nécessaires suivantes :

II.B.1 - Condition et la méthode pour étudier la convergence

La convexité de la fonction f étudiée Fonction objectif : pour tout t, $f_t(\boldsymbol{\theta})$ est une fonction convexe sur $\boldsymbol{\theta}$.
C'est-à-dire que, étant donné tout $\boldsymbol{\theta}_1$ $\boldsymbol{\theta}_1$ et $\boldsymbol{\theta}_2$ pour $\forall a \in (0,1)$,on a par première définition de la fonction convexe :

$$f_t\left(a\boldsymbol{\theta}_1 + (1-a)\boldsymbol{\theta}_2\right) \leq a f_t\left(\boldsymbol{\theta}_1\right) + (1-a) f_t\left(\boldsymbol{\theta}_2\right)$$

et par la deuxième définition de la fonction convexe :

$$f_t\left(\boldsymbol{\theta}_2\right) \geq f_t\left(\boldsymbol{\theta}_1\right) + \left\langle \nabla f_t\left(\boldsymbol{\theta}_1\right), \boldsymbol{\theta}_2 - \boldsymbol{\theta}_1 \right\rangle$$

Indicateur de jugement $R(T)/T \to 0$ Lorsque $f_t(\boldsymbol{\theta})$ est une fonction convexe, l'indicateur de jugement est choisi comme étant la statistique $R(T)$ (Regret) :

$$R(T) = \sum_{t=1}^{T} f_t\left(\boldsymbol{\theta}^{(t)}\right) - \min_{\boldsymbol{\theta}} \sum_{t=1}^{T} f_t(\boldsymbol{\theta})$$

Lorsque $T \to \infty$, la valeur moyenne de $R(T)$ vérifie $R(T)/T \to 0$, nous considérons un tel algorithme converge.
C'est-à-dire

$$\boldsymbol{\theta} \to \arg\min_{\boldsymbol{\theta}} \sum_{t=1}^{T} f_t(\boldsymbol{\theta}) \triangleq \boldsymbol{\theta}^{\star}$$

non seulement converge vers une certaine valeur, mais cette valeur aussi minimise la fonction objectif.

Sous la condition de la convergence de l'algorithme , on considère généralement que :

- Plus lentement $R(T)$ croît en fonction de T, l'algorithme converge plus rapidement.
- Au même taux de croissance, plus lentement le taux d'apprentissage décroît , l'algorithme se rétablit plus vite .

Condition sur la borne de la variable $\boldsymbol{\theta}$ On a :

$$\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2 \leq D \text{ , } \forall \boldsymbol{\theta}, \boldsymbol{\theta}' \text{ est bien borné.}$$

ou bien quelque soit la dimension i de la variables $\boldsymbol{\theta}$:

$$\|\boldsymbol{\theta} i - \boldsymbol{\theta}'_i\|_2 \leq D_i, \forall \boldsymbol{\theta}_i, \boldsymbol{\theta}'_i \text{ est bien borné.}$$

Condition sur la borne du gradient $\mathbf{g}_t$ On a :

$$\|\mathbf{g}_t\|_2 \leq G \text{ , } \forall t \text{ est bien borné.}$$

ou bien quelque soit la dimension i du gradient $\mathbf{g}_t$:

$$\|\mathbf{g}_{t,i}\|_2 \leq G_i, \ \forall t \text{ est bien borné.}$$

II.B.2 - Démonstration de la convergence

Généralement, c'est difficile de obtenir $R(T)$ précisément. Donc plus souvent, nous cherchons la borne supérieure de $R(T)$, ensuite on divise cette borne avec T, étudie sa limite tend vers zéro ou pas pour déterminer la convergence.
Commencer par Indicateur de Jugement $R(T)$ Comme on a : $\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \sum_{t=1}^{T} f_t(\boldsymbol{\theta})$ Donc :

$$\begin{aligned} R(T) &= \sum_{t=1}^{T} f_t\left(\boldsymbol{\theta}^{(t)}\right) - \min_{\boldsymbol{\theta}} \sum_{t=1}^{T} f_t(\boldsymbol{\theta}) \\ &= \sum_{t=1}^{T} f_t\left(\boldsymbol{\theta}^{(t)}\right) - \sum_{t=1}^{T} f_t\left(\boldsymbol{\theta}^\star\right) \\ &= \sum_{t=1}^{T} \left[f_t\left(\boldsymbol{\theta}^{(t)}\right) - f_t\left(\boldsymbol{\theta}^\star\right) \right] \end{aligned} \tag{1}$$

De plus, comme $f_t(\Theta)$est une fonction convexe, on a :

$$f_t\left(\boldsymbol{\theta}^\star\right) \geq f_t\left(\boldsymbol{\theta}^{(t)}\right) + \left\langle \mathbf{g}_t, \boldsymbol{\theta}^\star - \boldsymbol{\theta}^{(t)} \right\rangle$$

C'est a dire :

$$f_t\left(\boldsymbol{\theta}^{(t)}\right) - f_t\left(\boldsymbol{\theta}^\star\right) \leq \left\langle \mathbf{g}_t, \boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}^\star \right\rangle$$

En utilisant $R_(T)$ on a :

$$R(T) \leq \sum_{t=1}^{T} \left\langle \mathbf{g}_t, \boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}^\star \right\rangle \tag{2}$$

On décompose l'équation(1) d'après le dimension de variable :

$$\sum_{t=1}^{T} \left\langle \mathbf{g}_t, \boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}^\star \right\rangle = \sum_{t=1}^{T} \sum_{i=1}^{d} g_{t,i} \left(\theta_i^{(t)} - \theta_i^\star \right)$$

En changent l'ordre de la sommation :

$$\sum_{t=1}^{T} \sum_{i=1}^{d} g_{t,i} \left(\theta_i^{(t)} - \theta_i^\star \right) = \sum_{i=1}^{d} \sum_{t=1}^{T} g_{t,i} \left(\theta_i^{(t)} - \theta_i^\star \right)$$

Finalement, on a :

$$R(T) \leq \sum_{i=1}^{d} \sum_{t=1}^{T} g_{t,i} \left(\theta_i^{(t)} - \theta_i^{\star} \right) \tag{3}$$

Construire la relation : De $\boldsymbol{\theta}^{(t)}$ a $\left\langle \mathbf{g}_t, \boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}^{\star} \right\rangle$ On est demande de séparer$g_{t,i} \left(\theta_i^{(t)} - \theta_i^{\star} \right)$. Commencer par l'expression itérative de l'Adam, pour chaque dimension i de la variable :

$$\begin{aligned}
\theta_i^{(t+1)} &= \theta_i^{(t)} - \alpha_t \frac{\hat{m}_i^{(t)}}{\sqrt{\hat{v}_i^{(t)}}} \\
&= \theta_i^{(t)} - \alpha_t \frac{1}{1-\beta_1^t} \frac{m_i^{(t)}}{\sqrt{\hat{v}_i^{(t)}}} \\
&= \theta_i^{(t)} - \alpha_t \frac{1}{1-\beta_1^t} \frac{\beta_1 m_i^{(t-1)} + (1-\beta_1)\, g_{t,i}}{\sqrt{\hat{v}_i^{(t)}}}
\end{aligned}$$

Comme il est difficile de montrer la convergence quand β_1 est constante, donc on laisse $\beta_1 = \beta_{1,t}$,c'est a dire on fait varie β_1 avec l'augmentation du nombre d'itérations. De plus , on laisse $\beta_{1,t}$ décroître pas strictement, c'est a dire $\beta_{1,1} \geq \beta_{1,2} \geq \ldots \geq \beta_{1,t} \geq \ldots$, la quantité de mouvement disparaît peu a peu. Finalement m_i va tendre vers g_i, et $\theta_i^{(t+1)}$ va devenir :

