Rapport UE Apprentissage statistique : Réconnaissance d'images

Nicolas Leclerc Damien Gabriel
Mohammed Boutahar

M1 Informatique 2015-2016

Table des matières

Chapitre 1

Introduction

Pour un humain, la classification des images est assez facile sur des images bien définies, ce qui est le cas du CIFAR-10.

Il est intéressant de pouvoir classifier automatiquement des images, ou de façon plus générale, de pouvoir extraire des informations automatiquement d'un ensemble de données. On peut imaginer de nombreuses applications comme par exemple la création automatique de légende pour des images ou encore l'identification de personnes.

Chapitre 2

CIFAR-10

La base CIFAR-10 est une base d'images de petites taille contenant 60000 images de 32 * 32 pixels. Ces images sont réparties dans 10 classes (avions, voiture, oiseau, chat, biche, chien, grenouille, cheval, bateau et camion), chacune contenant 6000 images.

Il y a 6 lots de données répartis en 5 d'apprentissage et 1 de test.

Le lot de test contient 10000 images avec 1000 images sélectionnées dans chaque classe. Les lots d'entrainement contiennent le reste.

Il n'existe pas de recouvrement entre les classes, elles sont mutuellement exclusives.

Les données peuvent être récupérées sous différents format. Dans le cadre d'une utilisation avec `python`, on utilisera des fichiers produits par cPikle, qui permettent d'obtenir un dictionnaire de la forme :

clé : data 10000x3072 d'entier non signés 8 bit (0-255). Chaque ligne contient une image 32 * 32 pixels. Les canaux sont ensuite dans l'ordre R (rouge), G (vert), B (bleu), chaque canal a 1024 (32 * 32) éléments. Les images sont stockées par lignes.

clé : label 10000 contient un nombre de 0 à 9 permettant de connaitre la classe des 10000 images.

La page `http ://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html` contient les meilleurs résultats obtenu sur la base CIFAR-10.

Actuellement le taux de reconnaissance atteint 96.53 % [2].

Il est très peu probable que nous atteignons un tel taux de reconnaissance. En effet, ceux-ci suppose des algorithmes complexes avec des réseaux de neurones comme dans [2]. Le temps limité ne nous permettra pas d'implémenter et de trouver les meilleurs paramètres pour obtenir les meilleurs résultats.

Un autre problème sera la puissance de calcul disponible qui ne permettront pas de traiter un très grand nombre d'images d'apprentissage.

Chapitre 3

Méthode naïve

Dans un premier temps, on utilisera comme caractéristique les valeurs de chaque pixel.

On peut utiliser comme valeur des pixels la « norme » des composantes R, G, B avec :

- R : la composante rouge de l'image
- G : la composante verte
- B : la composante bleu

Avec cette définition de la valeur, *val* des pixels, celle-ci peut s'écrire :

$$val = \sqrt{R^2 + G^2 + B^2} \tag{3.1}$$

Ci-dessous on présente les résultats obtenus en utilisant la totalité du `data_batch1` et du `test_batch`.

Plusieurs cas on été réalisés :

- Le perceptron non moyenné et sans shuffle sur 100 périodes
- Le perceptron non moyenné avec shuffle sur 100 périodes
- Le perceptron moyenné avec shuffle sur 100 périodes
- La k-moyenne sur 30 périodes

3.1 Perceptron

Le perceptron est un séparateur linéaire permettant de classifier avec supervision de manière binaires des images.

Pour une entrée x on renvoit soit 0 ou 1 (ou -1 et 1) selon la valeur du produit scalaire entre les poids w et le vecteur d'entrée est supérieur à un seuil b.

- si $w \,.x + b > 0 : f(x) = 1$
- sinon $f(x) = 0$

Si la valeur retournée est positive alors on considère que x appartient a cette classe.

Si la valeur retournée est nulle alors x n'appartient pas à la classe considéré.

Dans le cas d'un perceptron multiclasse, on utilise un perceptron pour chaque classe.

Dans un premier temps on fait « apprendre » à l'algorithme à reconnaitre les images dont on connait déjà la classe. On procède de manière itérative en déterminant les poids associé à un vecteur de caractéristique afin de minimiser les erreurs de prédiction. Les vecteurs de caractéristiques sont créés de façon à représenter les objets de l'on manipule. Par exemple, dans le cas d'image, on peut utiliser des valeurs associée aux pixels telle que la norme avec différents regroupements.

