PROJET : DÉTECTION DE VISAGES

Anh Tu NGUYEN - Groupe 13

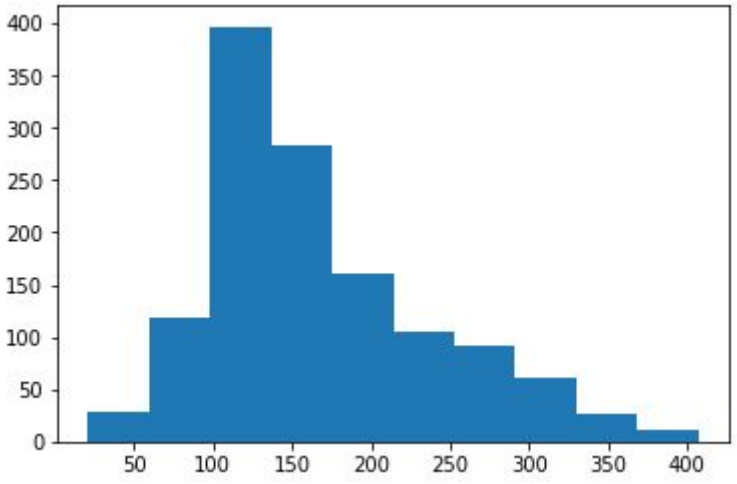
I. Exploitation et prétraitement de données:

1. Exploitation de données d'apprentissage et de données de test:

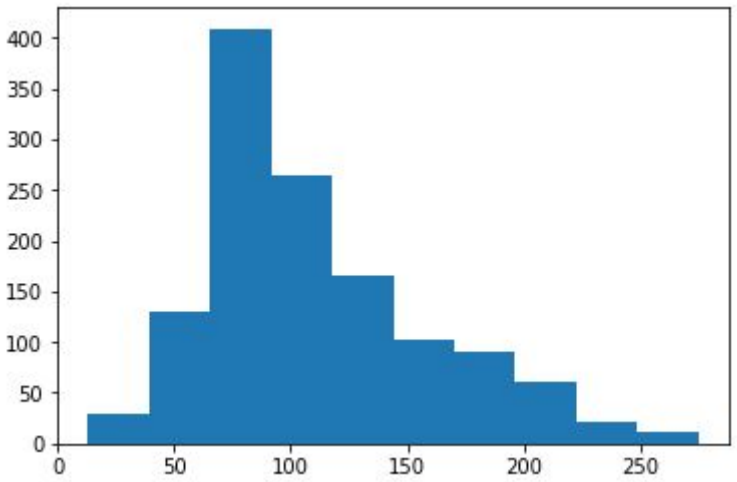
Le jeu de données dans ce projet comporte de 2 ensembles de données comme d'habitude : un ensemble d'apprentissage et un ensemble de test.

L'ensemble d'apprentissage contient au total 1000 images et un fichier "label.txt". Les images d'apprentissage sont des images sous format RGB, il y a un ou plusieurs visages sur chaque image. On fait des zooms sur le fichier "label.txt". Il y a 5 numbres dans chaque ligne de ce fichier. Ils sont le numéro de l'image, les coordonnées (ligne, colonne) du coin supérieur gauche de la boîte de visage et la taille (hauteur, largeur) de la boîte. Avec ces informations, on pourra récupérer des boîtes de visages dans chaque image de l'ensemble d'apprentissage pour apprendre notre classifier de visage.