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \alpha_t \frac{1}{1-\prod_{s=1}^{t} \beta_{1,s}} \frac{\beta_{1,t} m_i^{(t-1)} + (1-\beta_{1,t})\, g_{t,i}}{\sqrt{\hat{v}_i^{(t)}}}$$

La variation de θ_i^t avec t est la même, comme la forme est complique, on fait un changement de variable, poser :

$$\gamma_t = \alpha_t \frac{1}{1-\prod_{s=1}^{t} \beta_{1,s}}$$

Alors on a :

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \gamma_t \frac{\beta_{1,t} m_i^{(t-1)} + (1-\beta_{1,t})\, g_{t,i}}{\sqrt{\hat{v}_i^{(t)}}}$$

Ensuit on commence a séparer $g_{t,i} \left(\theta_i^{(t)} - \theta_i^{\star} \right)$:

$$\begin{aligned}
&\theta_i^{(t+1)} = \theta_i^{(t)} - \gamma_t \frac{\beta_{1,t} m_i^{(t-1)} + (1-\beta_{1,t})\, g_{t,i}}{\sqrt{\hat{v}_i^{(t)}}} \\
\Longrightarrow &\left(\theta_i^{(t+1)} - \theta_i^{\star} \right)^2 = \left[\left(\theta_i^{(t)} - \theta_i^{\star} \right) - \gamma_t \frac{\beta_{1,t} m_i^{(t-1)} + (1-\beta_{1,t})\, g_{t,i}}{\sqrt{\hat{v}_i^{(t)}}} \right]^2 \\
\Longrightarrow &2\gamma_t \frac{\beta_{1,t} m_i^{(t-1)} + (1-\beta_{1,t})\, g_{t,i}}{\sqrt{\hat{v}_i^{(t)}}} \left(\theta_i^{(t)} - \theta_i^{\star} \right) \\
&= \left(\theta_i^{(t)} - \theta_i^{\star} \right)^2 - \left(\theta_i^{(t+1)} - \theta_i^{\star} \right)^2 + \gamma_t^2 \frac{\left[\beta_{1,t} m_i^{(t-1)} + (1-\beta_{1,t})\, g_{t,i} \right]^2}{\hat{v}_i^{(t)}}
\end{aligned}$$

Comme $\beta_{1,t} m_i^{(t-1)} + (1-\beta_{1,t})\, g_{t,i} = m_i^{(t)}$ Donc on a :

$$g_{t,i}\left(\theta_i^{(t)} - \theta_i^\star\right) = \underbrace{\frac{\sqrt{\hat{v}_i^{(t)}}\left[\left(\theta_i^{(t)} - \theta_i^\star\right)^2 - \left(\theta_i^{(t+1)} - \theta_i^\star\right)^2\right]}{2\gamma_t\left(1-\beta_{1,t}\right)}}_{(1)}$$

$$\underbrace{-\frac{\beta_{1,t}}{1-\beta_{1,t}} m_i^{(t-1)}\left(\theta_i^{(t)} - \theta_i^\star\right)}_{(2)} + \underbrace{\frac{\gamma_t}{2\left(1-\beta_{1,t}\right)} \frac{\left(m_i^{(t)}\right)^2}{\sqrt{\hat{v}_i^{(t)}}}}_{(3)}$$

En suite on va étudier les trois partie(1),(2),(3) respectivement.
Étudier la borne supérieure de $R(T)$ Dans l'Adam, on peut étudier la borne supérieur de $R(T)$ en se concentrent sur les trois partie(1),(2),(3).

- Pour la partie (1) :

$$\begin{aligned}
&\sum_{t=1}^{T} \frac{\sqrt{\hat{v}_i^{(t)}}\left[\left(\theta_i^{(t)} - \theta_i^\star\right)^2 - \left(\theta_i^{(t+1)} - \theta_i^\star\right)^2\right]}{2\gamma_t\left(1-\beta_{1,t}\right)} \\
=&\sum_{t=1}^{T} \frac{\sqrt{\hat{v}_i^{(t)}}\left[\left(\theta_i^{(t)} - \theta_i^\star\right)^2 - \left(\theta_i^{(t+1)} - \theta_i^\star\right)^2\right]}{2\alpha_t \frac{1}{1-\prod_{s=1}^{t}\beta_{1,s}}\left(1-\beta_{1,t}\right)} \\
=&\sum_{t=1}^{T} \frac{\sqrt{v_i^{(t)}}\left[\left(\theta_i^{(t)} - \theta_i^\star\right)^2 - \left(\theta_i^{(t+1)} - \theta_i^\star\right)^2\right]\left(1-\prod_{s=1}^{t}\beta_{1,s}\right)}{2\alpha_t\left(1-\beta_{1,t}\right)} \\
\leq&\sum_{t=1}^{T} \frac{\sqrt{v_i^{(t)}}\left[\left(\theta_i^{(t)} - \theta_i^\star\right)^2 - \left(\theta_i^{(t+1)} - \theta_i^\star\right)^2\right]}{2\alpha_t\left(1-\beta_{1,1}\right)}
\end{aligned}$$

On fait varie la sommation :

$$\begin{aligned}
&\sum_{t=1}^{T} \frac{\sqrt{\hat{v}_i^{(t)}}\left[\left(\theta_i^{(t)} - \theta_i^\star\right)^2 - \left(\theta_i^{(t+1)} - \theta_i^\star\right)^2\right]}{2\gamma_t\left(1-\beta_{1,t}\right)} \\
\leq&\sum_{t=1}^{T} \frac{\sqrt{\hat{v}_i^{(t)}}\left(\theta_i^{(t)} - \theta_i^\star\right)^2}{2\alpha_t\left(1-\beta_{1,1}\right)} - \frac{\sqrt{\hat{v}_i^{(t)}}\left(\theta_i^{(t+1)} - \theta_i^\star\right)^2}{2\alpha_t\left(1-\beta_{1,1}\right)} \\
=&\frac{\sqrt{\hat{v}_i^{(1)}}\left(\theta_i^{(1)} - \theta_i^\star\right)^2}{2\alpha_1\left(1-\beta_{1,1}\right)} - \frac{\sqrt{\hat{v}_i^{(T)}}\left(\theta_i^{(T+1)} - \theta_i^\star\right)^2}{2\alpha_T\left(1-\beta_{1,1}\right)} \\
&+\sum_{t=2}^{T}\left(\theta_i^{(t)} - \theta_i^\star\right)^2 \cdot \left[\frac{\sqrt{\hat{v}_i^{(t)}}}{2\alpha_t\left(1-\beta_{1,1}\right)} - \frac{\sqrt{\hat{v}_i^{(t-1)}}}{2\alpha_{t-1}\left(1-\beta_{1,1}\right)}\right]
\end{aligned}$$

Remarque : Il faut se concerner sur la troisième partie du formule précédent :

$$\sum_{t=2}^{T}\left(\theta_i^{(t)}-\theta_i^{\star}\right)^2\cdot\left[\frac{\sqrt{\hat{v}_i^{(t)}}}{2\alpha_t\left(1-\beta_{1,1}\right)}-\frac{\sqrt{\hat{v}_i^{(t-1)}}}{2\alpha_{t-1}\left(1-\beta_{1,1}\right)}\right]$$