3.1.1 Non moyenné et sans shuffle

La figure ci-dessous représente, pour chaque classe, le nombre d'image mal reconnues en fonction de la période.

Chaque classe contient 1000 images, ce qui nous permet de comparer directement le taux de reconnaissance de chaque classe.

Globalement, le nombre d'erreur est important et même dans le cas des images représentant des avions et des bateaux, qui sont les classes les mieux reconnues, on n'a qu'environ 300 images reconnues correctement. Soit un taux de reconnaissance d'environ 30 %.

On peut penser que la faible variabilité de la couleur du ciel et de la mer, dans le cas des images d'avions et de bateau, permet de les identifier facilement, une fois qu'un l'on connait la couleur de base. Ainsi un processus basé sur l'utilisation de couleur (par opposition à une détection de contour) est elle suffisante. Au contraire, on peut penser, pour les classes mal reconnues, que la variabilité des couleurs est trop forte pour pouvoir identifier correctement une image uniquement à l'aide de la norme euclidienne des composantes R,G,B des pixels.

De même la composition des images ne sera pas identique. Autant un avion ou un bateau sera très souvent sur un fond relativement uni, autant une photo de la classe cerf aura pour fond une forêt avec de fortes variations de couleur autour du cerf.

On observe une convergence de chaque classe bien que pour certaines le taux de d'images mal reconnues soit croissant. En effet il se peut que l'algorithme prédisait trop souvent ce type d'image ce qui implique qu'il se trompait souvent sur les autres classes et non sur la classe prédite.

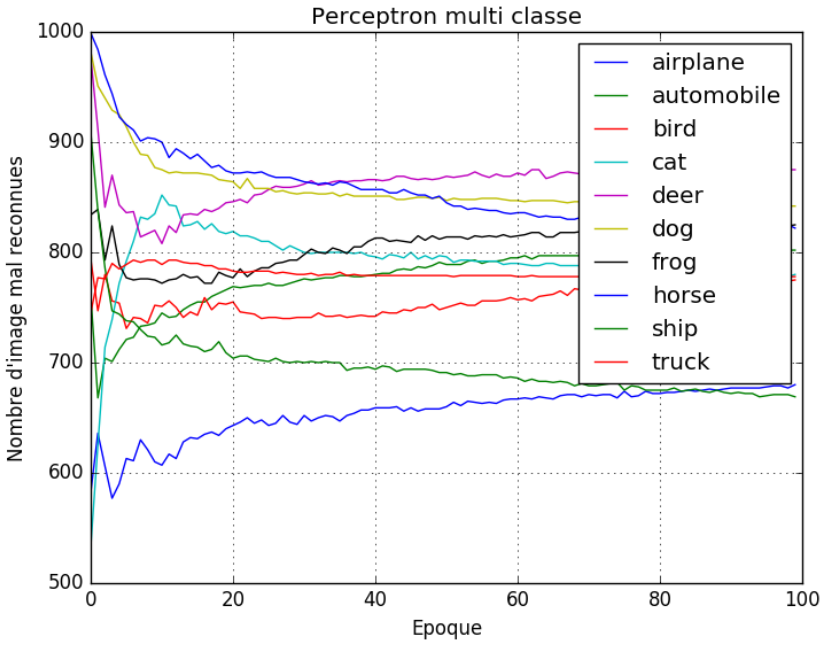

FIG. 3.1 : Nombre d'images mal reconnues pour chaque période et pour chaque classe pour le perceptron non moyenné et sans shuffle

La figure suivante représente le taux d'erreur global pour les différentes périodes. Au cours des période le taux d'erreur global diminue et tend à se stabiliser bien que lentement. Il reste en dessus du taux sur l'apprentissage puisque les jeux de données sont distincts.

