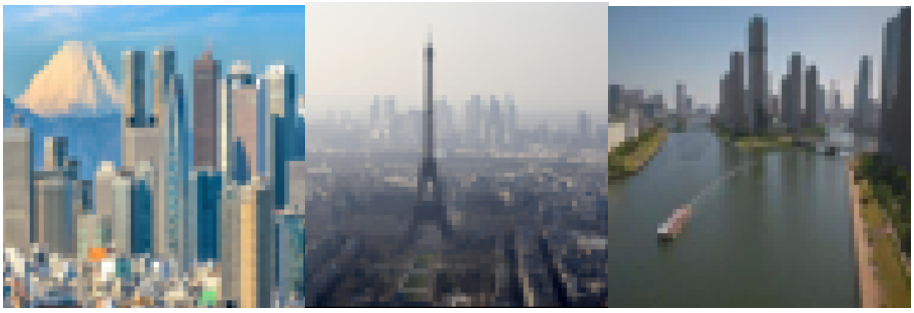


Rapport Final Projet Apprentissage Profond

Anaclet Paul
Cormary Nicolas
Crouzet Sylvain
Desprats Théo

Département Sciences du Numérique - 2A SIL
2021-2022

Table des matières

1 Présentation du sujet

1.1 Informations générales

Le projet “Capitals Learning”, dont il est question dans ce rapport, sera porté par le groupe 1 des L1-L2, composé de Paul Anaclet, Nicolas Cormary, Sylvain Crouzet et Théo Desprats. Nous allons, dans la suite de ce rapport, décrire le sujet de ce projet ainsi que les méthodes employées pour constituer la base de données (BDD) relative à celui-ci.

1.2 Description du sujet

L’objectif de ce projet est d’entraîner un réseau de neurones (perceptron multicouches) pour la classification d’images de villes et notamment de capitales du monde. Comme montré dans la Figure 1 ci-dessous, l’intelligence artificielle devra, à partir d’une image d’une rue ou d’un monument, reconnaître et prédire dans quelle ville se situe cette rue ou ce monument.

FIGURE 1 – Représentation des résultats attendus

2 Base de donnée

2.1 Acquisition et annotation des données

Nous avons pu acquérir une première partie des données en “scrappant” des images provenant de Google Images grâce au script suivant écrit par l’utilisateur “ohyicong” sur GitHub : `https://github.com/ohyicong/Google-Image-Scraper.`

Ce script nous a permis d’obtenir des images de villes à l’aide des mots clés de recherche suivants : “[nom de la ville]”, “[nom de la ville] ville” et “[nom de la ville] rue” que nous avons stockées dans notre BDD hébergée sur GitHub : `https://github.com/TCD32/Projet-CapitalsLearning/tree/main/data`

L’annotation s’effectuera simplement en associant à l’image le nom de la ville à laquelle elle appartient par le biais du script de chargement fourni et légèrement remanié disponible sur le lien suivant : `https://github.com/TCD32/Projet-CapitalsLearning/blob/main/src/features/load_data.py`

Le nombre de classes de résultats différentes (villes) sera initialement peu élevé le temps de la calibration puis nous comptons l’augmenter au fur et à mesure et observer l’évolution des résultats.

2.2 Partition des images en ensembles d'entraînement / validation / test

Nous comptons répartir les images de la manière suivante pour chaque classe d'image différente : environ 70% dans 'training', environ 15% dans 'test' et environ 15% dans 'validation'. Comme nous avons à peu près 500 images par ville, nous pensons prendre environ 375 images pour l'entraînement du réseau, en nous assurant d'avoir des images de nuit comme de jours, mais aussi de monuments, comme la tour Eiffel et de quartiers.

2.3 Résultats attendus

Pour ce qui est de la prédiction de la ville lorsque certains bâtiments, monuments, emblématiques sont présents nous nous attendons à de bons résultats. La reconnaissance de Paris grâce à la tour Eiffel ou de l'arc de triomphe semble facile, comme celle de Tokyo grâce à la tokyo tower, ou New-York grâce à Times Square.

En revanche, nous sommes beaucoup moins confiant pour la reconnaissance des quartiers, les rues de Tokyo et de New-York notamment sont très récentes, compactes et bétonnées, donc les ressemblances sont moins évidentes qu'elles pourraient l'être avec Paris où il y a davantage de bâtiments à base de pierre.

