



Compte rendu 4

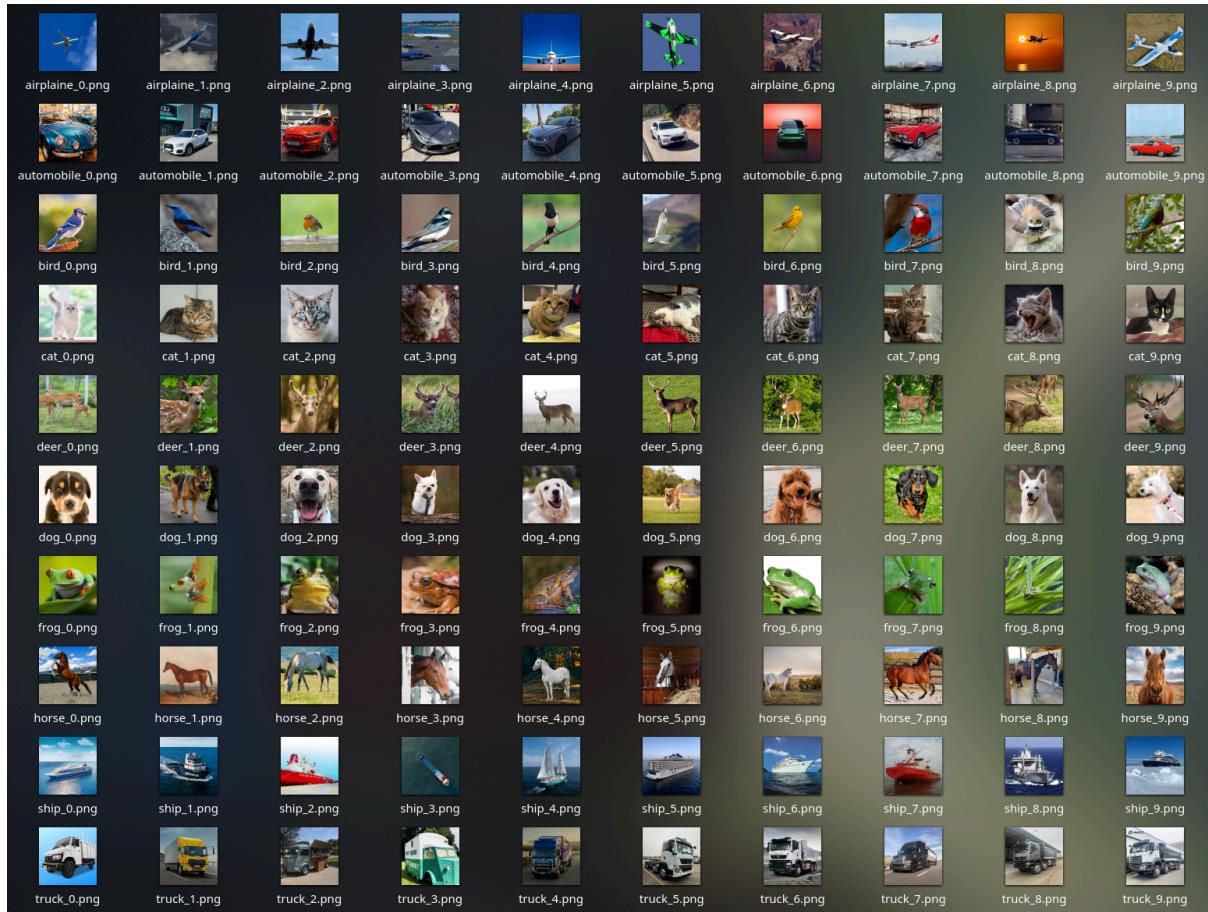
BELDJILALI Maxime, CHATEAUNEUF Arthur
13 October 2024

| | |
|----------------------------------|----------|
| Obscurator | 2 |
| Mise en place de la base de code | 2 |
| Premiers filtres naïfs | 2 |
| Pistes pour la semaine prochaine | 2 |
| Annexes | 3 |
| Références | 3 |
| Dépot Github | 3 |

Obscurator

Nouvelle Base de test

Afin de tester efficacement nos filtres nous avons décidé de télécharger un panel de 100 images de haute résolution que nous avons recadré en 512x512 pixels. Nous avons ainsi 10 images pour chaque catégorie de CIFAR-10 : Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship et Truck :



Nous avons également mis à jour le makefile afin d'appliquer en parallèle un programme de filtre au choix à l'ensemble des données de tests. Cela nous permet d'appliquer nos 5 filtres actuels sur l'ensemble des données de test en ~3 secondes.