Ce n'est que pour tout t,$\frac{\sqrt{\hat{v}_i^{(t)}}}{\alpha_t}\geq\frac{\sqrt{\hat{v}_i^{(t-1)}}}{\alpha_{t-1}}$ on peut avoir l'inégalité suivent bien définie :

$$\begin{aligned}&\sum_{t=2}^{T}\left(\theta_i^{(t)}-\theta_i^{\star}\right)^2\cdot\left[\frac{\sqrt{\hat{v}_i^{(t)}}}{2\alpha_t\left(1-\beta_{1,1}\right)}-\frac{\sqrt{\hat{v}_i^{(t-1)}}}{2\alpha_{t-1}\left(1-\beta_{1,1}\right)}\right]\\ \leq&\sum_{t=2}^{T}D_i^2\cdot\left[\frac{\sqrt{\hat{v}_i^{(t)}}}{2\alpha_t\left(1-\beta_{1,1}\right)}-\frac{\sqrt{\hat{v}_i^{(t-1)}}}{2\alpha_{t-1}\left(1-\beta_{1,1}\right)}\right]\\ =&D_i^2\left[\frac{\sqrt{\hat{v}_i^{(T)}}}{2\alpha_T\left(1-\beta_{1,1}\right)}-\frac{\sqrt{\hat{v}_i^{(1)}}}{2\alpha_1\left(1-\beta_{1,1}\right)}\right]\end{aligned}$$

Mais l'hypothèse n'est pas toujours valide pour l'Adam, donc on a posé Amsgrad pour combler ce lacune.
Comme :

$$-\frac{\sqrt{\hat{v}_i^{(T)}}\left(\theta_i^{(T+1)}-\theta_i^{\star}\right)^2}{2\alpha_T\left(1-\beta_{1,1}\right)}\leq 0$$

et :

$$\frac{\sqrt{\hat{v}_i^{(1)}}\left(\theta_i^{(1)}-\theta_i^{\star}\right)^2}{2\alpha_1\left(1-\beta_{1,1}\right)}\leq\frac{D_i^2\sqrt{\hat{v}_i^{(1)}}}{2\alpha_1\left(1-\beta_{1,1}\right)}$$

Donc on a :

$$\begin{aligned}&\sum_{t=1}^{T}\frac{\sqrt{\hat{v}_i^{(t)}}\left[\left(\theta_i^{(t)}-\theta_i^{\star}\right)^2-\left(\theta_i^{(t+1)}-\theta_i^{\star}\right)^2\right]}{2\gamma_t\left(1-\beta_{1,t}\right)}\\ \leq&\left[\frac{D_i^2\sqrt{\hat{v}_i^{(T)}}}{2\alpha_T\left(1-\beta_{1,1}\right)}-\frac{D_i^2\sqrt{\hat{v}_i^{(1)}}}{2\alpha_1\left(1-\beta_{1,1}\right)}\right]+\frac{D_i^2\sqrt{\hat{v}_i^{(1)}}}{2\alpha_1\left(1-\beta_{1,1}\right)}\\ \leq&\frac{D_i^2\sqrt{\hat{v}_i^{(T)}}}{2\alpha_T\left(1-\beta_{1,1}\right)}\end{aligned}$$

De plus, comme $v_i^{(t)}=(1-\beta_2)\sum_{s=1}^{t}\beta_2^{t-s}g_{s,i}^2$ En utilisant la condition sur la borne du gradient :

$$\left.\begin{array}{l}v_i^{(t)}\leq\\ \hat{v}_i^{(t)}=\end{array}\right\}\frac{v_i^{(t)}}{1-\beta_2^t}\leq\frac{(1-\beta_2)\sum_{s=1}^{t}\beta_2^{t-s}G_i^2}{1-\beta_2^t}=\frac{G_i^2\left(1-\beta_2^t\right)}{1-\beta_2^t}=G_i^2$$

On peut avoir finalement pour la partie (1) :

$$\begin{aligned}&\textstyle\sum_{t=1}^{T}\frac{\sqrt{\hat{v}_i^{(t)}}\left[\left(\theta_i^{(t)}-\theta_i^{\star}\right)^2-\left(\theta_i^{(t+1)}-\theta_i^{\star}\right)^2\right]}{2\gamma_t\left(1-\beta_{1,t}\right)}\\ &\leq\frac{D_i^2\sqrt{\hat{v}_i^{(T)}}}{2\alpha_T\left(1-\beta_{1,1}\right)}\leq\frac{D_i^2G_i}{2\alpha_T\left(1-\beta_{1,1}\right)}\end{aligned}\tag{4}$$

on conserve α_T pour designer le taux d'apprentissage optimal dans la suite

- Pour la partie(2) :
  En utilisant la condition sur la borne du gradient, on a :

$$\begin{aligned}\sum_{t=1}^{T} -\frac{\beta_{1,t}}{1-\beta_{1,t}} m_i^{(t-1)} \left(\theta_i^{(t)} - \theta_i^{\star}\right) &= \sum_{t=1}^{T} \frac{\beta_{1,t}}{1-\beta_{1,t}} m_i^{(t-1)} \left[-\left(\theta_i^{(t)} - \theta_i^{\star}\right)\right] \\ &\leq \sum_{t=1}^{T} \frac{\beta_{1,t}}{1-\beta_{1,t}} \left|m_i^{(t-1)}\right| D_i\end{aligned}$$

  Et puis on concerne sur $m_i^{(t-1)}$:

$$\begin{aligned}m_i^{(t)} &= \beta_{1,t} m_{i,t-1,t-1,t-1,t-1,t}^{(t-1)} + (1-\beta_{1,t})\, g_{t,i} \\ &= \beta_{1,t}\beta_{1,t-1} m_i^{(t-2)} + \beta_{1,t}\,(1-\beta_{1,t-1})\, g_{t-1,i} + (1-\beta_{1,t})\, g_{t,i} \\ &= \beta_{1,t}\beta_{1,t-1}\beta_{1,t-2} m_i^{(t-3)} + \beta_{1,t}\beta_{1,t-1}\,(1-\beta_{1,t-2})\, g_{t-2,i} \\ &\quad + \beta_{1,t}\,(1-\beta_{1,t-1})\, g_{t-1,i} + (1-\beta_{1,t})\, g_{t,i} \\ &= \beta_{1,t}\beta_{1,t-1}\cdots\beta_{1,1} m_i^{(0)} + \beta_{1,t}\beta_{1,t-1}\cdots(1-\beta_{1,1})\, g_{1,i} + \cdots \\ &\quad + \beta_{1,t}\beta_{1,t-1}\,(1-\beta_{1,t-2})\, g_{t-2,i} + \beta_{1,t}\,(1-\beta_{1,t-1})\, g_{t-1,i} + (1-\beta_{1,t})\, g_{t,i} \\ &\overset{(a)}{=} \beta_{1,t}\beta_{1,t-1}\cdots(1-\beta_{1,1})\, g_{1,i} + \cdots + \beta_{1,t}\beta_{1,t-1}\,(1-\beta_{1,t-2})\, g_{t-2,i} \\ &\quad + \beta_{1,t}\,(1-\beta_{1,t-1})\, g_{t-1,i} + (1-\beta_{1,t})\, g_{t,i} \\ &= \sum_{s=1}^{t} (1-\beta_{1,s}) \left(\prod_{k=s+1}^{t} \beta_{1,k}\right) g_{s,i}\end{aligned}$$

  Maintenant on utiliser la condition sur la borne du gradient, on a pour tout t :

$$\begin{aligned}\left|m_i^{(t)}\right| &\leq \sum_{s=1}^{t} (1-\beta_{1,s}) \left(\prod_{r=s+1}^{t} \beta_{1,r}\right) |g_{s,i}| \\ &\leq \sum_{s=1}^{t} (1-\beta_{1,s}) \left(\prod_{r=s+1}^{t} \beta_{1,r}\right) G_i = G_i \left(1 - \prod_{s=1}^{t} \beta_{1,s}\right) \leq G_i\end{aligned}$$