Nous avons également noté certaines images qui si elles sont placées dans la base de donnée de vérification ou de test seront assurément fausses, comme par exemple celle-ci qui représente le Washington Square Arch, avec derrière le One World Center, et nous avons peu d'espoir que le réseau de neurone reconnaisse ce dernier derrière, et ignore l'apparent arc de triomphe au premier plan, fig 2.

FIGURE 2 – Le faux arc de triomphe

De plus, nous avons deux types d'environnement bien distincts pour chaque ville. En effet, il y a majoritairement des photos aériennes des villes ou bien des photos des rues qui les composent. Cela mènera sûrement à de moins bons résultats que si on ne s'était concentré que sur une seule de ces deux catégories.

3 Résolution du problème

Comme ce réseau de neurone a pour but la reconnaissance d'image il était évident que nous utiliserions un modèle de réseau convolutif, pour avoir un minimum de paramètres. Pour réaliser un tel réseau nous avons utiliser les bibliothèques tensorflow, et plus particulièrement son package Keras, numpy pour manipuler plus facilement les images, et matplotlib.pyplot pour pouvoir afficher les images et les courbes pour exploiter plus facilement les résultats obtenus. De plus les images ayant des formats presque tous différents nous les redimensionnerons pour qu'elles suivent le format 64x64 pixels.

3.1 Premier réseau

Le premier réseau que nous avons réalisé a été exactement le même que le premier étudié en TP pour l'étude des images. Ce dernier comporte 4 couches filtrage par convolution de fenêtres 3x3 en utilisant la fonction d'activation ReLU séparées par des couches de Pooling 2x2, cf figure 3. Enfin nous avons une couche dense de 512 neurones avec la même fonction d'activation que précédemment, et une couche de 3 neurones pour déterminer la classe de l'image. Cette architecture nous donne un réseau de 449 699 paramètres, tous modifiables. Pour ce réseau, comme pour les suivants, nous utiliserons comme fonction de perte la "spare-categorical-entropy" de Keras, comme nos images ne sont associées qu'à une seule classe, et un taux d'apprentissage de 3e-4, une valeur relativement standard, la métrique choisie sera 'accuracy', parce que certaines autres pouvaient poser des problèmes et qu'elle convient très bien.

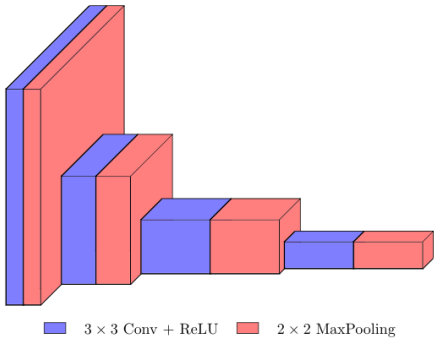


FIGURE 3 – Modèle du réseau de neuronne convolutif vu lors du TP3, Axel Carlier

3.2 Augmentation de la base de donnée

Comme nous l'avions vu en TP le surapprentissage est un problème récurrent de l'apprentissage profond, et que nous aurons donc nécessairement besoin de réaliser une augmentation de donnée sur nos images pour palier à ce défaut. Mais avant d'entreprendre cette augmentation de donnée nous nous somme demandés quelles modifications seraient pertinentes. Si par exemple on modifiait une image en la tordant et que cela générait des courbes qui ne seraient pas représentatives d'une certaine ville on risquerait de biaiser nous même nos résultats. Nous avons donc utilisé la classe *ImageDataGenerator* de Keras pour augmenter notre base de donnée en se contentant d'appliquer des rotations de toute l'image et des translations pour ne pas modifier les caractéristiques d'une image et simplement pouvoir reconnaître un même objet sur des images prises différemment.

3.3 Greffon de VGG 16

Cependant même avec ce premier réseau et l'augmentation de donnée nous ne nous attendions pas à obtenir des résultats particulièrement bon, comme nous l'avions déjà étudié en cours. Nous étions donc directement parti dans l'optique d'utiliser un autre réseau qui se reposerait sur un

VGG 16, déjà entraîné sur ImageNet, et auquel on grefferait un réseau avec peu de couches qui permettrait d'interpréter l'analyse du VGG pour la résolution de notre problème.

