Évaluation de la performance de la classification d'images à l'aide de trois algorithmes

Lea-Marie Normandin, Zhibin Lu et Xiaocheng Liu

Introduction

Nous sommes intéressés par la classification d'images. Nous allons tester trois algorithmes pour en comparer leur performance et ainsi, choisir celui qui donne les meilleurs résultats. Pour chaque algorithme, le taux d'erreur sera calculé. Nous évaluerons pour différents noyaux et différentes valeurs d'hyper-paramètres afin de trouver le modèle optimal. Pour chaque algorithme sur chacune des bases de données, nous afficherons des exemples qui ont été difficilement classés et d'autres qui ont été facilement classés.

Les trois algorithmes utilisés seront le classificateur de Bayes naïf (un algorithme qui permet de classifier les données en appliquant la règle de Bayes pour obtenir la probabilité à postériori des classes aux points d'entrée (les composantes seront supposées indépendantes)), la machine à vecteurs de support (un algorithme qui apprend une fonction discriminante linéaire en maximisant la distance d'un hyper-plan à un exemple d'entraînement le plus proche de celui-ci), le réseau convolutif (un algorithme qui consiste en la construction d'un réseau de neurones convolutifs).

Nous utiliserons une base de données constituée de chiffres manuscrits (MNIST) ainsi qu'une base de données qui contient une grande quantité d'images en couleur de format 32x32 (CIFAR-10). Dans cette dernière, les images sont divisées en 10 classes et chaque classe contient 6000 images.

1 Classificateur de Bayes naïf

L'algorithme du classificateur de Bayes est construit autour de la règle de Bayes qui dit, dans notre contexte, que

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

où $X = (X_1, X_2, \ldots, X_d) \in \mathbb{R}^d$ et $Y = c \in \{1, \ldots, m\}$.

Le classificateur de Bayes naïf ajoute la condition d'indépendance entre les composantes du vecteur X étant donnée une certaine classe. On obtient donc que

$$P(X|Y = c) = P(X_1|Y = c)P(X_2|Y = c) \ldots P(X_d|Y = c)$$

Dans les deux bases de données, il y a 10 classes.

1.1 MNIST

Un hyper-paramètre très souvent considéré dans l'algorithme du classificateur de Bayes est celui du lissage. Nous avons donc calculé les erreurs obtenues sur les ensembles d'entraînement et de validation pour différentes valeurs de cet hyper-paramètre. Ces erreurs sont affichées dans la table 1.

Table 1: Erreurs sur l'ensemble d'entraînement et l'ensemble de validation pour différentes valeurs de l'hyper-paramètre de lissage

	Erreurs	
Valeur de HP	Entraînement	Validation
0.2	0.15500	0.1673
0.5	0.15456	0.1642
2	0.15492	0.1646
20	0.15656	0.1658
40	0.15794	0.1668

Dans les deux cas, l'erreur la plus petite correspond à une valeur de l'hyper-paramètre de 0.5.

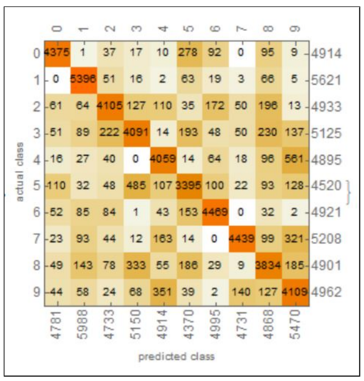

Figure 1: Matrice de confusion sur l'ensemble d'entraînement

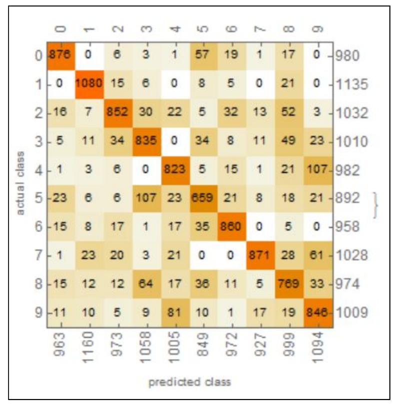

Figure 2: Matrice de confusion sur l'ensemble de test

Les matrices de confusion pour l'ensemble d'entraînement et l'ensemble de test avec un hyper-paramètre de lissage de 0.5 sont affichées aux figures 1 et 2.