  Donc finalement on a pout la partie(2) :

$$\begin{aligned}&\sum_{t=1}^{T} -\frac{\beta_{1,t}}{1-\beta_{1,t}} m_i^{(t-1)} \left(\theta_i^{(t)} - \theta_i^{\star}\right) \\ \leq &\sum_{t=1}^{T} \frac{\beta_{1,t}}{1-\beta_{1,t}} G_i D_i \\ = &G_i D_i \sum_{t=1}^{T} \frac{\beta_{1,t}}{1-\beta_{1,t}}\end{aligned} \tag{5}$$

- Pour la partie(3) :
  On se concerne sur

$$\frac{\left(m_i^{(t)}\right)^2}{\sqrt{\hat{v}_i^{(t)}}} = \sqrt{1-\beta_2^t} \frac{\left(m_i^{(t)}\right)^2}{\sqrt{v_i^{(t)}}} \leq \frac{\left(m_i^{(t)}\right)^2}{\sqrt{v_i^{(t)}}}$$

En considerant la forme de $m_i^{(t)}$ et de $v_i^{(t)}$ on peut transformer$\left(m_i^{(t)}\right)^2$ comme le suivant :

$$\left(m_i^{(t)}\right)^2 = \left(\sum_{s=1}^{t} \frac{(1-\beta_{1,s})\left(\prod_{r=s+1}^{t}\beta_{1,r}\right)}{\sqrt{(1-\beta_2)\,\beta_2^{t-s}}} \cdot \sqrt{(1-\beta_2)\,\beta_2^{t-s}}\, g_{s,i}\right)^2$$

Cela favorise l'application de l'inégalité de Cauchy :

$$\begin{aligned}
\left(m_i^{(t)}\right)^2 &= \left(\textstyle\sum_{s=1}^{t} \frac{(1-\beta_{1,s})\left(\prod_{r=s+1}^{t}\beta_{1,r}\right)}{\sqrt{(1-\beta_2)\beta_2^{t-s}}} \cdot \sqrt{(1-\beta_2)\,\beta_2^{t-s}}\, g_{s,i}\right)^2 \\
&\leq \textstyle\sum_{s=1}^{t} \left(\frac{(1-\beta_{1,s})\left(\prod_{r=s+1}^{t}\beta_{1,r}\right)}{\sqrt{(1-\beta_2)\beta_2^{t-s}}}\right)^2 \cdot \sum_{s=1}^{t} \left(\sqrt{(1-\beta_2)\,\beta_2^{t-s}}\, g_{s,i}\right)^2 \\
&= \textstyle\sum_{s=1}^{t} \frac{(1-\beta_{1,s})^2\left(\prod_{r=s+1}^{t}\beta_{1,r}\right)^2}{(1-\beta_2)\beta_2^{t-s}} \cdot \underbrace{\sum_{s=1}^{t} (1-\beta_2)\,\beta_2^{t-s} g_{s,i}^2}_{v_i^{(t)}}
\end{aligned}$$

De plus comme $v_i^{(t)}$ est borne, on a finalement pour la partie(3) :

$$\begin{aligned}
&\sum_{t=1}^{T} \frac{\gamma_t}{2\,(1-\beta_{1,t})} \frac{\left(m_i^{(t)}\right)^2}{\sqrt{\hat{v}_i^{(t)}}} \leq \sum_{t=1}^{T} \frac{\gamma_t}{2\,(1-\beta_{1,t})} \\
&\sum_{s=1}^{t} \frac{(1-\beta_{1,s})^2\left(\prod_{r=s+1}^{t}\beta_{1,r}\right)^2}{(1-\beta_2)\,\beta_2^{t-s}} \cdot \frac{\sum_{s=1}^{t}(1-\beta_2)\,\beta_2^{t-s} g_{s,i}^2}{\sqrt{\sum_{s=1}^{t}(1-\beta_2)\,\beta_2^{t-s} g_{s,i}^2}} \\
&= \sum_{t=1}^{T} \frac{\gamma_t}{2\,(1-\beta_{1,t})} \sum_{s=1}^{t} \frac{(1-\beta_{1,s})^2\left(\prod_{r=s+1}^{t}\beta_{1,r}\right)^2}{(1-\beta_2)\,\beta_2^{t-s}} \sqrt{v_i^{(t)}} \\
&\leq \sum_{t=1}^{T} \frac{\gamma_t}{2\,(1-\beta_{1,t})} \cdot \sum_{s=1}^{t} \frac{(1-\beta_{1,s})^2\left(\prod_{r=s+1}^{t}\beta_{1,r}\right)^2}{(1-\beta_2)\,\beta_2^{t-s}} \cdot G_i
\end{aligned} \tag{6}$$

Finalement d'après l'equations (4),(5),(6),on a pour la borne superieur de $R(T)$:

$$\begin{aligned}
R(T) \leq & \sum_{i=1}^{d} \frac{D_i^2 G_i}{2\alpha_T\,(1-\beta_{1,1})} + \sum_{i=1}^{d} G_i D_i \sum_{t=1}^{T} \frac{\beta_{1,t}}{1-\beta_{1,t}} \\
& + \sum_{i=1}^{d} G_i \sum_{t=1}^{T} \frac{\gamma_t}{2\,(1-\beta_{1,t})} \cdot \sum_{s=1}^{t} \frac{(1-\beta_{1,s})^2\left(\prod_{r=s+1}^{t}\beta_{1,r}\right)^2}{(1-\beta_2)\,\beta_2^{t-s}} \\
= & \frac{\sum_{i=1}^{d} D_i^2 G_i}{2\alpha_T\,(1-\beta_{1,1})} + \left(\sum_{i=1}^{d} G_i D_i\right)\left(\sum_{t=1}^{T} \frac{\beta_{1,t}}{1-\beta_{1,t}}\right) + \left(\sum_{i=1}^{d} G_i\right) \cdot \\
& \left[\sum_{t=1}^{T} \frac{\gamma_t}{2\,(1-\beta_{1,t})} \cdot \sum_{s=1}^{t} \frac{(1-\beta_{1,s})^2\left(\prod_{r=s+1}^{t}\beta_{1,r}\right)^2}{(1-\beta_2)\,\beta_2^{t-s}}\right]
\end{aligned} \tag{7}$$

Designer les hyper-paramètres optimaux Dans cette partie on concerne sur la choix des hyper-paramètres$\{\gamma_t\}, \{\beta_{1,t}\}, \beta_2$ Tout a bord on a :

- $\beta_{1,t} \in (0,1), \beta_{1,1} \geq \beta_{1,2} \geq \ldots \geq \beta_{1,T} \geq \ldots, \forall t$
- $\beta_2 \in (0,1), \quad \frac{\beta_{1,t}}{\sqrt{\beta_2}} \leq \sqrt{c} < 1, \quad \forall t$

Donc on a pour l'équation (4) :

$$\frac{\sum_{i=1}^{d} D_i^2 G_i}{2\alpha_T (1-\beta_{1,1})}$$

Pour l'équation (5) :

$$\left(\sum_{i=1}^{d} G_i D_i\right)\left(\sum_{t=1}^{T} \frac{\beta_{1,t}}{1-\beta_{1,t}}\right) \leq \left(\sum_{i=1}^{d} G_i D_i\right)\left(\frac{1}{1-\beta_{1,1}} \sum_{t=1}^{T} \beta_{1,t}\right)$$