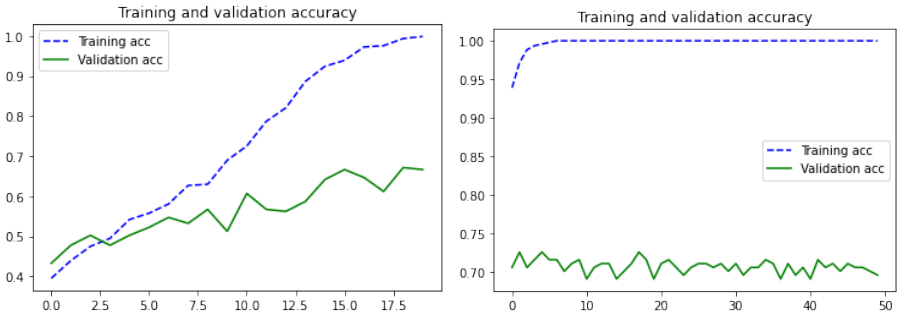
Comme indiqué précédemment, le nouveau réseau créé sera un greffon d'un réseau VGG 16 déjà entraîné sur lequel sera ajouté une couche dense de 256 neurones avec la fonction d'activation ReLU, puis une couche de 3 neurones avec la fonction d'activation softmax, qui nous permet donc de déterminer la classe de chaque image. Une telle architecture possède un nombre bien plus important de paramètres dans sa globalité , ici il y en a environ 15 millions. Cependant nous pouvons décider que la propagation des mises à jour des paramètres ne se fasse pas au porte-greffe, VGG 16, et donc se contente des deux couches décrites dernièrement.

En exploitant ce nouveau réseau nous attendons de meilleurs résultats car ce réseau possède une capacité d'apprentissage bien plus importante. En revanche il est également beaucoup plus long à apprendre et sensible au surapprentissage. Dans un premier temps nous n'avons pas propagé l'apprentissage du réseau aux paramètres du VGG 16, ce qui réduit le nombre de paramètres à entraîner à 525 315 ce qui est un nombre assez semblable au réseau précédent.

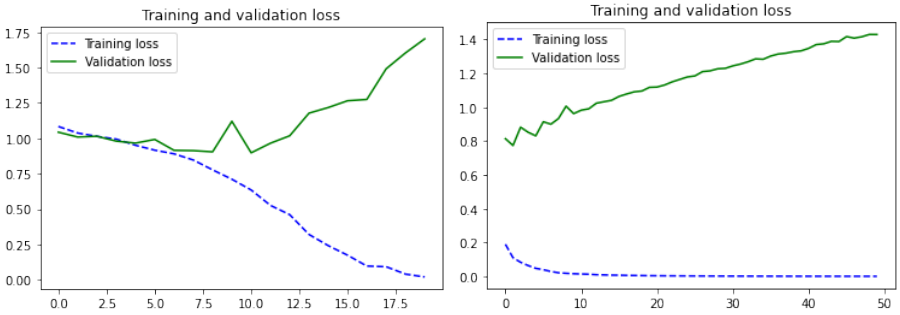
4 Analyse des résultats

4.1 Premier résultats

En suivant ce premier modèle, 3.1, nous avons donc d'abord fait apprendre notre réseau sur 20 epoch avec une taille de batch de 10, et nous avons obtenu des résultats meilleurs qu'attendus, cf fig 4. Avec seulement 3 classes, un réseau qui n'apprendrait rien aurait un taux de validation de 33%, en sachant cela nous ne nous attendions pas à ce que ce réseau relativement très simple et entraîné sur une durée inférieure à une dizaine de minutes donne un taux de réussite supérieur à 60%, nous imaginions davantage une précision de prédiction autour de 50%. Cependant même si ce premier résultat est encourageant, nous pouvons constater un problème de surapprentissage assez flagrant par cette augmentation constante de la perte sur la validation, ce constat peut déjà se faire sur seulement 20 epoch mais est sans équivoque pour 50.



