Projet CIFAR

Nicolas Bock, Camile Skalli

Majeure SCIA-G

EPITA

Contents

1 Introduction et État de l'Art

1.1 Introduction

La base de données CIFAR-10 est un dataset très connu utilisé pour la reconnaissance d'images. Il contient 60 000 images de couleur de 32x32 pixels réparties en 10 classes avec donc 6 000 images par classe. Notre projet est de classifier ces images en utilisant des méthodes de machine learning sans recourir au deep learning.

Le projet est disponible à cette adresse : https://github.com/nicolas-bock/cifar_mlrf

1.2 État de l'Art

Historiquement, la classification d'images a évolué avec l'amélioration des méthodes de machine learning traditionnelles, les premières auxquelles on pense sont les SVM et les k-NN puis ces dernières ont évolué vers les réseaux de neurones convolutifs en raison de leurs performances largement supérieures. Cependant, notre objectif ici est d'explorer les méthodes traditionnelles en utilisant des techniques d'extraction de caractéristiques classiques et des classifieurs connus et reconnus. Parmi ces dernières, on retrouve par exemple SVM, KNN, RandomForest ou Bag of Features qui sont le plus cité. Plusieurs méthodes d'extraction de caractéristiques sont également testées notamment HOG qui est très souvent citée. Les meilleurs résultats tournent autour des 60% de précision. Même si nous sommes bien loin des 95% obtenus avec des réseaux de neurones nous allons essayé de nous en approcher en testant différentes méthodes et combinaisons de méthodes.

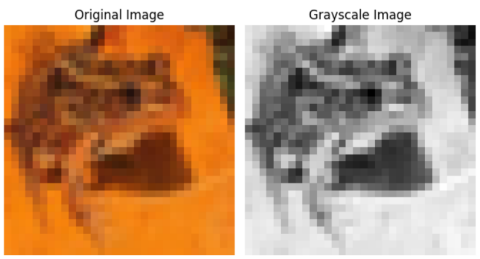
2 Méthode

2.1 Algorithmes d'Extraction de Caractéristiques

Pour notre projet, nous avons utilisé plusieurs méthodes d'extraction de features et nous avons comparé leurs effets sur les performances des différents classifieurs que nous présenterons plus tard. Ces méthodes sont les suivantes:

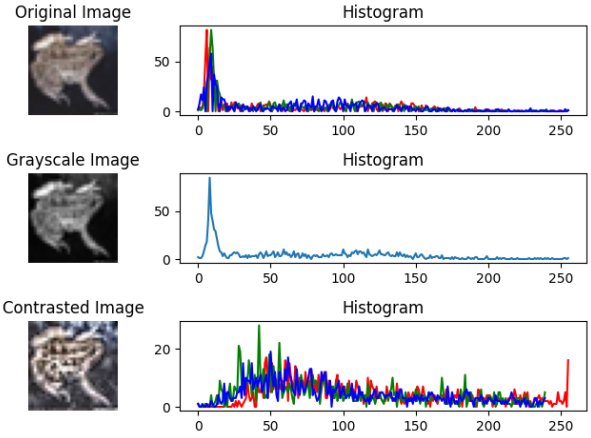
- Flatten: La méthode flatten consiste à transformer chaque image en un vecteur à unidimensionnel en aplatissant les pixels. Nous allons transformer nos images de taille 32x32, qui sont exprimées avec 3 canaux de couleur (donc 32x32x3), en un vecteur de dimension 3072. Cette méthode est simple mais efficace pour obtenir les informations globales de l'image. Le défaut de cette méthode est qu'elle ne préserve pas la structure spatiale de l'image, ce qui peut être un problème quand on effectue des classifications plus compliquées.

- Grayscale: La méthode grayscale convertit l'image en différents niveaux de gris afin de réduire ses dimentions. Ainsi, nous allons passer d'une taille 32x32x3 à 32x32, en enlevant la troisième dimention associée aux couleurs RGB, réduisant la quantité de données à traiter. Cette méthode diminue la complexité mais garde les informations sur la luminosité, laissant la possibilité d'analyser assez facilement les contours et les formes.