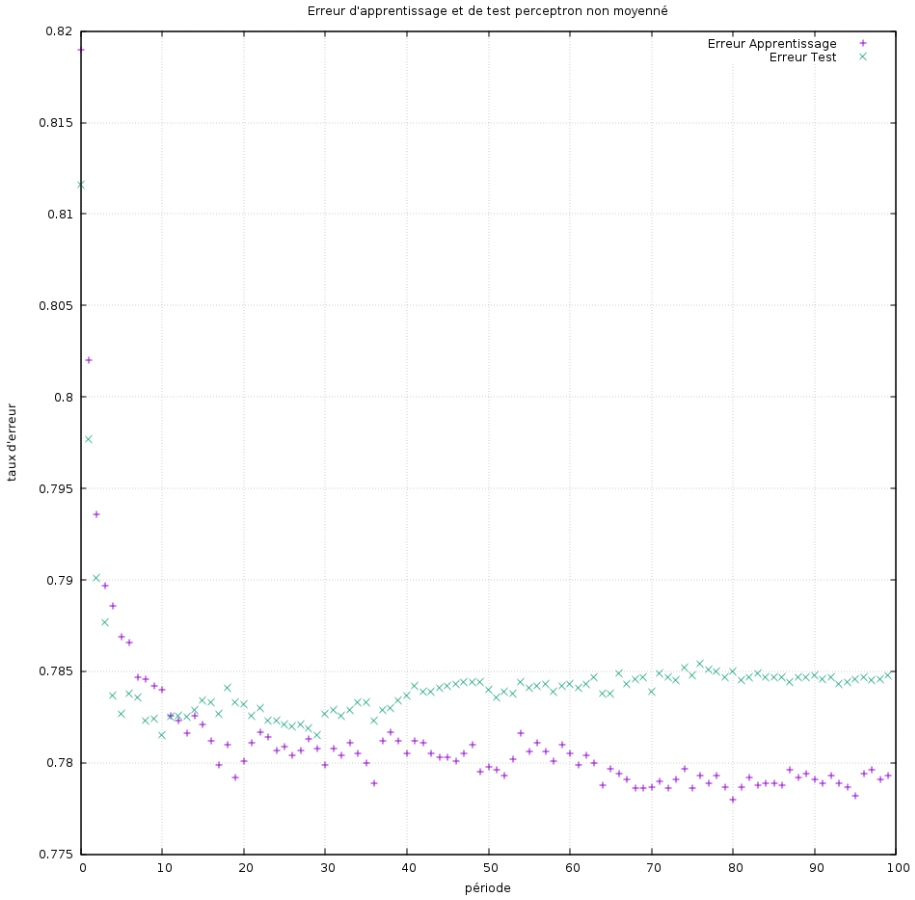

FIG. 3.2: Taux d'erreur global, perceptron non moyenné sans shuffle

3.1.2 Non moyenné avec shuffle

La figure 3.3 représente le nombre d'images mal reconnues, pour chaque classe au cours des différentes périodes. L'atout du shuffle est de minimiser le poids du ou des dernières itérations de l'apprentissage par une(des) image(s) trop singulières. Les résultats sont sensiblement meilleur bien que très variables car les images de fin d'apprentissage varient elles aussi.

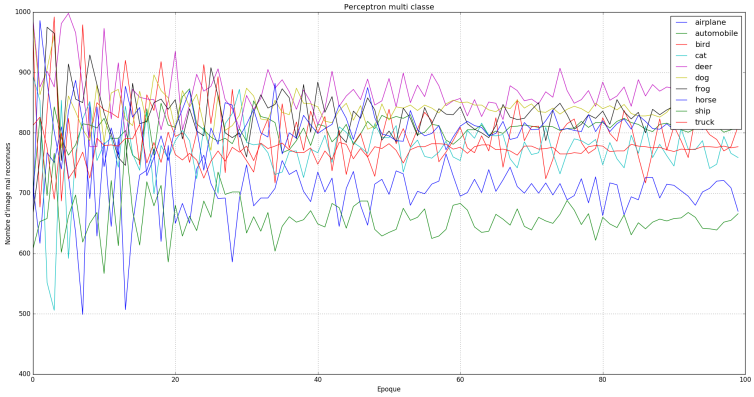

FIG. 3.3 : Nombre d'images mal reconnues pour chaque période et pour chaque classe pour le perceptron non moyenné avec shuffle

La figure 3.4 représente le taux d'erreur global pour les différentes périodes.