L'ensemble de test contient 500 images en couleur. Chaque image peut avoir plusieurs visages. Il y a quelques images sans visages. Il y a 2 colonnes à exploiter qui sont assez importante : hauteur et largeur. Le choix de la taille de boîte dans le prochain étape (générer des modèle de classifieur) dépend des valeurs de ces 2 colonnes. On calcule les valeurs moyennes : hauteur_moyenne est 167 pixel et largeur_moyenne est de 110 pixels. Ensuite, on construit un histogram des valeurs de ces 2 colonnes. Il est évident que les valeurs de l'hauteur de la boîte sont concentré dans l'intervalle [100,150] et que les valeurs de la largeur de la boîte sont concentré dans l'intervalle [80,100]



Histogram des hauteurs de boîte d'image dans l'ensemble d'apprentissage



Histogram des largeurs de boîte d'image dans l'ensemble d'apprentissage

2. Prétraitement de données d'apprentissage:

Pour pouvoir générer des model de classifieur de visages, il faut d'abord générer des données d'apprentissage. Comme les modèles qu'on veut sont des classifieurs qui pourront classifier un image à un de 2 classes : visage ou non visage, des données d'apprentissage doivent être des images de visages. Ces images doivent être plus proche que possible pour que les modèles puissent bien apprendre. Grâce aux informations dans le fichier "label.txt", on peut trouver des boîtes de visage (un visage par boîte) dans chaque image.

Pour simplifier la tâche de calcul, on transforme toutes les boîtes de visage aux images noir et couleur. C'est-à-dire au lieu de faire des calculs sur 3 dimensions (RGB), on ne doit faire que maintenant des calculs sur 2 dimensions.
Ensuite, on redimensionne les boîte de visage à la taille fixe que on a choisit : hauteur largeur . On a maintenant des données d'aprentissage positives.

Pour des données d'apprentissage négatives, il est simple de générer aléatoirement 10 boîtes par image au niveau de position de coin supérieur gauche et au niveau de taille. On a totalement 10 000 images d'apprentissage négatives. Nous utilisons d'abord cet ensemble d'apprentissage pour générer un premier classifieur de visage.

II. Génération des classifiers de visages:

On décide de générer quelques modèle de classifieurs de visages par SVM avec des kernels différents (ex : linéaire, RBF - Radial Basic Function, sigmoid) et par AdaBoost. On choisit ces modèles pour réaliser parce qu'ils sont des modèles assez

sensibles aux données brutées. Par ailleurs, ces modèles sont éfficaces avec les jeux de données à grande dimension, notamment le SVM.

On a 1284 boîtes de visage et 10 000 données négatives dans l'ensemble d'appretissage. La validation croisée est utilisée pour évaluer les modèle de classifieurs. Les taux d'erreur obtenus par la validation croisée sont toujours très faibles parce que le nombre de données négatives est très dominant que le nombre de boîtes de visage. On génère les premiers modèles en utilisant une partie de cet ensemble comme des données d'apprentissage et puis détecte des visages dans l'ensemble de validation (une autre partie de cet ensemble). On récupère des résultats Faux Négative et les rajoute dans l'ensemble d'apprentissage et puis recommence à construire les modèles de classifieur. D'après des exercices dans TD, on a observé que la transformation de l'image original au HOG (Histogram Oriented Gradients) améliorera l'apprentissage des modèles. En effet, il y a des forts changements des pixels au contour d'objets et le HOG peut reconnaître ces changement. Par conséquence, on applique la transformation du HOG à l'ensemble de données d'apprentissage.

A la fin du projet, si on a assez de temps, on essayera de générer un réseau de neurones simple pour améliorer la performance du détecteur.

III. Réalisation d'un détecteur de visages par méthode "Sliding Window":

Cette méthode consiste à passer une boîte pré-definite toute l'image. A chaque fois, la boîte récupérée est classifié par un modèle de classifieur de visage qu'on a généré dans la partie précédente. Comme l'objet à détecter a peut-être plusieurs tailles donc la taille de la boîte doit varier. Pour simplifier la tâche de programmation, on décide de créer une pyramide d'images et réaliser la détections sur tous les images dans la pyramide. Enfin, puisque un visage peut être détecté plusieurs fois, il faut supprimer des doublons et conserver des boîtes poosèdant meilleurs scores.