(a) 20 epoch (b) 50 epoch

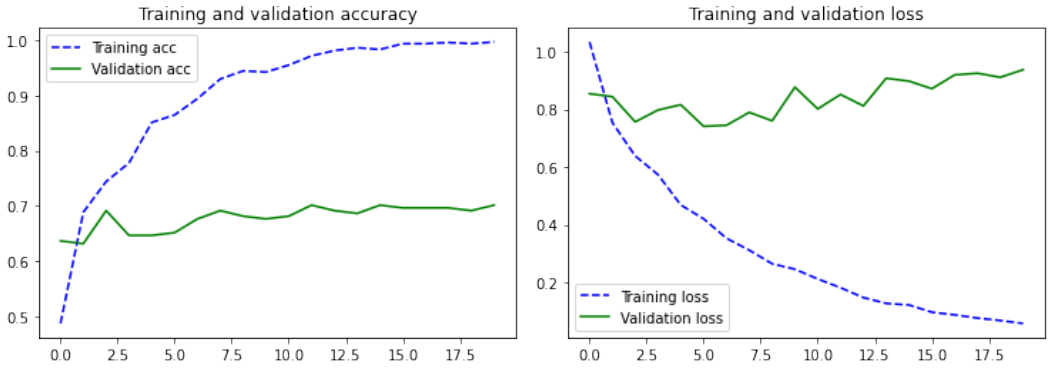


(c) 20 epoch (d) 50 epoch

FIGURE 4 – Résultat du CNN (Convolution Neural Network) selon le nombre d'epoch avec une talle de batch de 10.

4.2 Résultat du Greffon

Comme nous désirons obtenir de meilleurs résultats nous ne nous somme pas trop attardé à l'amélioration des résultats avec le premier réseau et nous avons directement entraîné le greffon avec les mêmes paramètres. Ici il n'y a donc que les deux dernières couches qui ont une mise à jour de leurs valeurs. Les résultats de cet apprentissage sans augmentation de la base de donnée, cf fig 5, correspond plus ou moins à ce qui était attendu. On observe toujours un certain surapprentissage par la même évolution de la perte de la validation, mais on remarque que la précision des prédictions s'est tout de même légèrement améliorée.



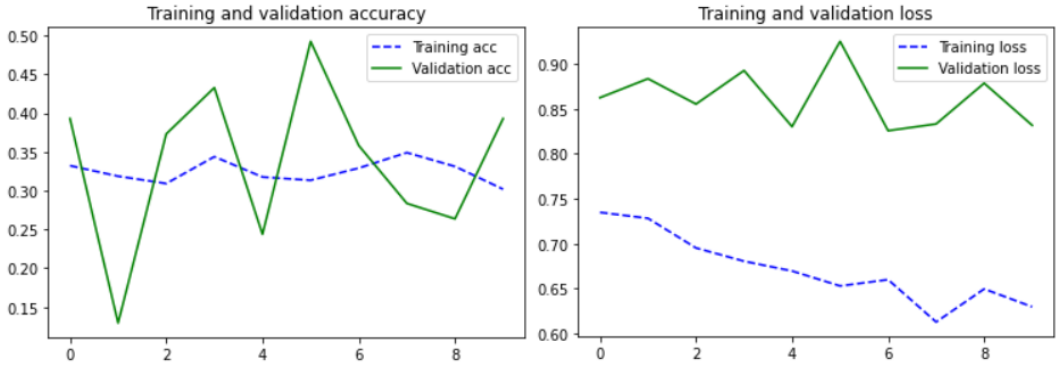
(a) Pourcentage de bonnes réponses (b) Pertes de l'entraînement et de la validation

FIGURE 5 – Apprentissage du greffon de VGG 16 sans augmentation de la base de donnée.

4.3 Premier test avec VGG 16 et la base de donnée augmentée

Pour un premier apprentissage sur la base de donnée augmentée par les images modifiées, nous avons utilisé comme paramètres 10 epochs et une taille de batch de 10 également, et le résultat, cf figure 6, nous a été un peu incompréhensible à première vue. Il n'y avait assurément pas de problème de surapprentissage, les résultats de validation oscillaient autour de 33% ce qui représente une absence d'apprentissage.

Nous avons par la suite compris que cette courbe chaotique représentait en effet une absence d'apprentissage, et la raison était un manque d'epoch. Cette valeur basse d'epochs avait pourtant donné un résultat compréhensible avec CNN et sans l'augmentation de donnée, mais l'augmentation de donnée a grandement augmenté la quantité d'information à apprendre.