Nouveaux filtres

Nous avons amélioré les filtres que nous avions déjà afin de les rendre plus rapides et prononcés. Nous avons également ajouté 2 nouveaux filtres :

- NAIVE WAVE DIVIDE : Filtre naïf effectuant la division de chaque pixel avec le précédent, le tout dans l'intervalle de valeur [-1, 1]. Ce filtre atténue grandement tous les détails de couleurs, tout en laissant apparent certains des contours et textures utiles à la reconnaissance du sujet.

- NAIVE LINE NOISE : Nous appliquons un bruit se régénérant tous les 128 pixels en longueur. Lorsqu'un observateur humain regarde l'image de loin, le sujet et les détails sont très visibles. Cependant, une grande partie de l'information de contour et de couleur est perdue.

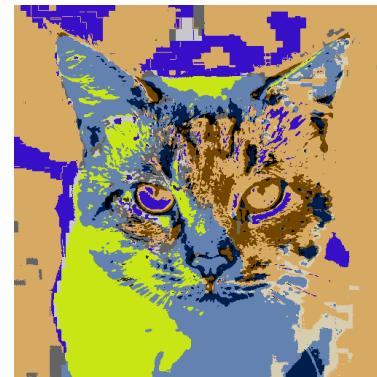
Voici un exemple de chacun de nos filtres naïfs :



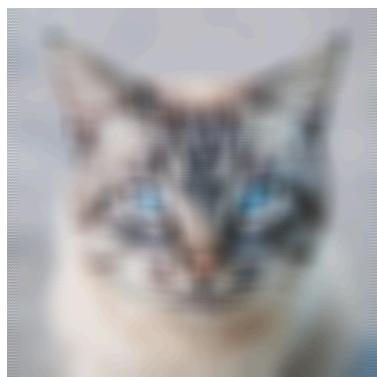
Image d'entrée



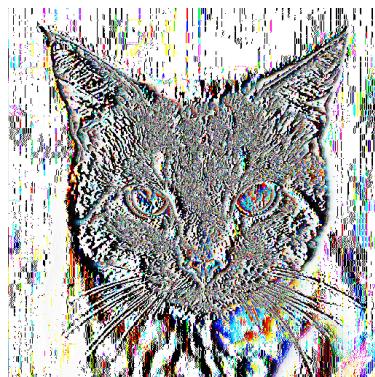
NAIVE LSB CUT



NAIVE LSB CUT [YCrCb]



NAIVE HEAVY BLUR



NAIVE WAVE DIVIDE



NAIVE LINE NOISE

Pistes pour la semaine prochaine

Lors de la prochaine semaine, nous souhaitons pouvoir tester de façon précise l'efficacité de ces filtres pour l'obscurcation ainsi qu'en créer de nouveaux, potentiellement réversibles et inspirés de nos résultats de tests.

Reconnaissance par CNN

Nouveau modèle :

La semaine dernière, nous avions une précision sur notre jeu de test de 70% avec notre CNN. Cela était dû à l'architecture trop simpliste du modèle de CNN. Afin d'obtenir de meilleurs résultats, nous nous sommes penchés sur des architectures déjà existantes dans la littérature et notamment une architecture de type VGG avec VGG16 [4]. Voici comment cette architecture se présente :

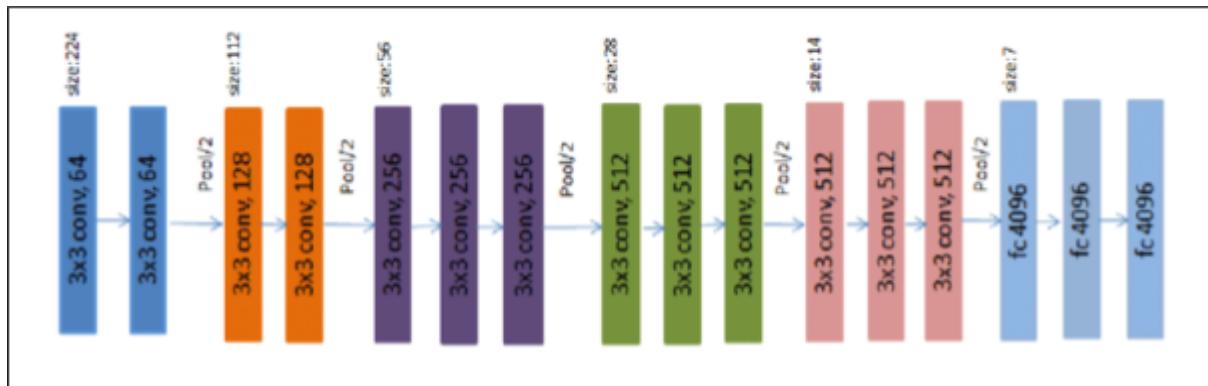


Schéma de l'architecture de CNN VGG16 [5]

Ce modèle inchangé attend des images de taille 224x224 en entrée. Or, nous faisons notre entraînement sur le dataset cifar 10 et les images de ce dataset ont une résolution de 32x32. Ainsi, nous devons choisir entre changer de dataset ou bien adapter cette architecture. Nous avons fait le choix de l'adapter à notre dataset. Pour cela, nous avons suivi une architecture "VGG like" [6].