1. Définition de la boîte

Pour bien choisir la taille d'une boîte, des informations que on a analysées dans la première partie sont importantes. Comme la taile moyenne des boîtes dans ensemble de données d'apprentissage est 110 pour largeur et 165 pour l'hauteur, et selon le histogram de l'hauteur et de la largeur des boîtes dans l'ensemble d'apprentissage et parce que l'hauteur de la boîte est normalement 1.5 fois plus que

la largeur, on peut choisir la largeur de la boîte est de 110 pixels et l'hauteur de la boîte est de 160 pixels.

2. Réalisation de "Sliding Window"

Un autre paramètre qui est assez important est le nombre de pixels à avancer après chaque étape. Dans un premier temps, on choisit la saute de 30 pixels à chaque étape pour finaliser l'algorithme "Sliding Window". Dans un second temps, on essayera plusieur autres valeurs pour pouvoir choisir une valeur qui maximise la performance du détecteur mais aussi ne ralentit pas la vitesse du détecteur.

A chaque image dans la pyramid d'images, on fait passer la boîte dans tout image. Pour chaque fenêtre recupérée par la boîte, on calcule son HOG et la classifie dans 2 classes : visage et non visage. Si la fenêtre est bien visage, on sauvegarde le numéro de l'image, ses coordonnées du coin supérieur gauche, la taille de la boîte et le score. On remarque que les coordonnées du coin supérieur gauche et la taille de la boîte sont celles-ci de l'image redimensionné dans la pyramid d'image. Par conséquent, on les transforme aux coordonnées et à la taille de l'image original avant les sauvegarder. Ces résultats seront traités avant passer la partie de l'évalutaiton du détecteur.

Quelques visages détectés par le détecteur de visage avec Sliding Window et SVM

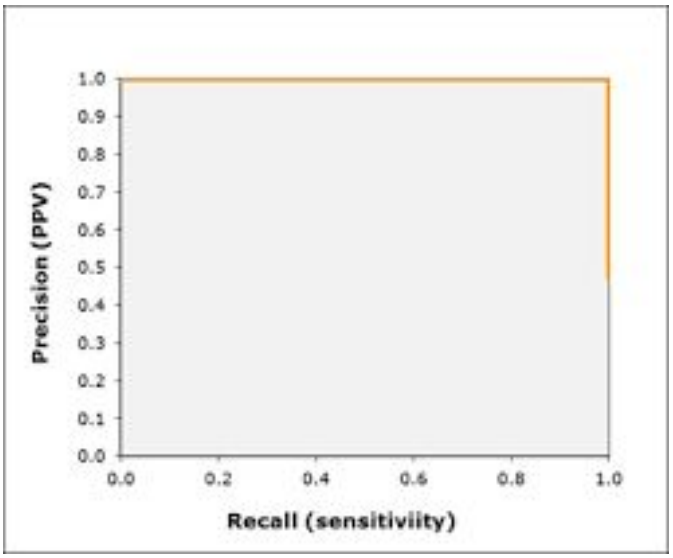
3. Suppression des doublons

Comme un visage peut être détecté plusieur fois, on réalise une fonction qui supprimera des doublons de boîtes. On d'abord trie l'ensemble de boîtes par ordre décroissant de score. Pour chaque paire de boîte, on calcule l'aire de recouvrement par la formule : $\frac{Aire(Bi \cap Bj)}{Aire(Bi \cup Bj)}$. Si l'aire de recourvrement est supérieur à 0.5, on supprime la boîte ayant plus faible score. Après cet étape, on a un ensemble de résultats moins lourd et il est l'ensemble de résultats final du détecteur.

IV. Evaluation:

Dans cette phase, on réalise des fonctions pour construire le courbe Précision/Rappel de chaque détecteur. La courbe Précision/Rappel nous permet d'évaluer la performance des détecteurs. Nous avons 4 éléments pour évaluer la performance d'un détecteur : la Précision, le Rappel, le score F1 et l'aire sous

courbe Précision/Rappel. Les fonctions que on réalise donne ces 4 éléments à partir d'un fichier de résultats. La courbe Précision/Rappel idéal est représenté dans la figure ci-dessous.