D'après ces dernières, les images où l'algorithme semble le plus confus sont celles représentant le 4. L'algorithme classe 561 images de 4 comme étant des 9 sur l'ensemble d'entraînement. Il y a également le 5 qui est prédit comme étant dans la classe du 3 pour 485 images d'entrées sur ce même ensemble. Pour l'ensemble de test, le 5 est souvent prédit comme un 3 et le 4 est souvent prédit par le 9 (107 fois chacun). Il demeure toutefois que la classe prédite la plus souvent correspond à la vraie classe.

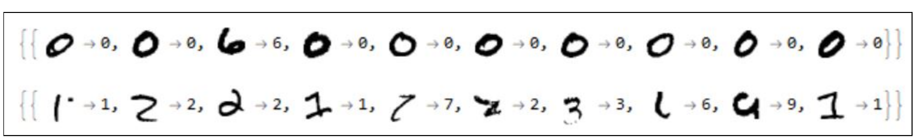

Figure 3: Exemples facilement classifiés et difficilement classifiés

On retrouve à la première ligne de la figure 3 quelques exemples de l'ensemble de test qui sont facilement classifié par l'algorithme et à la seconde ceux qui causent plus de difficultés.

1.2 CIFAR-10

Nous suivons les mêmes étapes sur cette nouvelle base de données.

L'hyper-paramètre de lissage conservé est 0.2 puisqu'il correspond à la plus petite erreur sur l'ensemble d'entraînement (voir la table 2).

Les matrices de confusion sur l'ensemble d'entraînement et l'ensemble de test sont les figures 4 et 5. Pour l'ensemble d'entraînement, plusieurs cases à l'extérieur de la diagonale principale sont

Table 2: Erreurs sur l'ensemble d'entraînement et l'ensemble de validation pour différentes valeurs de l'hyper-paramètre de lissage

	Erreurs	
Valeur de HP	Entraînement	Validation
0.2	0.5879	0.7385
0.5	0.5881	0.7385
2	0.5882	0.7385
20	0.5884	0.7385
40	0.5893	0.7390

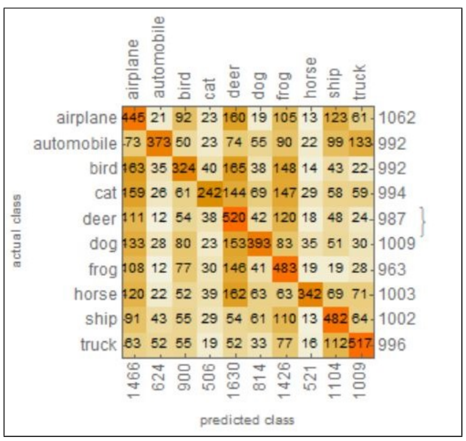

Figure 4: Matrice de confusion sur l'ensemble d'entraînement

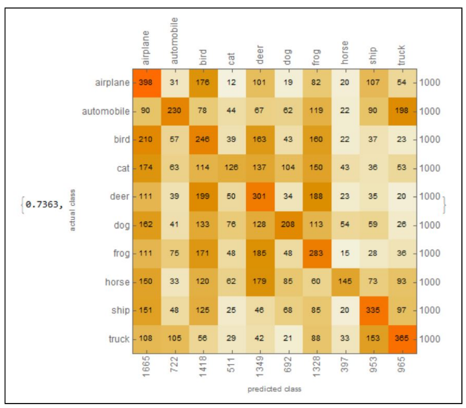

Figure 5: Matrice de confusion sur l'ensemble de test

assez foncées ce qui réflète grandement le taux d'erreur obtenu de 0.5879. Ce taux d'erreur, malgré qu'il soit le plus petit parmi les différentes valeurs pour l'hyper-paramètre, demeure un taux très grand. L'algorithme se trompe plus d'une fois sur deux. Pour l'ensemble de test, il est plus difficile de distinguer la diagonale principale qui est normalement un bon indicateur que l'algorithme a bien prédit chacune des classes. On voit ici que ce classificateur est très souvent mélangé et interprète mal une bonne quantité d'images. Cela peut fort probablement être expliqué par la supposition d'indépendance des composantes d'entrées et donc le manque de force de ce classificateur.