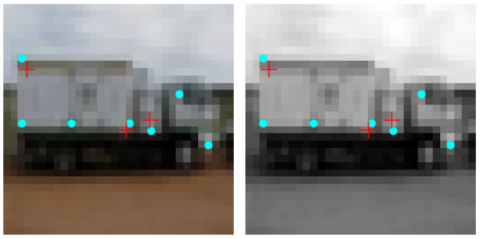
- Contrast: La méthode contrast est assez simple, on ajuste le contraste des images pour améliorer les différences entre les objets et leur arrière-plan, ou encore entre les différentes couleurs des pixels. Cette méthode va augmenter les variations de l'intensité de chaque pixel et permettre de mieux observer les contours des objets sur l'image.

- COLOR HISTOGRAM: L'histogramme des couleurs représentant graphique la distribution des pixels de l'image, selon leur intensité, que ce soit pour des images de deux dimentions (en noir et blanc), ou des images en couleurs (de dimention 3).

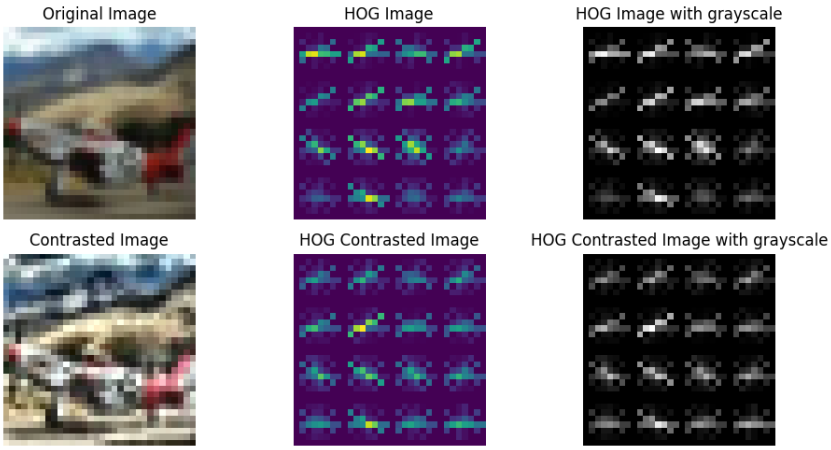


- PCA: L'Analyse en Composantes Principales (PCA) est une méthode pour réduire la dimensionnalité en transformant les données en un nouveau système de coordonnées où les composantes principales retiennent la variance maximale des données. Pour les images, PCA compresse les informations mais garde les caractéristiques essentielles. Ainsi on peut réduire la répétition des images et le bruit sur ces dernières.

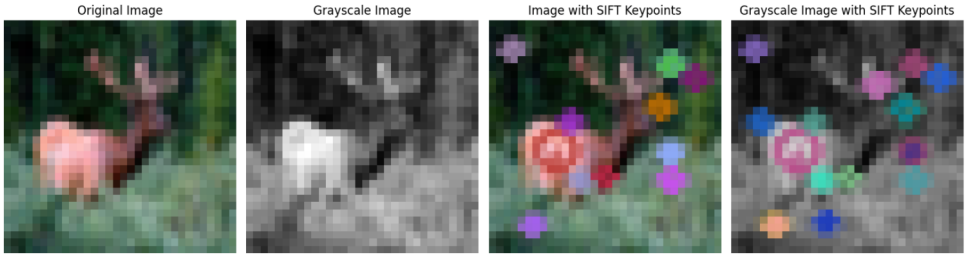
- CORNER DETECTION: La détection de coins est une méthode d'extraction qui identifie les points dans une image où les intensités changent fortement dans plusieurs directions. Les coins sont invariants aux transformations géométriques; il est donc important de les reconnaître lors de la classification d'images.



- HOG: La méthode Histogramme des Gradients Orientés (HOG) va diviser l'image en régions adjacentes de petite taille que nous appellerons cellules. Puis nous calculons pour chaque cellule l'histogramme des directions du gradient pour les pixels à l'intérieur de ces cellules. La combinaison des histogrammes forme alors la valeur du HOG. Cette méthode est une méthode assez réputée qui fournis en général de très bons résultats.



- SIFT: La méthode SIFT est une méthode qui identifie des points clés dans les images et génère des descripteurs qui sont invariants à l'échelle et à la rotation c'est à dire qu'elle reste appropriée même si on effectue des zooms ou des rotations de l'image. On peut ainsi obtenir des caractéristiques locales importantes qu'on utilisera pour la reconnaissance d'objets dans l'image.