Pour l'equation (6) :

$$\begin{aligned}
&\left(\sum_{i=1}^{d} G_i\right) \cdot \sum_{t=1}^{T} \frac{\gamma_t}{2(1-\beta_{1,t})} \cdot \sum_{s=1}^{t} \frac{(1-\beta_{1,s})^2 \left(\prod_{r=s+1}^{t} \beta_{1,r}\right)^2}{(1-\beta_2)\beta_2^{t-s}} \\
&= \left(\sum_{i=1}^{d} G_i\right) \cdot \sum_{t=1}^{T} \frac{\alpha_t}{2(1-\beta_{1,t})\left(1-\prod_{s=1}^{t} \beta_{1,s}\right)} \cdot \sum_{s=1}^{t} \left(\frac{1-\beta_{1,s}}{\sqrt{1-\beta_2}}\right)^2 \prod_{r=s+1}^{t} \left(\frac{\beta_{1,r}}{\sqrt{\beta_2}}\right)^2 \\
&\overset{(a)}{\leq} \left(\sum_{i=1}^{d} G_i\right) \cdot \sum_{t=1}^{T} \frac{\alpha_t}{2(1-\beta_{1,1})^2(1-\beta_2)} \sum_{s=1}^{t} \prod_{r=s+1}^{t} c \\
&= \left(\sum_{i=1}^{d} G_i\right) \cdot \sum_{t=1}^{T} \frac{\alpha_t}{2(1-\beta_{1,1})^2(1-\beta_2)(1-c)} = \left(\sum_{i=1}^{d} G_i\right) \cdot \frac{\sum_{t=1}^{T} \alpha_t}{2(1-\beta_{1,1})^2(1-\beta_2)(1-c)}
\end{aligned}$$

On trouve que l'ordre de grandeur ne dépends que de $\frac{1}{\alpha_T} \cdot \sum_{t=1}^{T} \beta_{1,t} \cdot \sum_{t=1}^{T} \alpha_t$, donc d'après la conclusion de SGD : si $\alpha_t = C/t^p$, alors $\frac{1}{\alpha_T} = \mathcal{O}(T^p), \sum_{t=1}^{T} \alpha_t = \mathcal{O}(T^{1-p})$. Quand $p = 1/2$, on peut atteint la borne supérieur optimal $\mathcal{O}\left(T^{1/2}\right)$, cela implique que la borne supérieure optimale de $R(T)$ est supérieur a $\mathcal{O}\left(T^{1/2}\right)$. Si on veut $\beta_{1,t}$ décroît le plus lentement possible, on peut poser :

$$\beta_{1,t} = \frac{\beta_1}{\sqrt{t}}$$

Alors on a $\sum_{t=1}^{T} \beta_{1,t} = \mathcal{O}\left(T^{1/2}\right)$, l'ordre du grandeur de $R(T)$ maintien au niveau $\mathcal{O}\left(T^{1/2}\right)$.

II.B.3 - Conclusion sur la démonstration

Quand la fonction objectif$f_t(\theta)$ est convexe pour quelconque t, l'ordre du grandeur de la borne supérieur optimale est $\mathcal{O}\left(T^{1/2}\right)$, donc l'algorithme de l'ADAM converge.

II.C - Étude de la complexité

Définition de la complexité

Nombre de parcours des sous-routines nécessaires pour résoudre le problème P avec précision ϵ.

Analyse de la complexité

D'après la démonstration sur la convergence de l'algorithme ADAM (cf.II.C), nous obtenons

$$\frac{R(T)}{T} \leqslant \mathcal{O}\left(T^{-1/2}\right)$$

Posons $\epsilon = \mathcal{O}\left(T^{-1/2}\right)$,
Il existe donc $k \in \mathcal{R}_+$, tel que $\epsilon \leqslant k\left(T^{-1/2}\right)$.
Ainsi,

$$T \leqslant \frac{k^2}{\epsilon^2} \quad \Rightarrow \quad T = \mathcal{O}\left(\frac{1}{\epsilon^2}\right)$$

Or, pour une taille de données n, on doit appliquer l'algorithme n fois, donc

$$T' = \mathcal{O}\left(\frac{n}{\epsilon^2}\right)$$

De plus, pour les données de p variables(ex. les données MNIST, toutes les figures comprennet p=28×28 variables), l'agorithme doit être appliqué p fois
Ainsi,

$$T'' = \mathcal{O}\left(\frac{np}{\epsilon^2}\right)$$

La complexité de cette algorithem est donc de l'ordre de $\frac{np}{\epsilon^2}$ qui dépend de la taille des données, du nombre des variables et de la précision exigée.

III - Partie Informatique

Nous avons obtenu 99.02% de précision avec ce CNN formé pour 10 fois pendant environ 20 minutes sur un Macbook Pro 2.3 GHz Intel Core i5.

III.A - Modélisation

III.A.1 - Introduction

Il s'agit d'un réseau de neurones convolutif séquentiel à 5 couches pour la reconnaissance de chiffres formé sur le jeu de données MNIST. Nous avons choisi de le construire avec l'API keras (backend Tensorflow) qui est très intuitif. Tout d'abord, nous avons préparé les données (images de chiffres manuscrits) puis nous nous sommes concentrés sur la modélisation et l'évaluation du CNN.

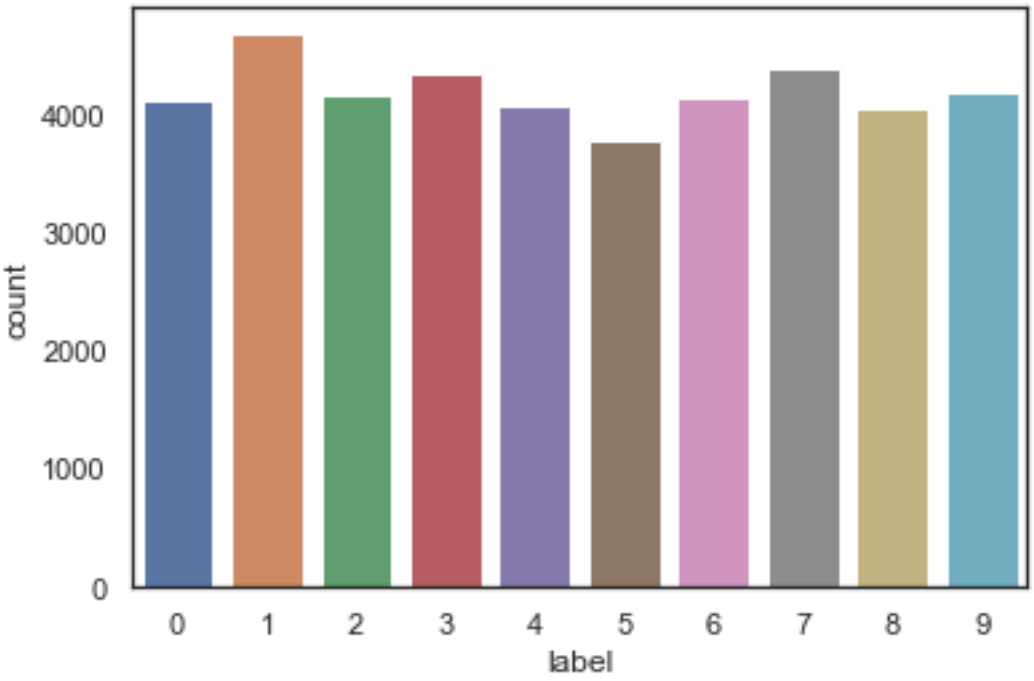

FIGURE 1 – Distribution des chiffres dans les données MNIST

III.A.2 - Préparation des données

- Nous effectuons une normalisation des niveaux de gris pour réduire l'effet des différences d'illumination. De plus, le CNN converge plus rapidement sur les données [0..1] que sur [0..255].
- Les images de formation et de test (28px x 28px) ont été stockées dans pandas.Dataframe en tant que vecteurs 1D de 784 valeurs. Nous remodelons toutes les données en matrices 3D 28x28x1. Keras exige une dimension supplémentaire à la fin qui correspond aux canaux. Les images MNIST sont en échelle de gris et n'utilisent donc qu'un seul canal.