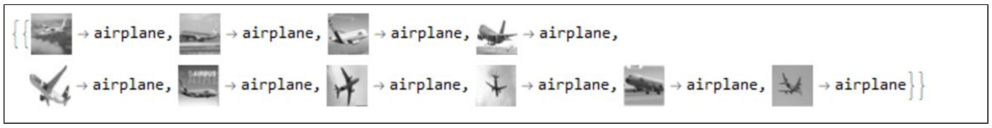

Figure 6: Exemples facilement classifiés

Figure 7: Exemples difficilement classifiés

Quelques exemples bien classifiés par l'algorithme se trouvent à la figure 6 et quelques exemples plus difficiles à classifer se trouvent à la figure 7.

2 Machines à vecteurs de support

Les machines à vecteurs de support constituent un classificateur binaire. Les évaluations se font donc un à un ou un contre tous les autres. Un exemple de cette dernière option : soit l'exemple d'entrée fait partie d'une certaine classe c ou non. À partir de ce principe, l'algorithme apprend une fonction discriminante linéaire à marge maximale. Un hyperplan est également trouvé. Il est meilleur si il sépare parfaitement les données et si il se trouve à une distance éloignée du point d'entraînement le plus proche.

2.1 MNIST

Table 3: Erreurs sur l'ensemble d'entraînement et l'ensemble de test pour plusieurs noyaux

	Erreurs	
Type de noyau	Entraînement	Test
Linéaire	0.03832	0.0524
Polynomial	0.02696	0.0372
Gaussien	0.04654	0.0493

Dans le but de trouver un classificateur non-linéaire, nous commençons par comparer différents noyaux possibles. Nous trouvons le taux d'erreur sur l'ensemble d'entraînement et l'ensemble de test pour le noyau linéaire, le noyau polynomial et le noyau gaussien. Ces taux sont affichés à la table 3. Dans le logiciel utilisé, le noyau linéaire est d'équation $K(x_1, x_2) = x_1 x_2$, le noyau polynomial est d'équation $K(x_1, x_2) = \gamma(c + x_1 x_2)^d$ et le noyau gaussien est $K(x_1, x_2) = \exp\{\gamma|x_1 - x_2|^2\}$.

Le type de noyau qui obtient la plus petite erreur autant sur l'ensemble d'entraînement que sur l'ensemble de test est le noyau polynomial. Nous considèrons donc différents degrés pour le polynôme (un hyper-paramètre) et en déterminons les erreurs sur l'ensemble de validation. Celles-ci sont affichées dans la table 4. L'erreur la plus petite correspond à un noyau de type polynomial de degré 9. Il reste donc finalement à chosisir l'hyper-paramètre gamma γ.

Sur l'ensemble de validation, les erreurs pour différentes valeurs de gamma sont inscrites à la table 5. Étant donné que l'erreur la plus petite est à la fois pour un gamma de 1 et un gamma de 10, on conservera un gamma de 1.

Table 4: Erreurs sur l'ensemble de validation pour différents degrés du polynôme

Degré	Erreurs de validation
3	0.0397
4	0.0314
5	0.0257
9	0.0234

Table 5: Erreurs sur l'ensemble de validation pour différents gammas

γ	Erreurs de validation
0.0001	0.0578
0.001	0.0223
0.01	0.0183
0.1	0.0184
1	0.0180
10	0.0180