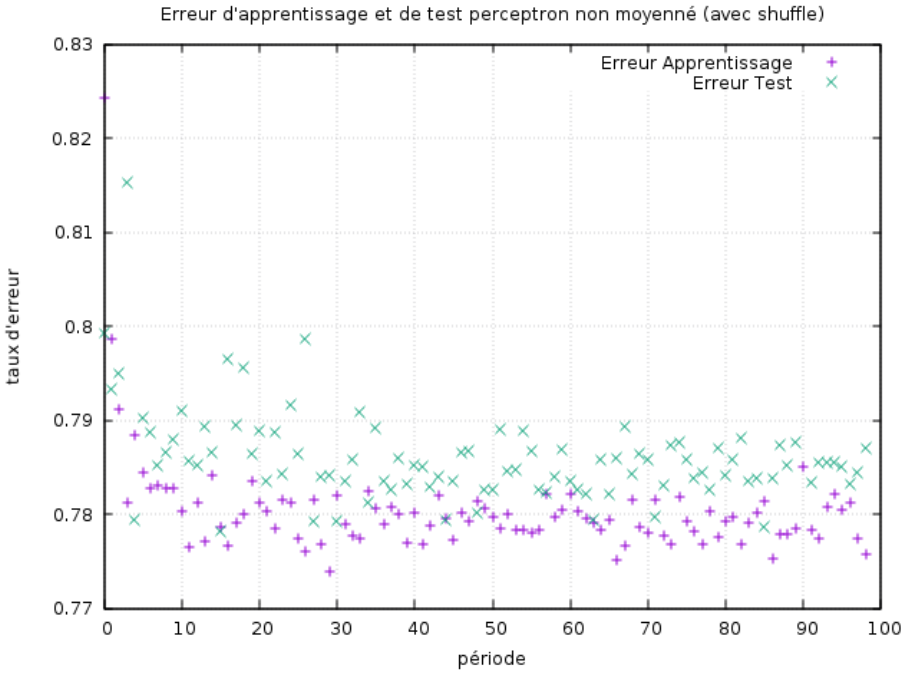

FIG. 3.4: Taux d'erreur global, perceptron non moyenné avec shuffle

3.1.3 Moyenné avec shuffle

La figure 3.5 représente le nombre d'images mal reconnues, pour chaque classe au cours des différentes périodes. Afin de supprimer la variantion trop important apporté par le suffle, nous utilisons un perceptron moyenné. Cela permet d'ammortir les changements de modèles du perceptron lors d'images trop singulières.

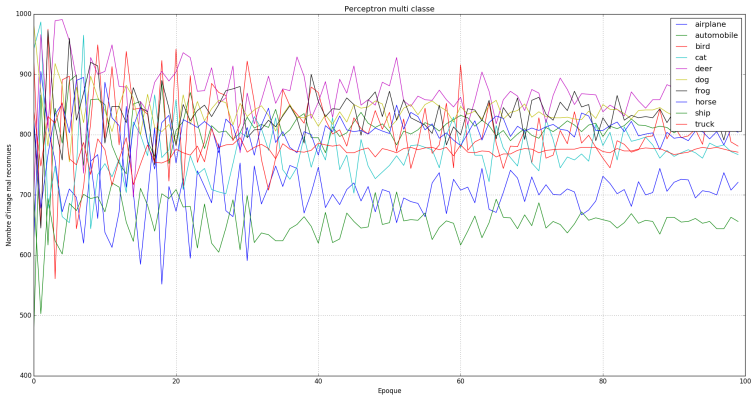

FIG. 3.5 : Nombre d'images mal reconnues pour chaque période et pour chaque classe pour le perceptron moyenné avec shuffle

La figure 3.6 représente le taux d'erreur global pour les différentes périodes.