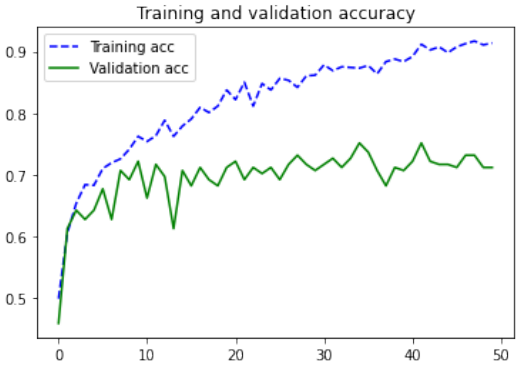
(a) Pourcentage de bonnes réponses (b) Pertes de l'entraînement et de la validation

FIGURE 6 – Apprentissage du greffon de VGG 16 avec augmentation de la base de donnée sur 10 epoch.

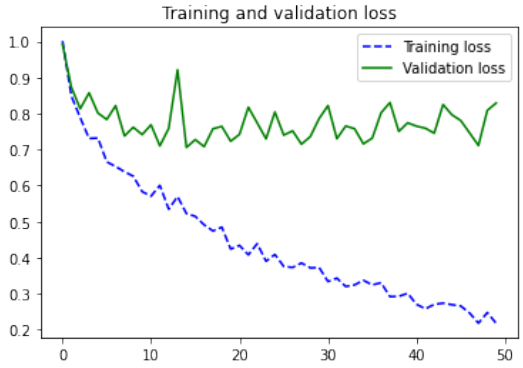
4.4 Résultats VGG 16

La solution pour obtenir un meilleur résultat est donc tout simplement de laisser le réseau apprendre sur un plus grand nombre d'epoch. Nous avons donc fait apprendre ce réseau sur 50 epoch et obtenu les courbes fig 7. La première chose à constater est que l'apprentissage prend plus de temps, je précise que l'on avait d'abord fait apprendre le VGG 16 sur 20 epoch mais on voit clairement dans les courbes obtenues que l'apprentissage n'est pas terminé à 20 epoch, alors qu'il commence à stagner à 50. Cependant continuer l'apprentissage outre mesure risquerait de détériorer la précision, on constate déjà une augmentation légère de la perte sur la base de donnée de validation.

Le résultat que l'on obtient est une précision allant de 71% pour New York à 79% pour Tokyo et Paris entre les deux avec 75%. La matrice de confusion, fig 7, est relativement équilibrée, les faux négatifs étant tous semblable le taux d'échec est similaire, ce qui est plus intéressant en revanche est le nombre plus élevé de faux négatifs pour New York. Notre interprétation est que New York se trouve à peu près à mi chemin entre Paris et Tokyo en terme d'esthétisme, parfois très colorée, parfois plus austère, et donc elle doit être plus souvent reconnue. On constate tout de même une plus grande difficulté à reconnaître Paris, car même si les échecs pour sa reconnaissance sont semblables aux autres, son nombre de vrai positif est plus faible.



(a) Pourcentage de bonnes réponses



(b) Pertes de l'entraînement et de la validation

```
Matrice de confusion
NY :   TP: 56  FN: 17  FP: 23
Paris :   TP: 42  FN: 19  FP: 14
Tokyo :   TP: 52  FN: 15  FP: 14
La précision globale est de 0.7%
```

(c) Matrice de confusion

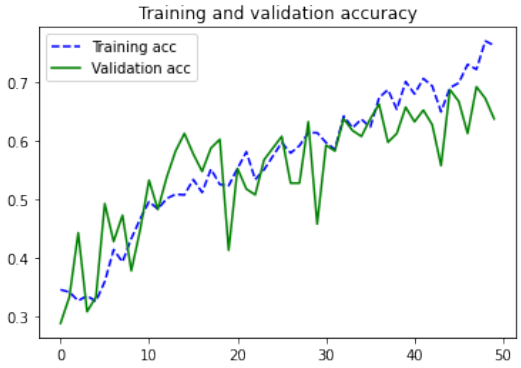
FIGURE 7 – Apprentissage du greffon de VGG 16 avec augmentation de la base de donnée sur 50 epoch.