- VLAD: La méthode Vector of Locally Aggregated Descriptors ou VLAD combine les descripteurs locaux, comme ceux générés par SIFT, en un vecteur global pour l'image entière. Il capture les distributions spatiales des descripteurs locaux et les agrège en un vecteur unique, offrant une représentation compacte et puissante de l'image.

2.2 Classifieurs

Nous avons ensuite utilisé quatre classifieurs pour évaluer les performances des différentes méthodes algorithmes d'extraction de caractéristiques :

- Régression Logistique

  Pour commencer la régression logistique, c'est une méthode simple et intuitive pour classifier les données. Nous allons utiliser une fonction mathématique (fonction logistique) afin de calculer la probabilité d'appartenir à l'une de nos 10 classes. La fonction de la régression logistique est linéaire, on trace donc une ligne droite pour séparer nos différentes classes. Par conséquent cette méthode est efficace dans les cas où on peut facilement séparer nos classes par une ligne droite. Pour les données comme celles de CIFAR-10, qui sont compliquées à différencier dans certains cas et qui ne sont pas linéaires, une simple séparation linéaire peut ne pas être suffisante.

- Support Vector Machine (SVM)

  Nous avons ensuite essayé la méthode SVM, une méthode puissante qui cherche à séparer les classes en essayant de maximiser la distance entre les points les plus proches de chaque classe. Nous appellerons cette distance maximale la marge. La méthode va donc maximiser cette marge pour obtenir la frontière de décision la plus optimale. Pour les cas de non linéarité, la méthode va utiliser des fonctions spéciales qu'on appelle noyaux. Ces noyaux permettent de transformer les données dans un espace où elles peuvent être séparées par une ligne droite. Cela signifie que même si les données sont compliquées ou ont des formes complexes, le SVM peut les manipuler de manière à les rendre plus faciles à traiter et à classifier.

- K-Nearest Neighbors:

  Nous avons également utilisé le K-Nearest Neighbors ou KNN. C'est un classifieur qui classe un échantillon en fonction des k exemples les plus proches dans l'espace. La classe la plus fréquente parmi ses voisins sera celle qu'on lui attribue. On ne fait pas d'hypothèse sur la distribution des données ainsi la décision n'est faite que grâce à la majorité des voisins les plus proches. Cette méthode est simple et intuitive, mais elle peut être sensible aux données bruitées car les voisins proches incorrects influenceront en mal la décision. De plus cette méthode demande beaucoup de puissance de calcul pour un datastet comme le notre.

- Random Forest

  Enfin le dernier classifieur avec des résultats satisfaisant est le Random Forest, le but de ce classifieur est d'utiliser plusieurs arbres de décision, où chaque arbre est formé à partir d'un sous-ensemble aléatoire des données. Chaque arbre fait une prédiction, et la classe finale est déterminée par la majorité des votes de tous les arbres. Par exemple, si la majorité des arbres prédisent qu'une image est un chat, alors le Random Forest classera cette image comme un chat. Cette méthode est efficace pour traiter un des problèmes principal des arbres de décision: l'overfitting, c'est-à-dire le sur-apprentissage, quand l'apprentissage est trop impacté par les données du jeu d'entraînement. Malgré tout, cette méthode reste elle aussi coûteuse et compliquée à interpréter par rapport à des méthodes comme la régression logistique.

- GradientBoosting

  Une méthode testée mais qui n'a pas abouti est le Gradient Boosting. Celle-ci construit un modèle prédictif en combinant plusieurs modèles simples comme des arbres de décision. La méthode utilise les résidus ou les erreurs des premiers modèles simples lancés pour apprendre des erreurs. Cette méthode est très puissante et est capable de capturer les relations complexes dans notre dataset mais le temps de calcul était beaucoup trop élevé. Nous avons donc décidé d'abandonner cette méthode car elle mettait beaucoup plus de temps que les autres à aboutir.

2.3 Entraînement

Pour la partie entraînement des modèles nous nous sommes basé sur la descente de gradient stochastique (SGD), une méthode d'optimisation qui a pour objectif de minimiser une fonction de perte. Cette dernière mesure la différence entre les prédictions et les vraies valeurs lors de l'entraînement. On commence par initialiser les poids, on divise les données en mini batchs puis pour chacun on calcule le gradient de la fonction de perte par rapport aux poids du modèle. Ce gradient va nous indiquer la direction dans laquelle les poids doivent être ajustés. Cette méthode est très efficace dans notre cas, car nous avons une quantité de données assez élevée et la méthode de SGD est efficace dans ce cas là.