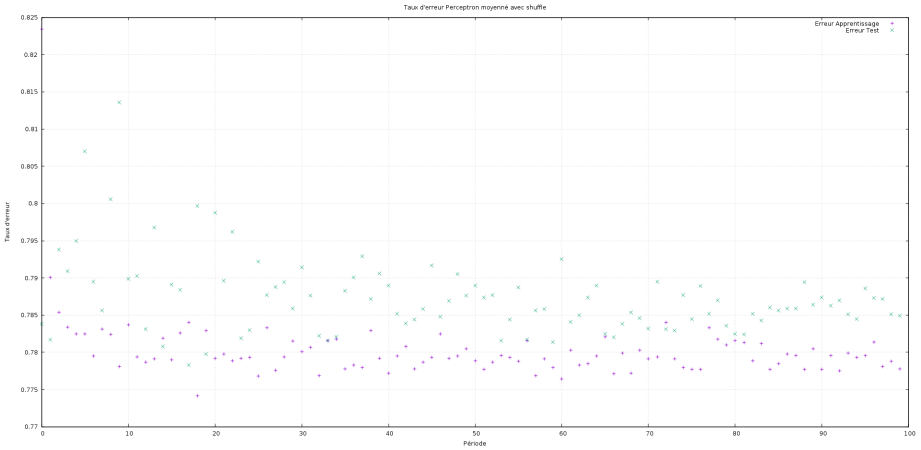

FIG. 3.6: Taux d'erreur globale, perceptron moyenné avec shuffle

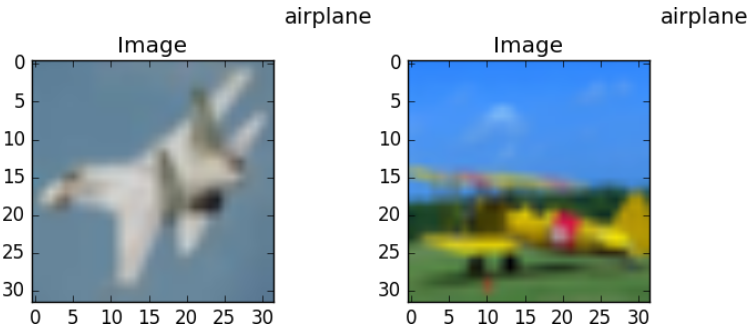

FIG. 3.7 : Deux images de la classe « airplane » : noter la composition homogène de la partie supérieure de l'image. De même le ciel ne présente pas une grande différence de teinte

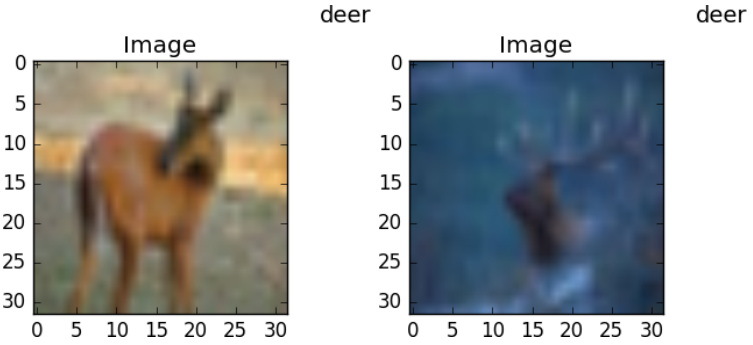

FIG. 3.8 : Deux images de la classe « deer » : noter la grande différence de couleur de fond entre les deux images

3.1.4 K-moyennes

L'algorithme des k-moyennes n'étant pas supervisé les résultats obtenus ne permettent pas de classer les images. La prédiction est loin d'être satisfaisante (près de 90 % de taux d'erreur). On peut cependant améliorer ses performances avec des centroïdes initiaux bien choisis. Malheureusement, avec notre choix de distance basée sur les valeurs de pixel, ainsi que les données des images non traitées, il n'est pas possible de créer des centroïdes représentatifs d'une classe.

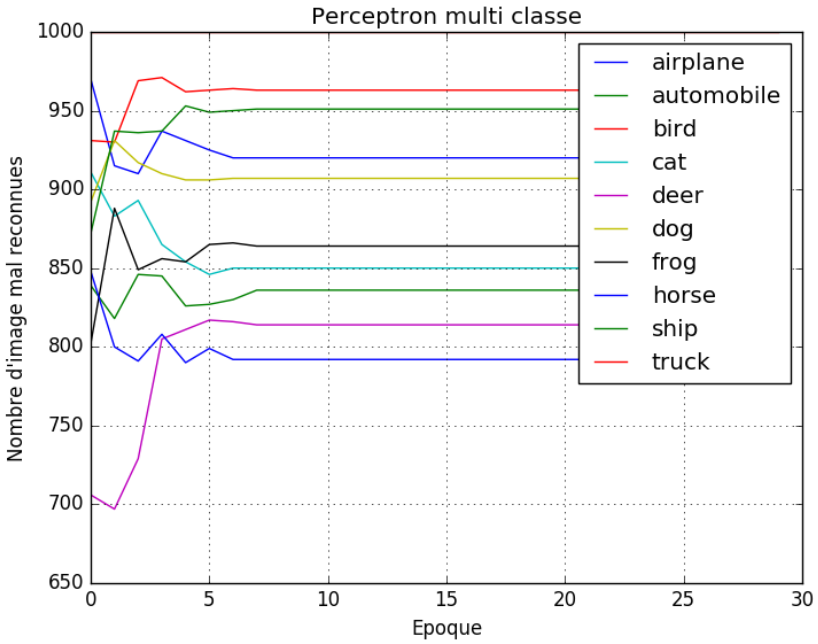

FIG. 3.9 : Nombre d'images mal reconnues pour chaque période et pour chaque classe pour la k-moyenne