4.5 Résultats VGG 16 Fine Tuning

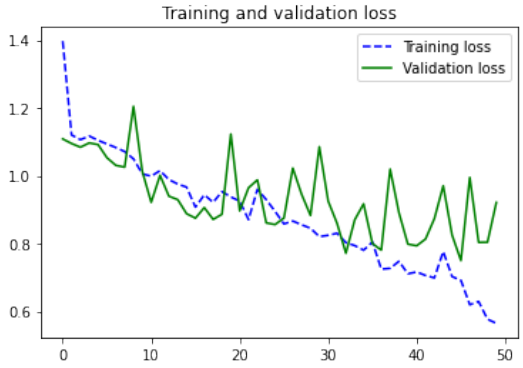
Même si nous ne constatons pas de surapprentissage nous ne sommes pas vraiment satisfait par ce taux de reconnaissance, finalement nous avons ajouté presque 15 millions de paramètres pour gagner environ 10% en taux de reconnaissance. Il nous reste donc une dernière modification faisable pour améliorer nos résultat et c'est de propager les mises à jour aux paramètres du VGG 16 qui étaient jusqu'alors inchangées, ce qui est appelé le Fine Tuning.

Les résultats, fig 8, indiquent tout d'abord que 50 epoch n'est pas suffisant pour entraîner un réseau avec 15 millions de paramètres. Nous avons plus tard essayé d'augmenter le nombre d'epoch mais nous avons été confronté à des échecs quelque peu incompréhensible comme par exemple un arrêt total de l'évolution du réseau. Donc nous pensons qu'il est possible qu'avec la propagation des mises à jour des paramètres à l'ensemble du réseau les résultats s'améliorer, même si dans l'état il n'est pas très intéressant de commenter les résultats d'un réseau qui n'a pas fini d'apprendre. Même si nous pouvons tout de même dire qu'une précision supérieure à 80% avec ce réseau semble irréalisable.

Étrangement ce réseau semble donné un grand favoritisme à Tokyo en le prédisant beaucoup plus que les autres réseau, et autre fait étonnant, il semble de plus souvent reconnaître New York quand il s'agit en réalité de Paris et Tokyo que quand il s'agit de la ville elle-même. Parmis ceux de ces trois villes, les résultats de Paris ressemblent le plus à ceux précédents, et à un apprentissage correct du réseau.



(a) Pourcentage de bonnes réponses



(b) Pertes de l'entraînement et de la validation

```
Matrice de confusion
NY :    TP: 30  FN: 37  FP: 16
Paris :    TP: 40  FN: 21  FP: 11
Tokyo :    TP: 65  FN: 8  FP: 39
La précision globale est de 0.7%
```