2.4 Définition et Hyper-paramètres

Les hyper-paramètres sont des paramètres externes au modèle et sont ajustés avant de lancer l'entraînement. Voici les définitions pour chaque méthode de classification:

- La régression logistique utilise un paramètre de régularisation et une pénalité lasso ou ridge. La valeur de la régularisation va permettre d'augmenter ou de diminuer l'impact des données sur le modèle. De la même manière la pénalité affecte la manière dont les poids sont régularisés.

- Pour SVM, le premier hyper-paramètre est le type de noyau utilisé, on peut avoir notamment recours à un noyau linéaire, un noyau polynomial ou un noyau RBF afin d'impacter sur la séparation des classes, ce noyau a un paramètre γ. On a ici aussi un paramètre de régularisation qui a le même effet, contrôler la tolérance aux erreurs d'entraînement.

- Dans la méthode K-NN, l'hyper-paramètre le plus important est le nombre de voisin k.

- Pour Random Forest, on doit choisir le nombre d'arbre dans l'ensemble, la profondeur maximale des arbres et le nombre d'échantillons minimum pour chaque feuille. Ces hyper-paramètres sont notamment important pour éviter l'overfitting.

Nous avons aussi eu recours à différentes stratégies multi-classes, par exemple pour la régression Logistique et SVM, nous utilisons la technique One vs Rest. Cette méthode demande d'entraîner 10 modèles différents. Chaque modèle a pour objectif de reconnaître une classe spécifique. Puis chaque modèle donne une probabilité pour que l'échantillon appartienne à sa classe spécifique. La classe finale prédite est celle avec la plus grande probabilité parmi toutes. K-NN et RandomForest utilisent des méthodes de stratégie multi-classes déjà expliquées plus haut.

3 Expérimentations

3.1 Hardware et Software Utilisés

Le projet a été réalisé sur des ordinateurs portables classiques, avec lesquels l'entraînement des modèles pouvait être plus ou moins long selon les performances. Pour le code nous avons utilisé Visual Studio Code et nous avons utilisé Git pour la gestion du code. Les bibliothèques classiques de machine learning ont été utilisées, notamment Scikit-learn pour les algorithmes de classification, NumPy pour les opérations numériques, et Matplotlib pour la visualisation des résultats.

3.2 Analyse de la Base de Données

Comme dit plus haut, les images du dataset sont initialement de taille 32x32x3. Nous avons 10 classes: avion, voiture, oiseau, chat, cerf, chien, grenouille, cheval, bateau, camion. Chaque classe est représentée par 6000 images, ce qui nous fait un total de 60 000 images.

3.3 Pré-traitement

Nous avons commencé par charger le dataset CIFAR-10. Le dataset est initialement divisé en deux ensembles principaux : un ensemble d'entraînement (50 000 images) et un ensemble de test (10 000 images). Dans l'objectif d'avoir un ensemble de validation, nous avons à nouveau séparé l'ensemble d'entraînement avec la proportion 80/20 c'est à dire 40 000 images pour l'entraînement et 10 000 images pour l'ensemble de validation. Les étiquettes des classes sont souvent encodées de manière numérique pour faciliter leur traitement par les modèles. Dans CIFAR-10, les étiquettes sont déjà fournies sous forme d'entiers de 0 à 9, représentant les 10 classes.

3.4 Choix des Hyper-paramètres

Pour choisir les hyper paramètres nous avons comparé les résultats avec différentes valeurs. Nous avons finalement opté pour les valeurs par défaut pour chaque méthode car elles nous donnaient les meilleurs résultats. Voici une explication des hyper-paramètres choisis pour chaque méthode:

- Régression logistique: Le paramètre de régularisation est fixé à 1, et le type de pénalité est fixé à L2 (ridge) c'est à dire la pénalité par défaut, elle permet de réduire le poids des coefficients pour réduire les chances d'overfitting.
- SVM Pour le noyau nous avons pris un noyau RBF efficace dans des cas non linéaires comme pour CIFAR-10. Le paramètre de régularisation est à nouveau fixé à 1 et le γ du noyau est fixé à 'scale' c'est à dire $\frac{1}{n_{feat}}$ donc $\frac{1}{3072}$ car les images sont de taille 32x32x3.
- k-NN La valeur par défaut du nombre de voisin est définie à k=5. On définit la classe à partir des 5 voisins les plus proches.
- RandomForest Pour RandomForest nous prenons 100 arbres pour pas avoir un temps d'execution trop grand. Nous mettons une profondeur maximale de 10 aux arbres, cet hyper-paramètre devrait être modifié si nous remarquions de l'overfitting. La valeur du nombre minimum d'échantillon par feuille est à 1.