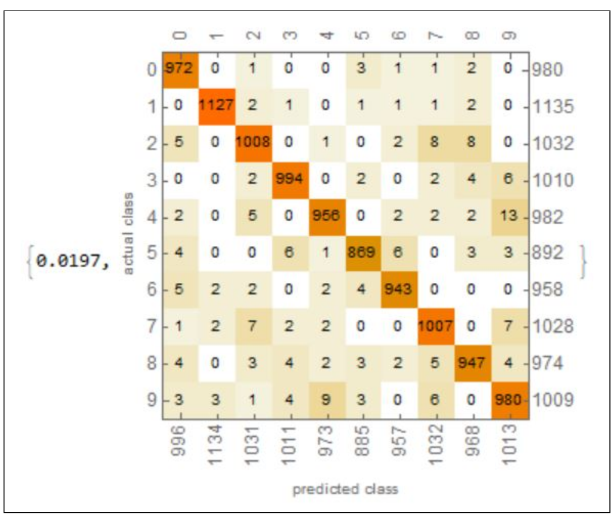

Figure 8: Matrice de confusion pour le modèle choisi

Sur l'ensemble de test, on obtient donc la matrice de confusion de la figure 8. Nous remarquons que les machines de vecteurs à support semblent mieux prédire les classes que le classificateur de Bayes naïf. En effet, en comparant les deux matrices de confusion (2 et 8) sur l'ensemble de test, il y a beaucoup moins de couleur foncée à l'extérieur de la diagonale principale pour les SVM que pour le classificateur de Bayes naïf.

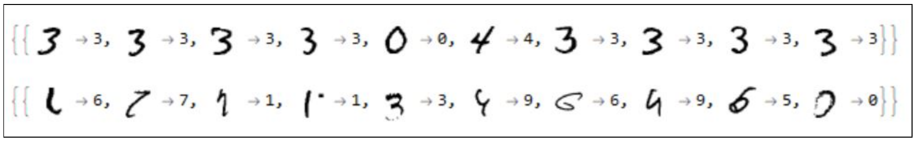

Figure 9: Exemples facilement classifiés et difficilement classifiés

Quelques exemples d'images facilement classifiées et difficilement classifiées se trouvent à la figure 9.

Table 6: Erreurs sur l'ensemble d'entraînement et l'ensemble de test pour plusieurs noyaux

	Erreurs	
Type de noyau	Entraînement	Test
Linéaire	0.5469	0.7820
Polynomial	0.7524	0.7502
Gaussien	0.8272	0.8213

2.2 CIFAR-10

En suivant la même logique que pour la base de données MNIST, nous trouvons les erreurs pour les différents noyaux.

L'erreur de test la plus petite est celle du noyau de type polynomial comme le montre la table 6.

Table 7: Erreurs sur l'ensemble de validation pour différents degrés du polynôme

Degré	Erreurs de validation
3	0.75
4	0.845
5	0.7495
9	0.7375

En calculant les erreurs sur l'ensemble de validation, le degré 9 pour le polynôme est celui qui comporte l'erreur la plus petite. Ensuite, en ayant conservé un degré de 9, la valeur de l'hyper-paramètre gamma qui donne la plus petite erreur est 0.001 comme il est possible de voir à la table 8.

La matrice de confusion sur l'ensemble de test est donc celle affichée à la figure 10 et les exemples facilement classifiés et difficilement classifiés sont aux figures 11 et 12 respectivement.

Table 8: Erreurs sur l'ensemble de validation pour différents gammas

γ	Erreurs de validation
0.0001	0.763
0.001	0.678
0.01	0.7155
1	0.776
10	0.7965

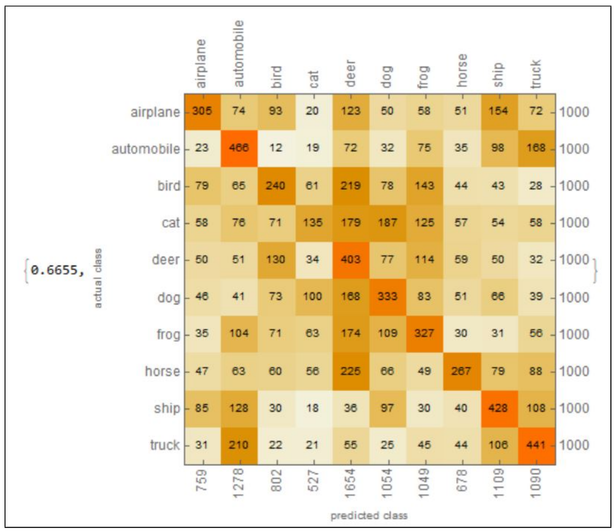

Figure 10: Matrice de confusion pour le modèle choisi

Figure 11: Exemples facilement classifiés

Figure 12: Exemples difficilement classifiés

3 Réseau convolutif

Le réseau de neurones convolutif considère une matrice de poids partagé (filtre) et fait une convolution sur l'image d'entrée dans le but d'extraire des informations spécifiques sans perdre de généralité. Il permet également de réduire le nombre de paramètres, c'est-à-dire le nombre de pixels.