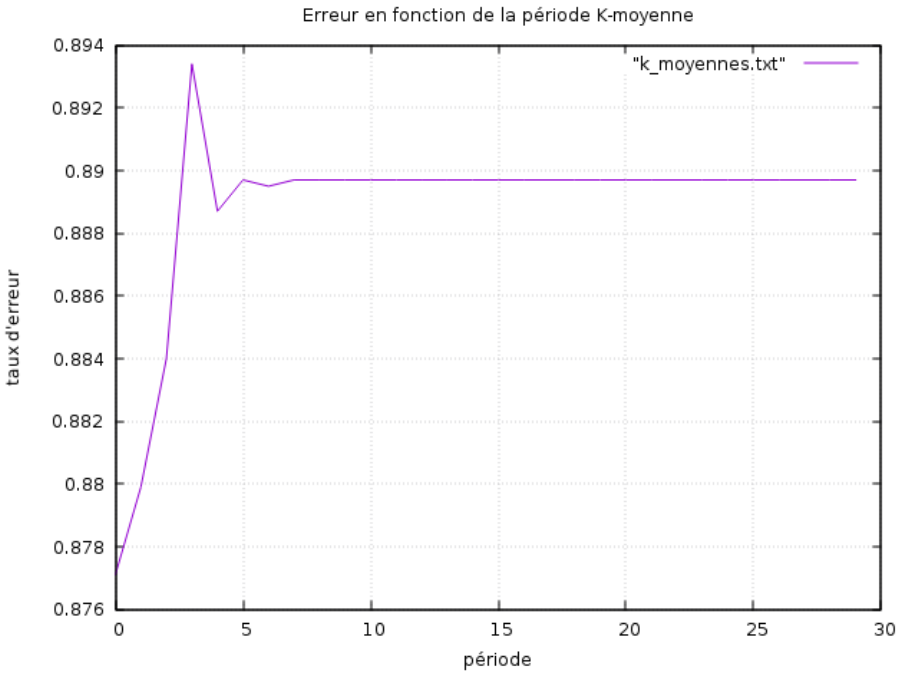

FIG. 3.10: Taux d'erreur globale pour la k-moyenne en fonction de la période

Chapitre 4

Apprentissage de caractéritiques

L'apprentissage de l'approche naïve (valeur de pixels) ne permet pas d'extraire les caractéristiques d'une image. L'extraction de caractéristique est un important pas dans l'apprentissage afin de mieux classifier les images par la suite.

Adam Coates, Honglak Lee et Andrew Y. Ng font part de leur méthode d'extraction de caractéristiques dans leur arcticle « An Analysis of Single-Layer Networks in Unsupervised Feature Learning » [1]. La méthode repose sur les étapes suivantes :

- Extraire des patchs de chaque image du corpus d'apprentissage.
- Collecter l'ensemble des patchs.
- Appliquer une étape de prétraitement sur les patchs (normaliser et réorganiser les données)
- Appliquer un algorithme non supervisé tel que le k-moyenne sur les patchs afin d'en extraire des caractéristiques.
- Organiser les patchs selon un vecteur de caractéristiques.
- Appliquer un algorithme d'apprentissage.
- Extraire de même des patchs, les collecter, les pré-traiter, et les organiser selon un vecteur de caractéristiques pour les images du corpus de test.
- Appliquer un algorithme de classification.

Nous avons donc procédé tel quel avec un nombre de patchs égal à quatre. Afin d'extraire des caractéristiques nous utilisons un algorithme de k-moyenne. Ensuite pour la phase d'apprentissage et de classification nous utilisons SVC (Support Vector Classification) , un algorithme supervisé.

Il faut noter que les problèmes de performances qu'entraîne le pré traitement est notable mais est aussi une étape fondamentale dans la méthode.

Comme dit dans le k-moyenne, l'algorithme est très sensible à l'image référence de départ pour chaque classe. Cela donne des résultats très aléatoires. On a donc utilisé un shuffle afin d'observer les différents résultats pouvant être obtenus. Cette approche est cependant assez satisfaisante bien que les performances ne dépassent pas 70 % de reconnaissance. En effet le faible nombre de patchs utilisés ne permet pas de capturer assez de caractéristiques.

Bibliographie

[1] Adam Coates, Andrew Y Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International conference on artificial intelligence and statistics*, pages 215–223, 2011.

[2] Benjamin Graham. Fractional max-pooling. *CoRR*, abs/1412.6071, 2014.

Annexes : codes utilisées

Voir les pièces-jointes