4 Résultats

Extraction de feature	Random Forest	SVM	Logistic Regression	KNN
Sans extraction de features	0.47	0.22	0.41	0.34
Flatten	0.42	0.24	0.40	0.33
HOG	0.14	0.11	0.13	0.13
SIFT	0.10	0.10	0.10	0.10
Vlad	0.17	0.11	0.12	0.21
Contrast + Flatten	0.14	0.08	0.13	0.12
Grayscale + Flatten + PCA	0.34	0.16	0.28	0.30
Contrast + SIFT	0.10	0.10	0.10	0.10
Cross-Validation	0.40	0.33	0.37	0.29

4.1 Analyse des méthodes

- Nous pouvons remarquer pour commencer que sans extraction de features, RandomForest et la régression logistique sont les méthodes les plus efficaces. Ce résultat vient probablement du fait que ces méthodes peuvent gérer des données bruts de manières efficace. SVM et KNN ont plus de mal car, ce sont des méthodes sensibles aux dimensions élevées du dataset.
- Pour flatten, on remarque que les résultats sont très proches de ceux sans extraction de features, nous montrant que cette méthode ne change pas grand chose à elle seule.
- HOG et SIFT présente de très mauvais résultats pour toutes les méthodes. Ces méthodes ne sont probablement pas adaptées à gérer les détails dans nos images car elles sont trop petites. En effet ces méthodes ne peuvent pas capturer suffisamment de détails dans des images de cette taille. De plus, la variation et la complexité des classes peuvent dépasser les capacités qu'on nos méthodes à trouver les différences.
- VLAD présente des performances à peine meilleure, notamment pour le KNN. Le type de données sur lequel nous travaillons n'est peut être pas idéal pour cette méthode d'extraction de feature.
- L'ajout de contraste combiné à l'aplatissement ou à SIFT apporte de très mauvaises performances, nous laissant penser que la méthode contrast n'est pas rentable.
- La combinaison grayscale, flatten et PCA nous donne des résultats plutôt acceptable. Même si plus faible que la moyenne, on peut expliquer ces résultats par la perte d'informations importantes comme la couleur, affectant négativement les performances.
- Enfin la Cross-Validation nous donne des résultats satisfaisant.

En résumé, les meilleurs classifieurs sont RandomForest et LogisticRegression. Ces résultats montrent que les méthodes classiques de machine learning atteignent une performance limitée sur CIFAR-10 sans utiliser les bonnes méthodes d'extraction de caractéristiques.

5 Conclusion et Améliorations

Ce projet nous a montré que malgré l'absence de deep learning, des méthodes traditionnelles de machine learning peuvent nous permettre d'obtenir des résultats satisfaisants pour la classification d'images. De plus, nous avons pu avoir des résultats plutôt cohérents, par exemple nos classifieurs confondaient souvent les bateaux, les avions et les camions ainsi que les animaux entre eux, comme on peut le voir dans les matrices de confusion en annexe (3). Ces résultats sont logique car dans des petites images comme celles de CIFAR 10, la différence entre des véhicules et des animaux est très fine, vu la faible résolution des images.

Enfin, nos méthodes restent évidemment moins performantes que les réseaux de neurones et notamment les CNN. Les améliorations futures pourraient inclure l'optimisation des hyper-paramètres, l'utilisation de techniques d'ensembles plus sophistiquées et l'intégration de pré-traitements avancés que nous ne connaissions pas forcément.