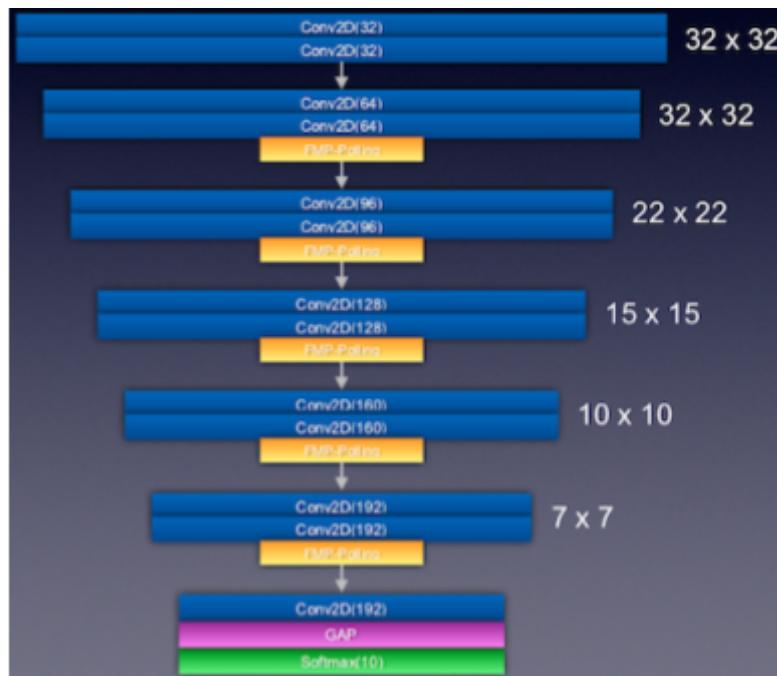


Schéma architecture VGG like [7]

Avec cette architecture, pour un entraînement de 64 époques et des tailles de batch de 64. Nous obtenons une précision sur le jeu de test du dataset cifar 10 de 90,42%.

Détection de régions d'intérêts :

Afin d'améliorer la robustesse et la précision de notre modèle. Nous souhaitons mettre en place une phase de pré-traitement des régions d'intérêts. C'est-à-dire déterminer des zones dans l'image où il sera plus probable d'avoir notre objet à reconnaître. Ainsi, cela facilitera la reconnaissance pour notre CNN, mais cela nous permettrait d'envisager la reconnaissance de plusieurs objets pour une seule image. Par exemple, la semaine passée nous avions cet exemple de reconnaissance de chat :



Image de chat reconnu comme appartenant à la classe Chat



Image de chat reconnu comme appartenant à la classe Avion

Ici l'objet que nous souhaitons détecter est un chat et dans le cas de la seconde image, il n'est pas reconnu. Nous pouvons supposer que cela est dû au fait qu'il n'occupe qu'une petite portion de l'image et que notre dataset ne contient que des images où l'objet occupe une large portion de l'image.

Une solution pourrait être l'utilisation de R-CNN tel que Yolo [8]. En faire un état de l'art plus poussé est un objectif pour la semaine prochaine.

Objectif pour la suite :

L'objectif pour la semaine prochaine est d'établir un état de l'art des méthodes de détection des régions d'intérêts et potentiellement l'implémentation d'une de ces méthodes. Un autre objectif est d'affiner notre modèle de CNN avec un entraînement plus conséquent. De plus, il serait intéressant de tester ses performances sur un jeu de données obscurci.

Annexes

Références

[1] - STB Image - Librairie C++ - <https://github.com/nothings/stb>

[2] - Colorm - Librairie C++ - <https://github.com/soreja/colorm>

[3] - Cifar-10 - Dataset - <https://www.cs.toronto.edu/~kriz/cifar.html>

[4] - Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1556.

[5] - Schéma VGG16 -

https://www.researchgate.net/figure/portrays-the-VGG16-model-for-ImageNet-40-It-has-13-convolutional-layers-and-three_fig2_331562880

[6] - Architecture VGG like - <https://github.com/laplacetw/vgg-like-cifar10>

[7] - Schéma VGG avec Fractional Max-Pooling -

https://github.com/laplacetw/vgg-like-cifar10/blob/master/model_summary/fmp_cifar10.png

[8] - Redmon, J., Divvala, S.K., Girshick, R.B., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779-788.

Dépot Github

- <https://github.com/Pataeon/Securite-visuelle>