3.1 MNIST

On commence par trouver les paramètres qui minimise le risque empirique. Cette étape, appelée entraînement, consiste en l'utilisation de la méthode de la descente du gradient pour l'optimisation des paramètres. À partir de cela, il est possible de varier certains hyper-paramètres en gardant les autres constants.

En considérant un nombre d'époques égal à 5, une taille des lots de 500 et un terme de régularisation nul, on fait varier le taux d'apprentissage.

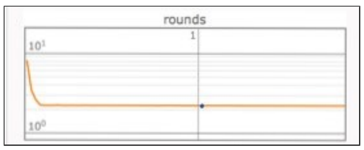

Figure 13: Taux d'apprentissage = 5

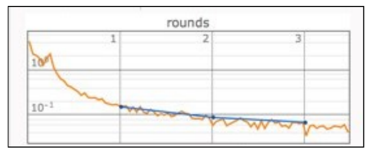

Figure 14: Taux d'apprentissage = 4

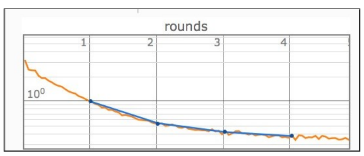

Figure 15: Taux d'apprentissage = 0.05

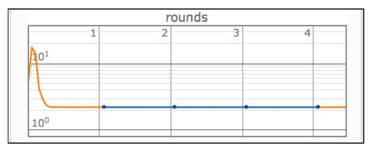

Figure 16: Taille des lots = 2000

Pour un taux d'apprentissage de 5, la méthode de descente du gradient n'aboutie pas. En effet, on remarque sur la figure 13 que le graphique affiche une courbe de loss constante à partir d'une certaine valeur.

Pour un taux d'apprentissage de 4 (voir la figure 14), malgré que la méthode de descente du gradient fonctionne, il semble y avoir de l'instabilité. On obtient, pour trois entraînements différents, des taux d'erreur très variables. Les trois taux d'erreur obtenus sont 0.0142, 0.0438 et 0.0133.

Pour un taux d'apprentissage de 0.05, la descente du gradient est lente. Cela est remarqué en regardant l'échelle du graphique de la figure 15.

Maintenant, en considérant un nombre d'époques égal à 5, un taux d'apprentissage de 3 et un terme de régularisation nul, on fait varier la taille des lots.

Pour une taille des lots de 2000, on remarque qu'une trop grosse taille des lots empêche l'utilisation de la descente du gradient. Après plusieurs époques, on voit à la figure 16 que la courbe est constante.

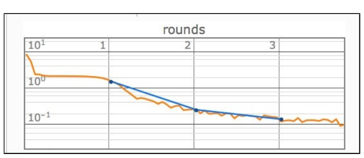

Figure 17: Taille des lots = 1000

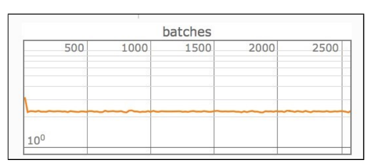

Figure 18: Taille des lots = 5

Pour une taille des lots de 1000, divers entraînements arrivent à différents taux d'erreur. On obtient les taux suivants : 0.0297, 0.0657 et 0.0205. On a donc, une fois de plus, instabilité comme le montre la figure 17.

Pour une taille des lots de 5, la descente du gradient est très lente pour ne pas dire inexistante (voir la figure 18).

Ensuite, on ajuste l'hyper-paramètre afin de contrôler le terme de régularisation.

En considérant un nombre d'époques de 30, un taux d'apprentissage de 1.8 et une taille des lots de 100, on tente de trouver la valeur de cet hyper-paramètres qui donne la meilleure performance.