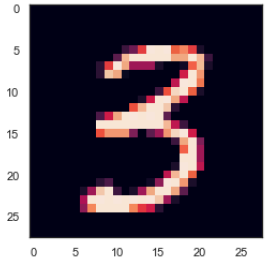

FIGURE 2 – Exemple d'un chiffre dans les données MNIST

- Les étiquettes sont des nombres à 10 chiffres de 0 à 9. Nous devons encoder ces étiquettes dans des vecteurs (one-hot vertor en anglais ex : 2 -> [0,0,1,0,0,0,0,0,0,0]).

- Nous avons choisi de diviser l'ensemble d'entraînement en deux parties : une petite fraction (10%) devient l'ensemble de validation sur lequel le modèle est évalué et le reste (90%) est utilisé pour entraîner le modèle. Puisque nous avons 42 000 images d'entraînement avec des étiquettes équilibrées (one-hot vector), une division aléatoire de l'ensemble d'entraînement n'entraîne pas une surreprésentation de certaines étiquettes dans l'ensemble de validation.

III.A.3 - Modèle CNN

Nous avons utilisé l'API Keras Sequential, où il suffit d'ajouter une couche à la fois, en partant de l'entrée.

La première est la couche convolutionnelle (Conv2D). C'est comme un ensemble de filtres apprenables. Nous avons choisi de mettre 32 filtres pour les deux premières couches Conv2D et 64 filtres pour les deux dernières. Chaque filtre transforme une partie de l'image en utilisant le filtre noyau. La matrice du filtre noyau est appliquée sur l'image entière. Les filtres peuvent être vus comme une transformation de l'image.

La deuxième couche importante du CNN est la couche de pooling (MaxPool2D). Cette couche agit simplement comme un filtre de sous-échantillonnage. Elle examine les deux pixels voisins et choisit la valeur maximale. Ces filtres sont utilisés pour réduire le coût de calcul et, dans une certaine mesure, pour réduire l'overfitting. Nous devons choisir la taille de la mise en commun (c'est-à-dire la taille de la zone mise en commun à chaque fois). Plus la dimension de la mise en commun est élevée, plus le sous-échantillonnage est important.

En combinant les couches de convolution et de mise en commun, les CNN sont capables de combiner les caractéristiques locales et d'apprendre des caractéristiques plus globales de l'image.

Le Dropout est une méthode de régularisation, où une proportion de nœuds dans la couche est aléatoirement ignorée (en mettant leurs valeurs à zéro) pour chaque échantillon de formation. Cela élimine aléatoirement une partie du réseau et force le réseau à apprendre les caractéristiques d'une manière distribuée. Cette technique améliore également la généralisation et réduit l'overfitting.

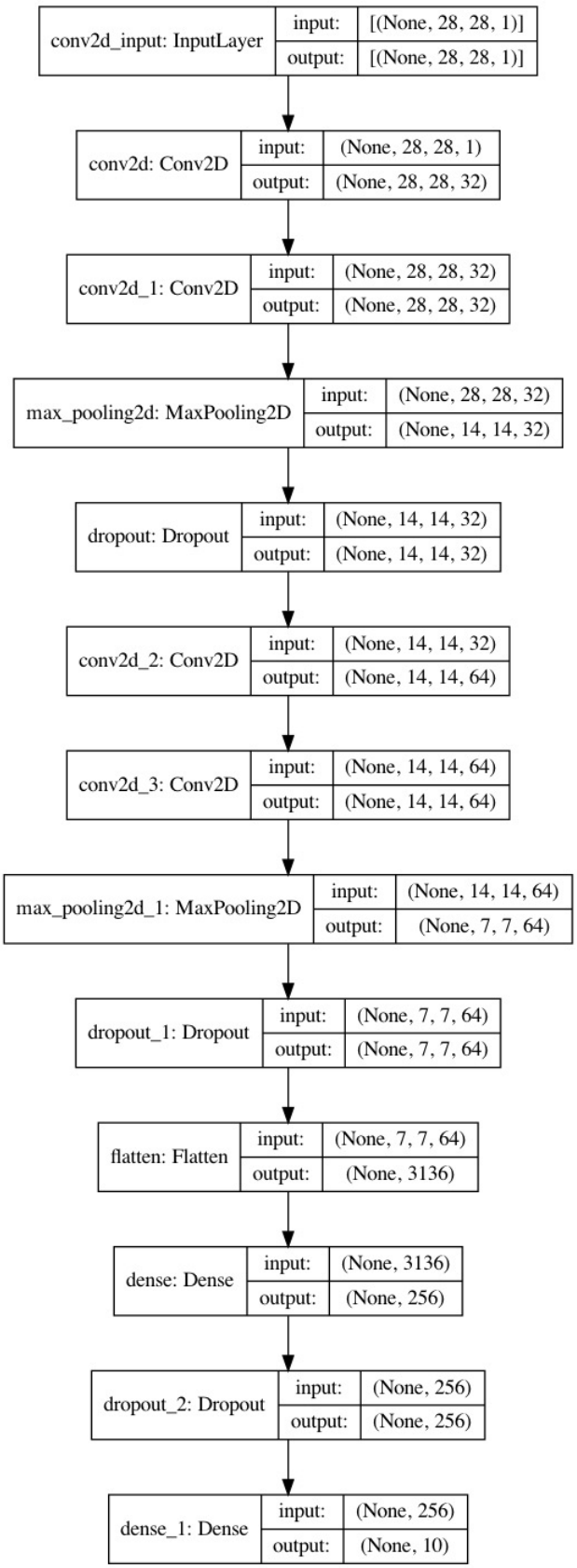

FIGURE 3 – Représentation du modèle CNN

La couche d'aplatissement est utilisée pour convertir les cartes de caractéristiques finales en un seul vecteur 1D. Cette étape d'aplatissement est nécessaire pour que nous puissions utiliser des couches entièrement connectées après des couches convolutionnelles/maxpool. Elle combine toutes les caractéristiques locales trouvées dans les couches convolutionnelles précédentes.
A la fin, nous avons utilisé les caractéristiques dans deux couches entièrement connectées (Dense) qui sont juste des classificateurs de réseaux neuronaux artificiels (ANN). Dans la dernière couche (Dense(10,activation="sigmoid")) le réseau sort la distribution de probabilité de chaque classe.

III.B - Configuration de l'optimiseur et du recycleur

III.B.1 - Fonction de perte

Nous définissons la fonction de perte pour mesurer les mauvaises performances de notre modèle sur des images avec des étiquettes connues. Il s'agit du taux d'erreur entre les étiquettes observées et les étiquettes prédites. Comme chacune des multiples catégories correspond à une probabilité, nous utilisons donc une forme spécifique pour les classifications catégorielles (>2 classes) appelée "categorical_crossentropy", de sorte que la formule de la fonction de perte d'entropie croisée pour les k catégories est la suivante.

$$l = -\sum_{i=1}^{K} y_i \log(p_i)$$

Où.

- K est le nombre de catégories, ici K=10
- y est le label, c'est-à-dire que si la catégorie est i, alors $y_i = 1$, sinon il est égal à 0
- p est la sortie du réseau neuronal, c'est-à-dire la probabilité que la catégorie soit i.