(c) Matrice de confusion

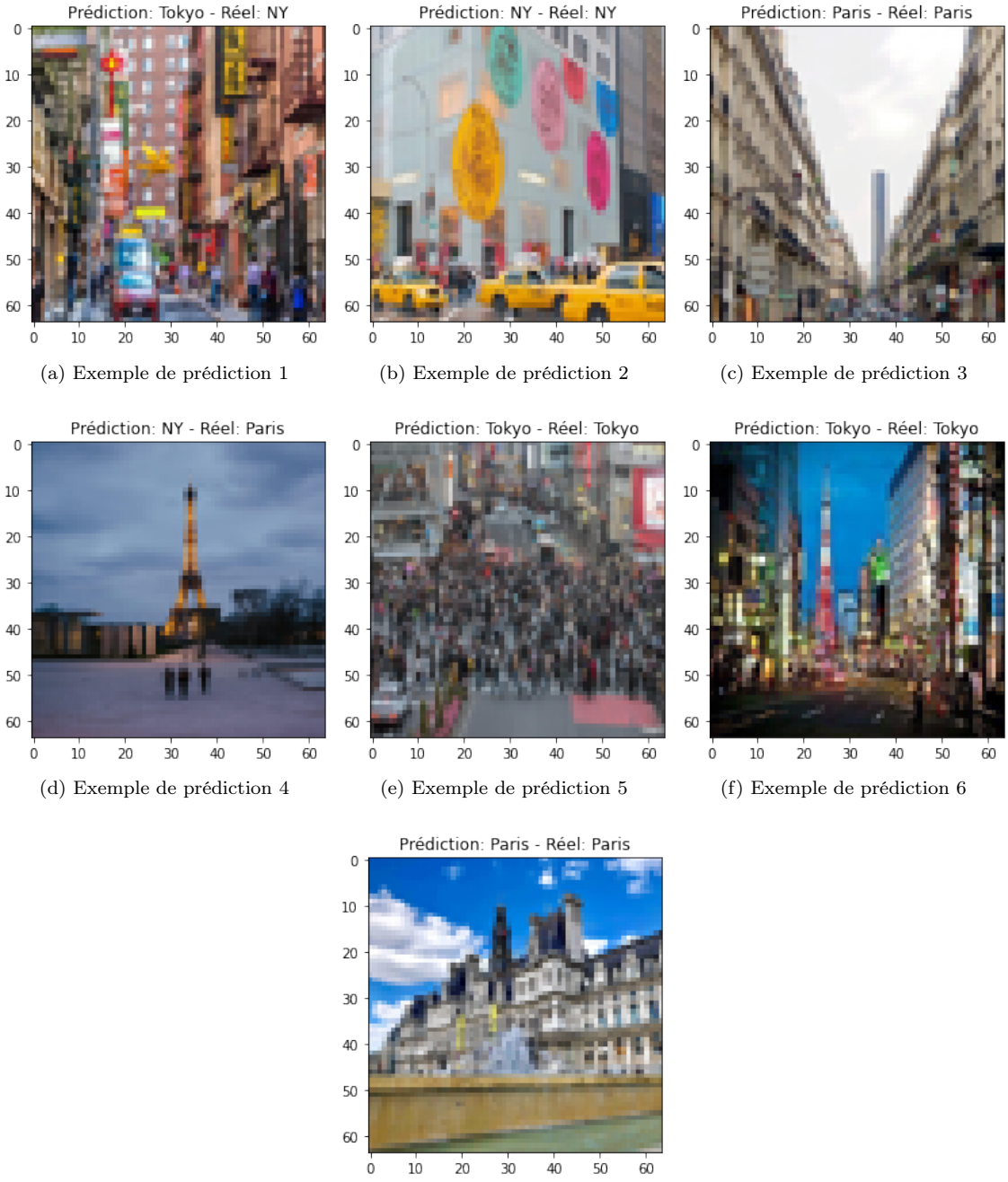
FIGURE 8 – Apprentissage du greffon de VGG 16 avec augmentation de la base de donnée sur 50 epoch.

4.6 Analyse des images

Les figures 9 et 10 nous permettent de regarder plus en détail les prédictions de notre réseau, et éventuellement de comprendre les erreurs et ce qui a marché pour améliorer encore nos résultats. Même si l'apprentissage du réseau avec le Fine Tuning est inachevé, les résultats des prédictions de l'échantillon pris sont assez souvent juste, même si il n'y a pas d'image étiquetée comme étant New York

Nous pensons que la première analyse que l'on peut faire de ces prédictions est qu'une image de résolution 64x64 perd beaucoup d'informations pour la reconnaissance d'éléments particuliers, par exemple que le premier réseau échoue à reconnaître l'image (d) comme Paris est inconcevable, même si le format actuel est suffisant pour reconnaître la Tour Eiffel.

Pour ce qui est de ces images nous avons toutefois été agréablement surpris de voir que l'image (f) n'avait jamais été confondue pour Paris alors que l'on distingue bien la tour de Tokyo qui est pourtant ressemblante avec la Tour Eiffel, et également que l'exemple (b) reconnaisse New York malgré les couleurs du bâtiments qui font très tokyoïte.

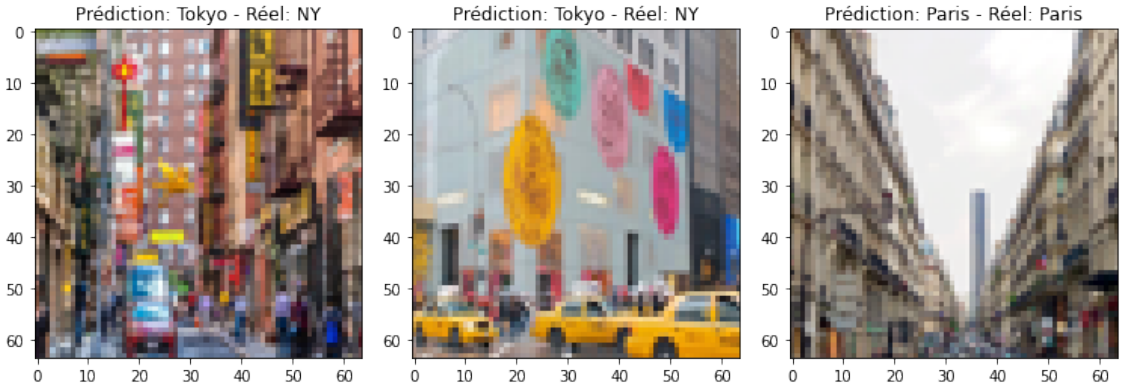


(a) Exemple de prédiction 1 (b) Exemple de prédiction 2 (c) Exemple de prédiction 3