La courbe Précision/Rappel idéal

On utilise les paramètres calculés par des fonctions pour évaluer les détecteurs de visages générés. On a d'abord fixé la taille de la boîte (110 pour largeur et 160 pour hauteur) et le nombre de pixels de passage chaque étape (30 pixels). Cela nous permet de comparer les performances des détecteurs avec des modèles classifieurs différents. On a le tableau de résultats ci-dessous:

Modèle	**Précision moyenne**	**Rappel moyenne**	**Score F1 moyenne**	**Aire sous la courbe P/R**
Linéaire SVC	0.6306	0.2144	0.2943	0.2347
SVC avec kernel RBF	0.6592	0.4043	0.4459	0.3198
SVC avec kernel sigmoid	0.2241	0.2149	0.2005	0.0854
AdaBoost	0.6914	0.2450	0.3308	0.3023

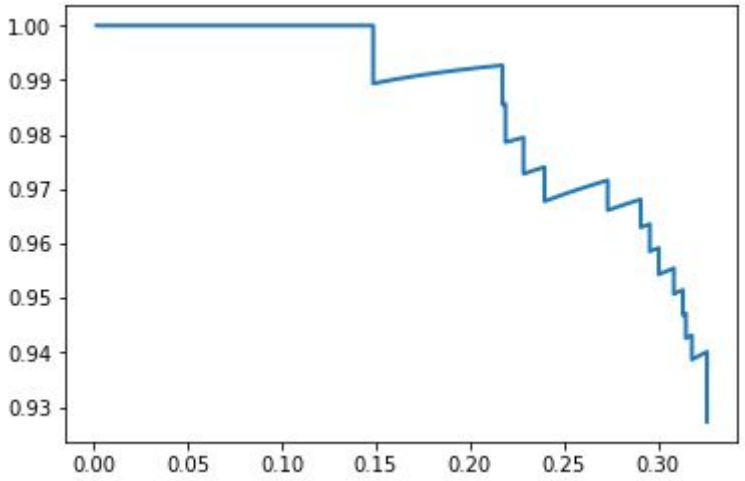
D'après ce tableau, le modèle SVC avec Radial Basic Function comme le kernel nous donne le meuilleur score moyenne. AdaBoost classifie des visage un peu mieux que SVC avec RBF mais le rappel moyenne est très faible par rapport au rappel moyenne de SVC. C'est-à-dire Adaboost détecte moins de vrais visages dans l'ensemble de test que SVC. Par conséquent, on choisit le modèle SVC avec le kernel RBF pour continer à améliorer la performance de détecteur.

Après avoir choisi le modèle de classifieur de visage, on continue à améliorer le détecteur de visage en optimisant un paramètres : le nombre de pixels à passage après chaque étape. On travaille d'abord sur le nombre de pixels à passage après chaque étape. J'ai crée les images corresponds aux boîtes de visage détectés par le détecteur. J'ai l'impression qu'il y a quelques images où les visages sont assez clairs mais le détecteur ne peut pas les détecter. Il y a des points communs : dans ces images, les visages sont assez grands par rapport à la dimension des images. Dans l'image original, c'est évident que la boîte ne peut pas récupérer tout le visage. Dans les images redimensionnés, le nombre de pixels à passage après chaque étape est important. J'essaie donc de diminuer ce nombre de pixels à 10 pixels. Ce ralentira la vitesse de détection mais on peut obtenir un meuilleur détecteur. On recrée les images corresponds aux boîtes détectés dans l'ensemble de test et le détecteur maintenant détecte mieux.