6 Bibliographie

1. La base de données CIFAR : https://www.cs.toronto.edu/~kriz/cifar.html
2. Structure du projet (Cookiecutter Data Science) : https://cookiecutter-data-science.drivendata.org
3. Classification binaire et multi-classe : https://kobia.fr/classification-metrics-multi-class-simple/#:~:text=Cas%20multi-classe%20%3A%20une%20probabilit,d%27tre%20lever%20de%20Soleil
4. Preprocessing de la donnée : https://scikit-learn.org/stable/modules/preprocessing.html
5. Comparaison entre les classifier : https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html
6. Histogram of Oriented Gradients (HOG) : https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html
7. Colormap : https://matplotlib.org/stable/users/explain/colors/colormaps.html
8. Histogramme de couleurs : https://datacorner.fr/image-processing-2/
9. Corner Detection : https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_corner.html#sphx-glr-auto-examples-features-detection-plot-corner-py
10. Méthode de contraste sur l'image :
11. Lab color space : https://www.xrite.com/blog/lab-color-space
12. Code : https://stackoverflow.com/questions/39308030/how-do-i-increase-the-contrast-of-an-image-i
13. Méthode Vlad : https://github.com/tim-hilt/vlad/blob/master/vlad/vlad.py
14. Exemple de détections de features : https://scikit-image.org/docs/dev/auto_examples/features_detection/index.html

7 Annexe

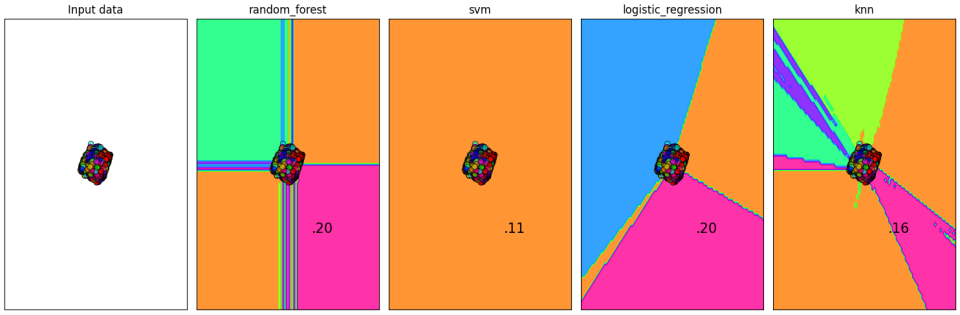


Figure 1: Comparaison des classifieurs en format 2D

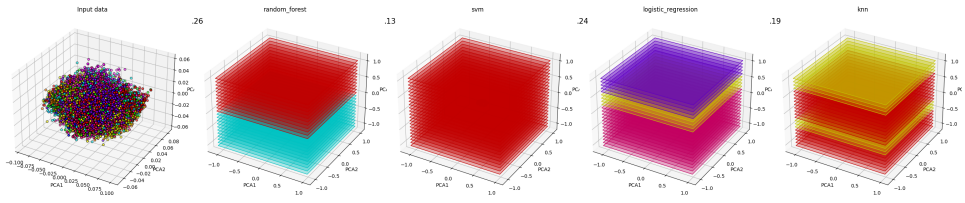


Figure 2: Comparaison des classifieurs en format 3D

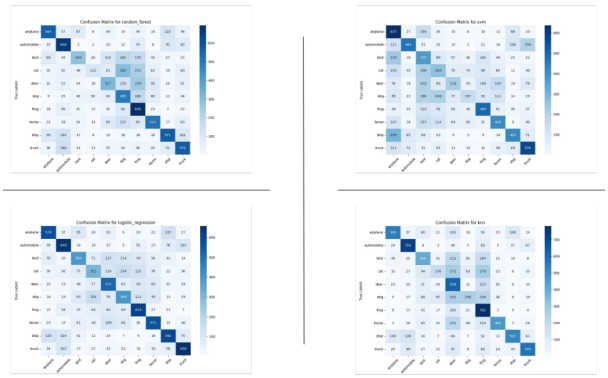


Figure 3: Matrice de confusion avec flatten et HOG

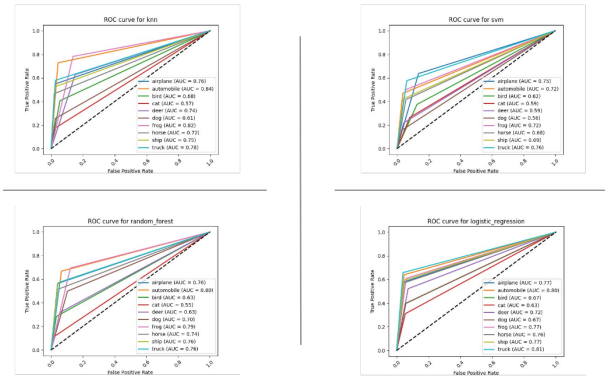


Figure 4: Courbes ROC associées aux différents classifieurs