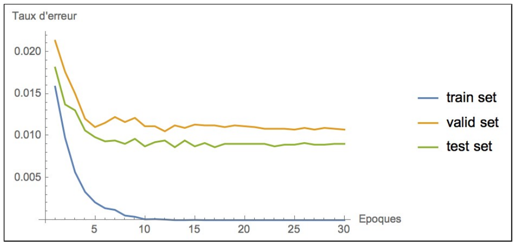

Figure 19: Terme de régularisation = 0.000000000001

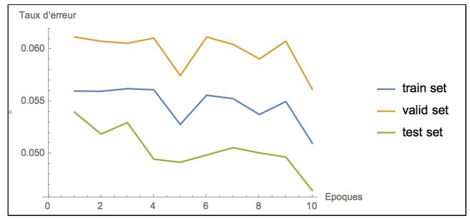

Figure 20: Terme de régularisation = 0.05

Pour un terme de régularisation nettement inférieur à zéro (0.000000000001), on remarque dans la figure 19 une situation de sur-apprentissage. En effet, on remarque une différence significative entre l'erreur sur l'ensemble d'entraînement et celle sur l'ensemble de test plus le nombre d'époques augmentent. Après l'époque 10, on constate que le taux d'erreur sur l'ensemble d'entraînement est proche de 0 alors que celui pour les deux autres ensembles est entre 0.010 et 0.014. Ce taux d'erreur sur l'ensemble de test est tout de même petit ce qui est expliqué, entre autres, par la taille de l'échantillon.

Pour un terme de régularisation de 0.05, on remarque une situation de sous-apprentissage. La figure 20 illustre effectivement un biais assez grand, mais une petite variance.

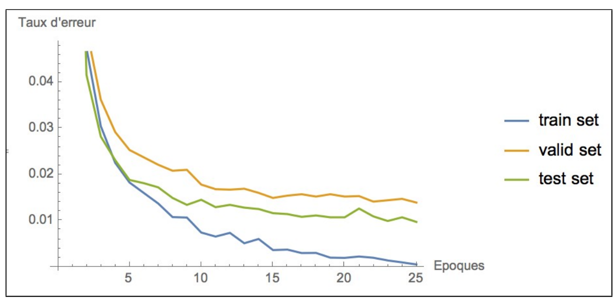

Figure 21: Graphique obtenu avec les paramètres et hyper-paramètres choisis

Suite à cette évaluation, nous avons décidé de fixer le nombre d'époques à 25, le taux d'apprentissage à 1.7, la taille des lots à 240 et le terme de régularisation à 0.000001. Pour ces valeurs, on obtient le graphique du taux d'erreur en fonction des époques à la figure 21.

On obtient donc les taux d'erreur et les matrices de confusion sur les ensembles d'entraînement et de test, respectivement aux figures 22 et 23.

Pour évaluer le niveau de la performance de l'algorithme, nous considérons l'entropie, c'est-à-dire le niveau d'incertitude par rapport aux différentes images d'entrée. Une entropie élevée signifie la possibilité de confusion et donc aussi la possibilité d'échec de l'algorithme. Une entropie faible signifie, pour sa part, une bonne chance de réussite de l'algorithme. À la figure 24, la première ligne contient les exemples où l'algorithme échoue alors que la seconde contient les exemples où l'algorithme réussit le mieux. D'après les matrices de confusion, nous pouvons constater que l'algorithme semble parfois confondre le 6 avec le 0, le 7 avec le 2 et le 4 avec le 9.