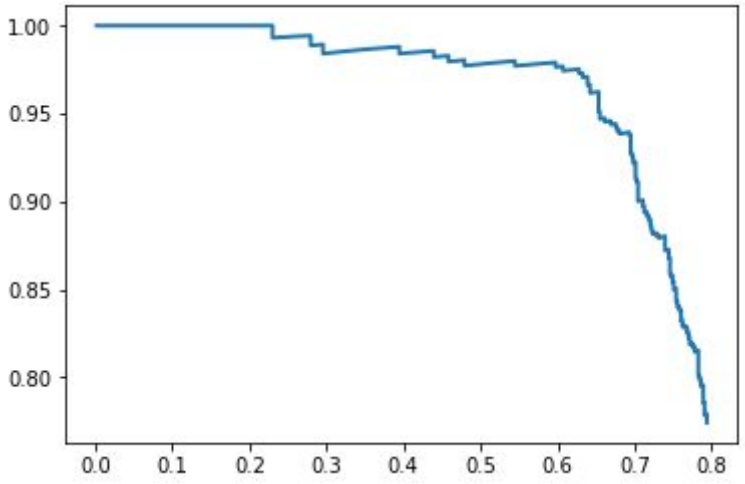
Je continue à observer des nouveaux images corresponds aux boîtes de visage détectés par le détecteur. Il y a encore des visages mal détectés ou des boîtes sont plus grand par rapprot aux visages. Donc je pense que la cause de ces problèmes est que la boîte glissante n'est pas en "bonne" dimension et je pourrais améliorer la performance du détecteur en redéfinissant la boîte glissante. En regardant sur le histogram des valeurs de largeur de la boîte dans l'ensemble d'apprentissage, pourtant la valeur moyenne de la largeur est environ 110 pixels, des valeurs sont plus concentrées dans l'intervale [70,90]. Donc on choisit de redéfinir que la largeur de la boîte est de 90 pixels, l'hauteur de la boîte est environ 1.5 fois plus que la largeur donc 135 pixels. Après avoir changé des paramètres importants du détecteur, on reprend l'apprentissage de ce modèle sur 500 premiers images d'apprentissage et évalue sa performance sur les restes images d'apprentissage. On obtien une amélioration remarquable présentée dans le tableau ci-dessous. Les courbes Précision/Appel du détecteur avant et après le changement des paramètres sont aussi représentées.

Modèle	**Précision moyenne**	**Rappel moyenne**	**Score F1 moyenne**	**Aire sous la courbe P/R**
Avant	0.6592	0.4043	0.4459	0.3198
Après	0.9475	0.4697	0.5771	0.7684

J'applique le détecteur sur l'ensemble de test pour obtenir un fichier de résultats et tester sur le site "Tableau de score" sur Moodle. Le résultat obtenu est : Précision : 79.18%, Rappel : 80.92%, score F1 : 80.04% et la précision moyenne est 78.52%.



Courbe Précision/Appel avant le changement des paramètres



Courbe Précision/Appel après le changement des paramètres

V. Conclusion

Dans le cadre de ce projet, j'ai des chances d'apprendre la méthode "Sliding Window" appliqué en détection de visage. Un modèle d'apprentissage automatique est aussi entrainé et appliqué comme la classifieur de visage - une partie importante du détecteur. Pour améliorer la performance du détecteur, j'ai enrichi des données d'appentissage par des résultats Faux Positive et ajusté des paramètres importants. Le détecteur obtient la précision moyenne de 78.52% sur l'ensemble de test et il est un résultat acceptable.

Annexes : Les livrables rendus

1. Codes :

Il y a deux fichiers python dans le livrable : train.py et test.py.
train.py : générations des données d'apprentissage pour le modèle (exemples positives et négatives), génération du modèle de classifieur, sauvegarde du modèle au fichier *SVC_RBGF_model.pkl* .
test.py : récupération du modèle de classifieur à partir du fichier SVC_RBGF_model.pkl, réalisation de Sliding Window, suppression des doublons et export des résultat au fichier result_test_SVC_RBF.txt

2. Autres fichiers:

SVC_RBGF_model.pkl : le modèle de classifieur de visage
result_test_SVC_RBF.txt : le fichier de résultats
rapport.pdf : rapport du projet qui détaille tous les étapes réalisés pour construire un détecteur de visage