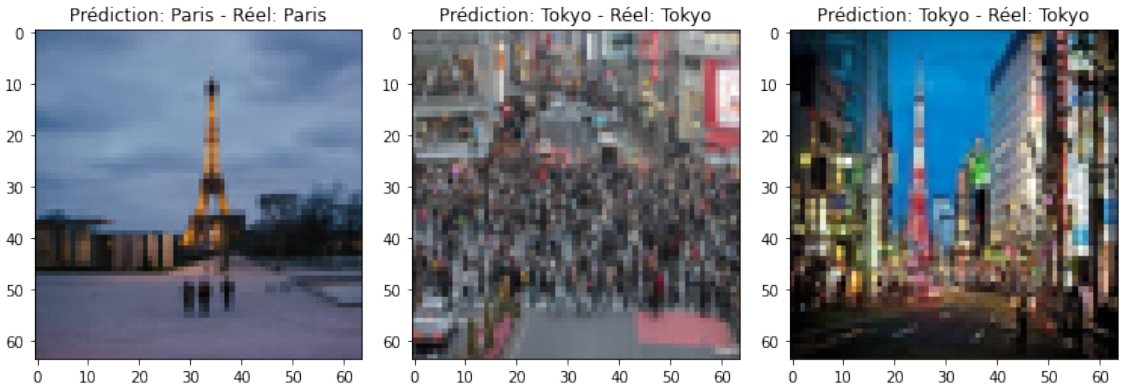
(d) Exemple de prédiction 4 (e) Exemple de prédiction 5 (f) Exemple de prédiction 6

(g) Exemple de prédiction 7

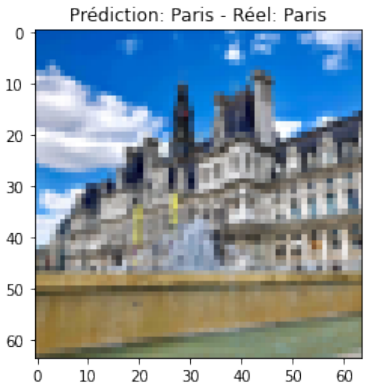
FIGURE 9 – Exemple de prédiction sans le fine Tuning.



(a) Exemple de prédiction 1 (b) Exemple de prédiction 2 (c) Exemple de prédiction 3



(d) Exemple de prédiction 4 (e) Exemple de prédiction 5 (f) Exemple de prédiction 6



(g) Exemple de prédiction 7

FIGURE 10 – Exemple de prédiction avec le fine Tuning.

5 conclusion

Ce projet a réellement pu cimenter ce que nous avions appris en TP et en cours sur l'apprentissage profond. Nous sommes globalement satisfait des résultats obtenus, en particulier pour le VGG sans Fine Tuning, même si nous sommes déçu d'avoir échoué à faire fonctionner convenablement le réseau VGG avec le Fine Tuning. Les difficultés pour implémenter ce que nous désirions au fil du projet reposaient majoritairement sur le fait que les librairies utilisées étaient très complètes et

prenaient du temps à comprendre, mais nous avons bien conscience que cela a également permis d'obtenir des résultats inaccessibles sans celles-ci. Heureusement les TP nous ont très bien guidé tout le long de ce projet

Pour ce qui est de l'amélioration de ce projet, la première chose que nous ferions serait d'augmenter le nombre de classe et d'ajouter par exemple d'autres capitales occidentales et asiatiques, mais aussi des capitales africaines, océaniques, du Moyen-Orient, et d'Amérique du Sud. Nous aimerions également augmenter la résolution des images pour permettre une meilleure précision. La dernière chose que nous ferions serait de constater si les erreurs de ces classes nous permettrait malgré tout de reconnaître les villes de chaque milieu culturel.