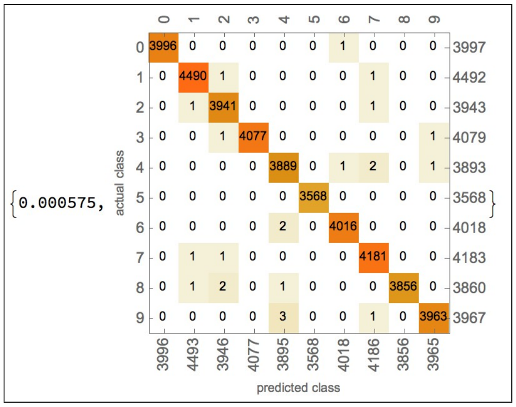

Figure 22: Taux d'erreur et matrice de confusion de l'ensemble d'entrainement

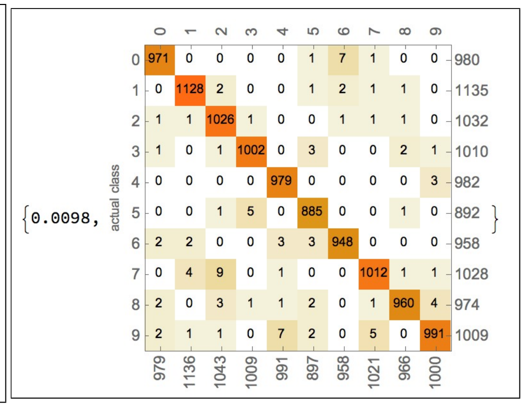

Figure 23: Taux d'erreur et matrice de confusion de l'ensemble de test

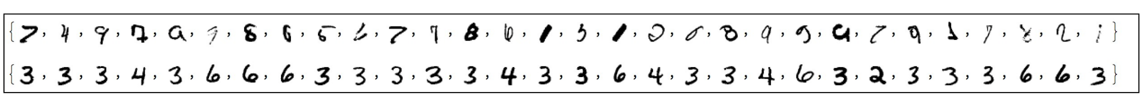

Figure 24: Exemples

3.2 CIFAR-10

Suite à l'expérience sur la base de données MNIST, on considère un nombre d'époques entre 10 et 20, une taille des lots entre 200 et 300, un terme de régularisation de 0.0000001, un taux d'apprentissage de 2.4 et deux couches cachées contenant respectivement 800 et 100 neurones.

On commence par vérifier l'influence de certains hyper-paramètres tels que le nombre de couches de convolution, le nombre de couches de pooling ainsi que leurs dimensions respectives.

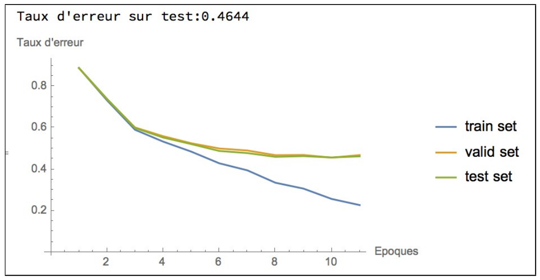

Figure 25: Premier test

On commence à tester l'algorithme dans une situation où l'on a une couche de convolution et une couche de pooling. La couche de convolution contient 30 filtres de dimension 5x5 et la couche de pooling est composée d'un seul filtre de dimension 2x2. La figure 25 représente le graphique du taux d'erreur en fonction des époques.

On constate une situation de sous-apprentissage. En effet, la capacité du modèle n'est pas suffisament élevée.

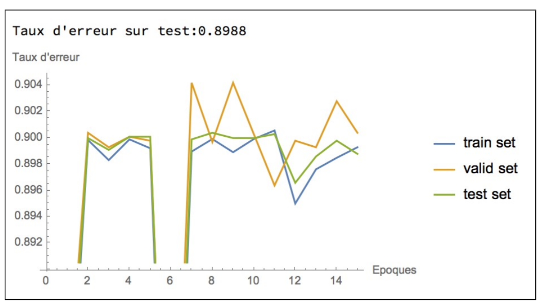

Figure 26: Deuxième test

On tente maintenant de considérer deux couches de convolution et deux couches de pooling. La première couche de convolution est composée d'une trentaine de filtres de dimension 9x9 et la seconde est composée de 90 filtres de même taille. Les deux couches de pooling maintiennent une dimension de 2x2 avec un seul filtre chacun. La figure 26 affiche un résultat incohérent qui nous permet de dire que la taille des filtres est trop grande pour des images de format 32x32.

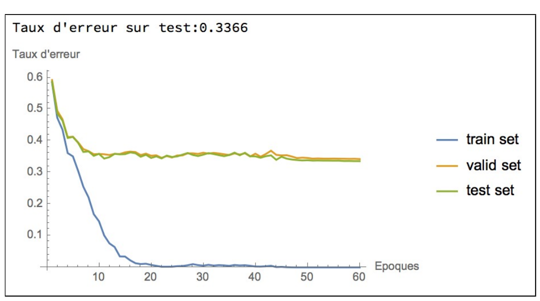

Figure 27: Troisième test

On recommence donc la dernière expérience en ne changeant que la taille des filtres des couches de convolution. La taille est établie à 5x5. La figure 27 affiche un résultat meilleur que ceux des deux tests précédents. Le taux d'erreur est plus bas.