S'il y a plusieurs échantillons, l'entropie croisée moyenne de tous les échantillons est

$$l = -\frac{1}{N} \sum_{n} \sum_{i} y_{i,n} \log(p_{i,n})$$

où,

- N est le nombres total d'échantillons
- $n \in \{1, 2, ..., N\}$

La probabilité de chaque classe dans la multi-classification est ensuite obtenue en utilisant la fonction d'activation softmax. Maintenant que la fonction de perte pour la multi-classification est obtenue, l'étape suivante consiste à utiliser l'algorithme de descente de gradient pour optimiser les paramètres du réseau de manière à minimiser la fonction de perte.

III.B.2 - Optimiseur ADAM

La fonction la plus importante est l'optimiseur. Cette fonction va améliorer itérativement les paramètres (valeurs des noyaux de filtres, poids et biais des neurones ...) afin de minimiser la perte. Et nous avons conçu nous-même l'optimiseur en appliquant l'agorithme ADAM dans python pour l'amélioration de notre modèle. Afin de faire converger l'optimiseur plus rapidement et plus près du minimum global de la fonction de perte, nous avons utilisé une méthode de recuit du taux d'apprentissage.

Le taux d'apprentissage est le pas par lequel l'optimiseur parcourt le "paysage des pertes". Plus le taux d'apprentissage est élevé, plus les pas sont grands et plus la convergence est rapide. Cependant, l'échantillonnage est très faible avec un taux d'apprentissage élevé et l'optimiseur pourrait probablement tomber dans un minimum local.

III.B.3 - Amélioration par recycleur

Il est préférable d'avoir un taux d'apprentissage décroissant pendant l'apprentissage pour atteindre efficacement le minimum global de la fonction de perte.
Pour garder l'avantage d'un temps de calcul rapide avec un taux d'apprentissage élevé, j'ai diminué le taux d'apprentissage dynamiquement tous les X pas (époques) selon que cela est nécessaire (quand la précision n'est pas améliorée).
Avec la fonction ReduceLROnPlateau de Keras.callbacks, nous choisissons de réduire le taux d'apprentissage de moitié si la précision n'est pas améliorée après 3 époques.

III.C - Évaluation du modèle et de l'algorithme

III.C.1 - Validaiton

Nous utilisons la matrice de confusion pour valider notre modèle, ici l'axe horizontal correspond à l'étiquette de prédiction et l'axe vertical correspond à la vraie étiquette. Le nombre situé sur la position (i,j) de la matrice est le nombre de chiffre i en étant prédit comme j.

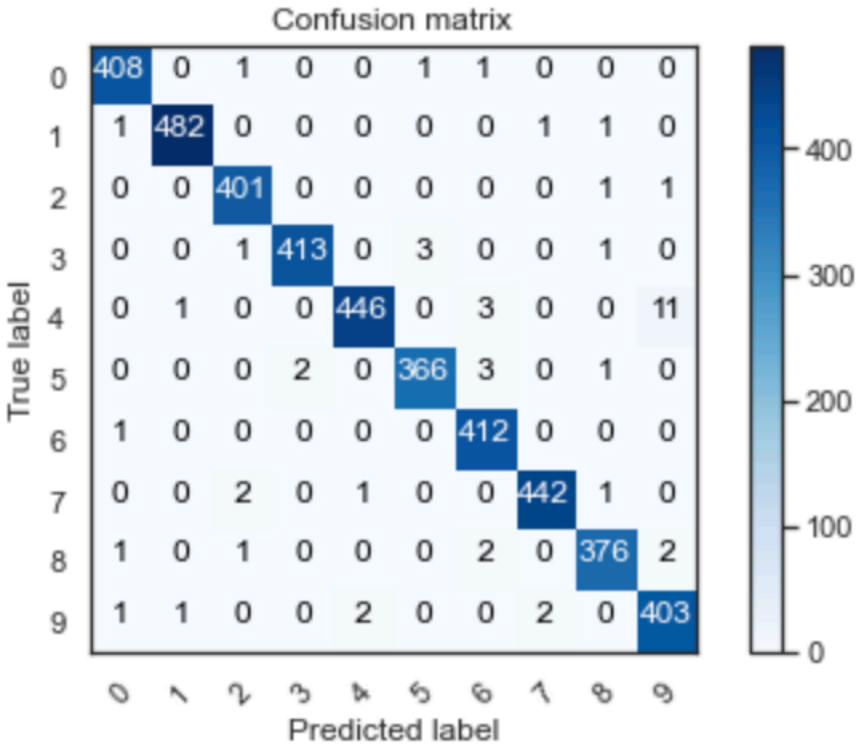

FIGURE 4 – Matrice de confusion

Nous pouvons voir ici que notre modèle fonctionne très bien sur tous les chiffres avec peu d'erreurs compte tenu de la taille de l'ensemble de validation (4200 images).

Par ailleurs, les erreurs que notre modèle a fait (cf. figure 5) paraissent acceptables, étant donné que certaines de ces erreurs peuvent aussi être faites par des humains.

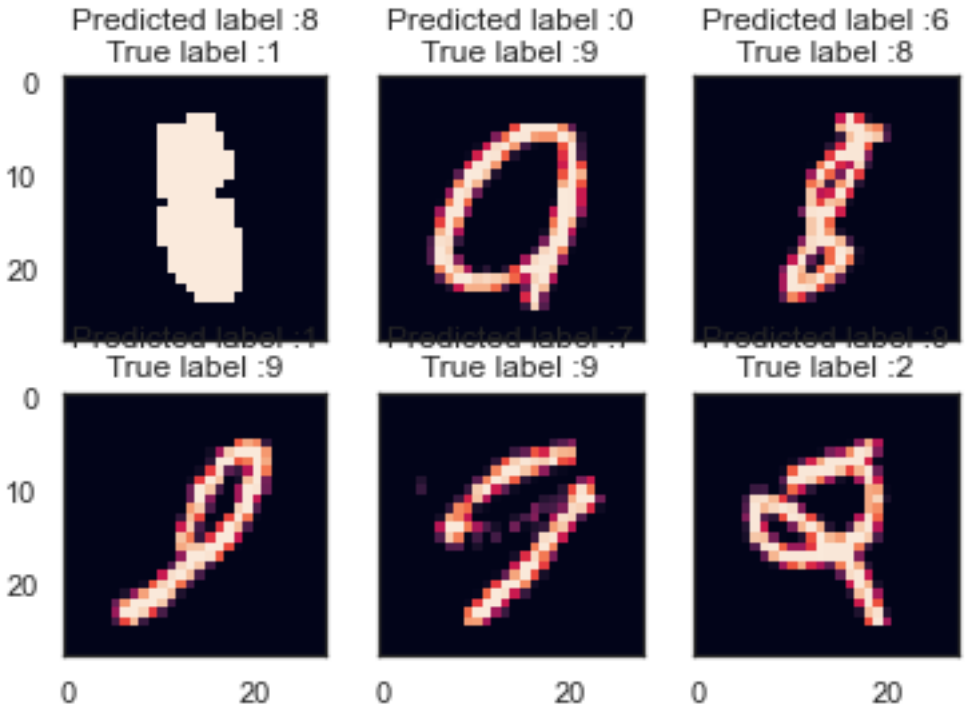

FIGURE 5 – Affichage de quelques erreurs

Enfin, nous avons testé notre modèle optimisé par les scripts de pixel 28*28, et les prédictions sont assez satisfaisantes.

```
number:5
type:1
[5]
number:0
type:1
[0]
number:4
type:1
[4]
```

FIGURE 6 – Résultats des tests

III.C.2 - Performances de l'algorithme

Nous avons fait 10 fois apprentisages,les pertes (loss) et les précision sont ci-desous :

Epoch 1/10
1182/1182 - 131s - loss : 0.1896 - accuracy : 0.9410 - val_loss : 0.0632 - val_accuracy : 0.9798
Epoch 2/10
1182/1182 - 127s - loss : 0.0787 - accuracy : 0.9762 - val_loss : 0.0495 - val_accuracy : 0.9850
Epoch 3/10
1182/1182 - 130s - loss : 0.0618 - accuracy : 0.9814 - val_loss : 0.0481 - val_accuracy : 0.9876
Epoch 4/10
1182/1182 - 143s - loss : 0.0544 - accuracy : 0.9838 - val_loss : 0.0523 - val_accuracy : 0.9838
Epoch 5/10
1182/1182 - 128s - loss : 0.0446 - accuracy : 0.9868 - val_loss : 0.0386 - val_accuracy : 0.9893
Epoch 6/10
1182/1182 - 116s - loss : 0.0433 - accuracy : 0.9870 - val_loss : 0.0372 - val_accuracy : 0.9879
Epoch 7/10
1182/1182 - 118s - loss : 0.0415 - accuracy : 0.9878 - val_loss : 0.0419 - val_accuracy : 0.9905
Epoch 8/10
1182/1182 - 118s - loss : 0.0388 - accuracy : 0.9889 - val_loss : 0.0430 - val_accuracy : 0.9895
Epoch 9/10
1182/1182 - 116s - loss : 0.0410 - accuracy : 0.9878 - val_loss : 0.0328 - val_accuracy : 0.9890
Epoch 10/10
1182/1182 - 115s - loss : 0.0343 - accuracy : 0.9902 - val_loss : 0.0473 - val_accuracy : 0.9881

En utilisant l'optimiseur ADAM, la précision d'entraînement est passée de 95.98% à 99.02% et le nombre d'itérations nécessaires réduit à environ un (avec un saut de plus de 3% pendant la première itération), c'est bien un changement assez important étant donné que nos données d'entraînement étaient d'environ 42000, pas même près d'un million, où les effets seraient encore plus importants ! De plus, le temps pour chaque itération a diminué d'environs 10 seconds avec notre optimiseur Adam.

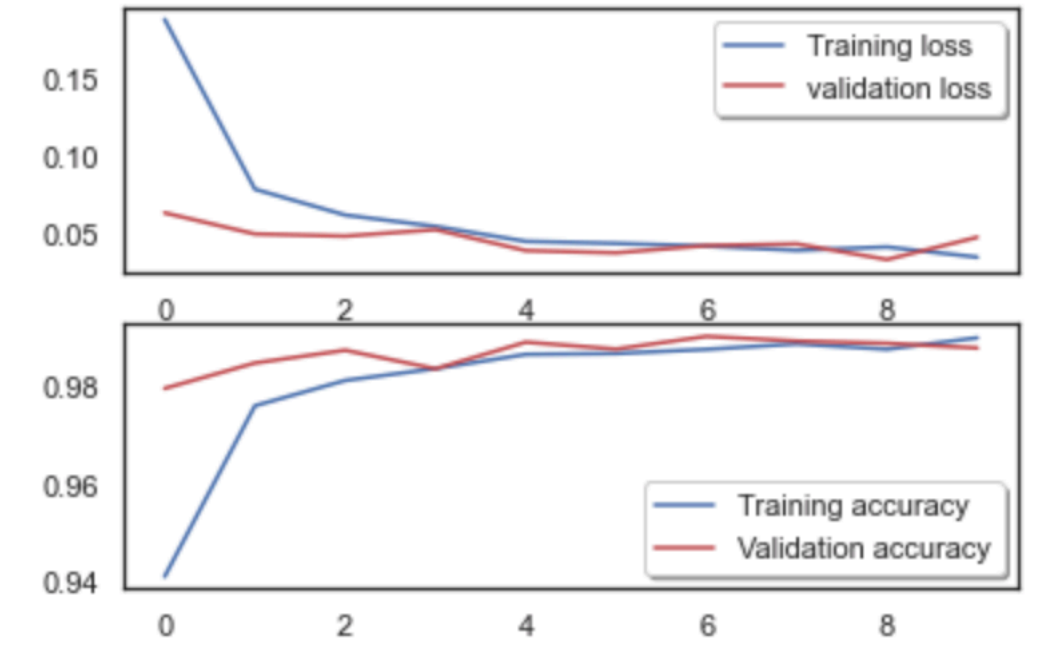

FIGURE 7 – Courbes d'entraînement et de validation

Le modèle atteint une précision de presque 99% sur l'ensemble de données de validation après 10 époques. La précision de validation est supérieure à la précision d'apprentissage presque à chaque fois pendant l'apprentissage. Cela signifie que notre modèle ne surajuste pas les données d'entraînement.
Notre modèle est très bien entraîné.

III.C.3 - Choix des paramètres

- α : Également connu sous le nom de taux d'apprentissage ou de facteur de pas, il contrôle le taux de mise à jour des poids (par exemple 0.001). Des valeurs plus grandes (par exemple, 0,3) entraîneront un apprentissage initial plus rapide avant la mise à jour du taux d'apprentissage, tandis que des valeurs plus petites (par exemple, 1E-5) feront converger l'apprentissage vers de meilleures performances.
- β_1 : taux de décroissance exponentielle de l'estimation du moment de premier ordre (par exemple, 0,9)
- β_2 : taux de décroissance exponentielle de l'estimation du moment de second ordre (par exemple, 0,99). Cet hyperparamètre doit être fixé à une valeur proche de 1 dans les gradients de coefficient.
- ϵ : ce paramètre est un très petit nombre, il sert à empêcher la division par zéro dans l'implémentation (par exemple 1e-8 qui est utilisé dans notre modèle).

III.C.4 - Avantages et inconvénients

Avantages de l'algorithme ADAM

- Ralentir lorsqu'il converge vers des minima locaux. Pour l'algorithme ADAM, la quantité de mouvement peut être vue comme une balle dévalant une pente, elle se comporte comme une balle lourde avec des frottements, qui préfère donc les minima plats dans la surface d'erreur.
- Il est adaptatif. Sinon, pour certains paramètres, même si la fonction objective a été optimisé au voisinage du minimum, certains paramètres ont encore un gradient important. Par conséquent, si le taux d'apprentissage est trop petit, les paramètres avec de grands gradients auront un taux de convergence très lent ; si le taux d'apprentissage est trop grand, les paramètres qui ont été optimisés presque aussi bien peuvent devenir instables.

Inconvénients de l'algorithme ADAM

- La divergence :
  L'article On the Convergence of Adam and Beyond, qui explore la convergence de l'algorithme d'Adam, a démontré par des contre-exemples qu'Adam peut ne pas converger dans certains cas.
- Rater la solution optimale globale :
  Les réseaux neuronaux profonds contiennent souvent un grand nombre de paramètres et, dans un espace de cette dimension, la fonction objectif non convexe a tendance à fluctuer, avec un nombre infini de hauts et de bas. Certains sont des sommets, qui peuvent être facilement franchis en introduisant un élan ; mais certains sont des plateaux, qui peuvent être explorés de nombreuses fois sans en sortir, et l'entraînement est arrêté.

IV - Conclusion

L'algorithme de l'ADAM une façon très utilisée pour accélérer le calcul.Il a profité du concept de la quantité de mouvement(momentum) avec pas aléatoires.Il risque de ne pas obtenir la convergence ou de rater la solution optimale globale, car une solution locale est trouvée d'après cet algorithme.

En pratique, quand il est appliqué dans le CNN pour identifier les chiffres manuscrits, l'ADAM nous permet de améliorer à la foid la vitess (par environ 10 secondes chaque fois) et la précision. Quand il faut plusieur fois d'apprentisage en pratique, cet algorithme nous permet de gagner enorme de temps.