Cette amélioration marquée nous pousse à augmenter davantage le nombre de couches et à diminuer la taille des filtres. Pour trois couches de convolution avec des filtres de dimension 3x3 et trois couches de pooling avec un filtre de 2x2 chacun. Le nombre de filtres pour chaque couche de convolution est 60, 120 et 240 respectivement.

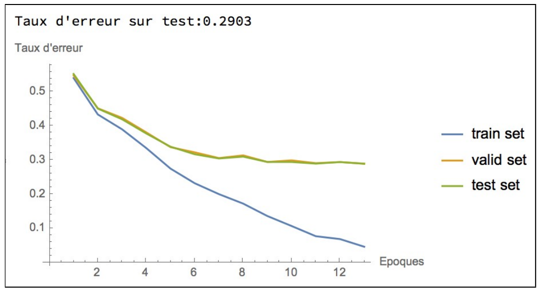

Figure 28: Quatrième test

On obtient donc le graphique de la figure 28.

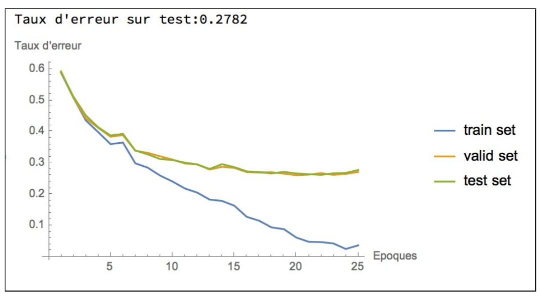

Figure 29: Cinquième test

Jusqu'à maintenant, la fonction exécutée sur chacune des couches de pooling est la fonction max. En considérant plutôt la fonction max seulement sur la première couche et la fonction moyenne sur les deux autres, le résultat obtenu est encore meilleur que les précédents comme il est possible de voir à la figure 29. Le taux d'erreur est à son plus bas.

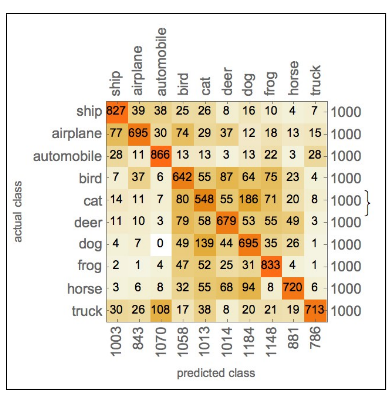

Figure 30: Matrice de confusion CIFAR-10

La matrice de confusion obtenue en exécutant le dernier algorithme choisi est celle affichée à la figure 30.

Figure 31: Entropie plus élevée

Figure 32: Entropie plus basse

Les images ayant les entropies les plus hautes et les plus basses sont aux figures 31 et 32 respectivement.

On constate, à partir de la matrice de confusion que 108 camions ont été considérés comme des automobiles, que 186 chats ont été classés comme chien par l'algorithme et que 139 chiens ont été placés dans la catégorie chat. Les images dont l'entropie est la plus élevée semblent avoir moins de contrastes que celles dont l'entropie est plus basse. Cela explique en grande partie la confusion de l'algorithme. Les images les mieux classées sont les automobiles et les chevaux.

4 Conclusion

Nous avons remarqué que le taux d'erreur décroît dans l'ordre des algorithmes présentés pour la base de données MNIST. L'algorithme du réseau convolutif serait donc le meilleur. Pour la base de données CIFAR-10, les deux premiers classificateurs ne donnent pas un super résultat et le réseau convolutif est de loin meilleur aux précédents. Le réseau convolutif est donc un bon classificateur pour un problème avec des images. Pour un projet futur, il serait intéressant d'essayer d'autres algorithmes sur ces mêmes bases de données afin de voir s'il y en a d'autres qui donnent de meilleurs résultats.

Répartition

La répartition fût équitable entre les membres de l'équipe.