Réseau convolutionnel pour la classification d'images

Rapport des travaux pratiques

Khalfaoui Amani - - Chemli Yanis - - Trimèche Iyed
Telecom Paritech/UPMC

RÉSUMÉ

Les réseaux de neurones profonds ont montré leur efficacité dans de nombreuses tâches de vision et traitement d'images. Ils ont permis de surpasser les résultats de nombreuses techniques, notamment pour la classification d'images. Mais, pour être efficaces, il leur faut une importante base d'apprentissage et un paramétrage efficace. Nous introduisons dans cet écrit les réseaux de neurones avec le perceptron en particulier, sur lequel nous avons expérimenté. Nous développerons ensuite les aspects de Deep Learning et de réseau convolutionnel. Sera étudier le réseau VGG, sur lequel nous avons fait des tests, d'une part en classification et en génération de caractéristiques. Enfin nous présenterons notre propre réseau convolutionnel. Dans cette partie nous discuterons sur les performances et sur les possibles méthodes de régularisation type augmentation et normalisation de données ou DropOut.

Les réseaux de neurones

Les réseaux de neurones artificiels constituent une classe d'outils de modélisation inspirés du fonctionnement du cerveau. Introduits dans [1], le principe est d'interconnecter de nombreuses unités de calculs simples, appelées neurones artificiels, afin de permettre la résolution de problèmes difficiles, non linéaires et multi-classes.

Les principaux avantages des réseaux de neurones artificiels sont :

- Leur aptitude à apprendre de façon adaptative, en utilisant des données d'entrainement pour résoudre des problèmes complexes,
- Leur capacité « d'auto-organisation » ; Ils peuvent mettre à jour leur organisation en fonction de l'information reçue lors de l'apprentissage,
- Leur capacité de performance en temps réel (de par leur nature ces architectures sont facilement parallélisables).

Le perceptron est un réseau de neurone largement utilisé pour la classification (ou la régression). C'est la structure la plus simple d'un réseau de neurone. Elle est composée d'une couche d'entrée, d'une couche cachée et d'une couche de sortie. [2]

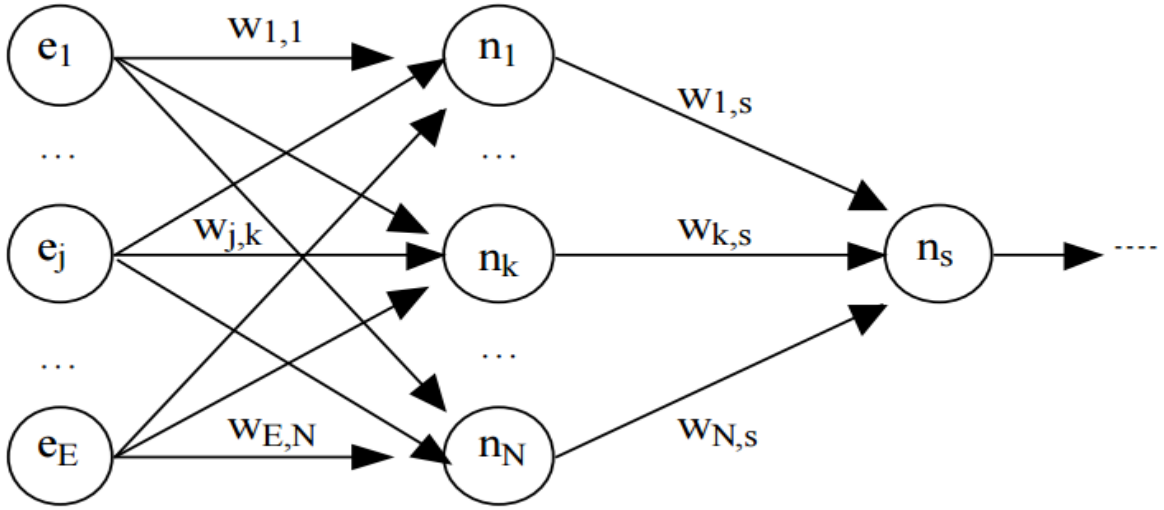


Figure 1 Perceptron à une couche cachée [2]

Chaque sortie est alors pondérée par un poids w. Usuellement un biais intervient et l'on propage les informations de l'entrée vers la sortie par une fonction d'activation (souvent non linéaire, type sigmoïde, ReLu ou tanh).

Pour un problème de classification, nous disposons de N exemples d'apprentissage, chacun étant une paire (x, y) constituée d'une donnée d'entrée x et de la classe vérité y associée. C'est à partir de ces données que le réseau de neurone peut être appris (c'est-à-dire que ses paramètres (poids et biais) sont ajustés aux données d'entrainement dans l'espoir d'une capacité de généralisation). La méthode de rétro propagation du gradient (ou back propagation) permet de réaliser cette opération.

Pour effectuer l'apprentissage, on présente au réseau, des exemples d'une base d'entrainement. A l'issue de cet apprentissage, les poids et les biais du réseau de

neurone sont fixés, et une nouvelle donnée de test peut être présentée au réseau pour être classée.

Le perceptron a une couche cachée

Nous avons mis en œuvre lors du TP 1 un perceptron, à trois couches, composé :

1. D'une couche d'entrée x (de taille nx) qui lit les signaux en entrée (taille des variables d'entrée)

2. D'une couche de sortie $\hat{y}$ (de taille ny) qui fournit la réponse du réseau.

   - Wy : la matrice de poids (nh*ny),
   - by : le vecteur de biais (ny),
   - une fonction d'activation non linéaire Softmax : $\hat{y}$ = Softmax (Wy * h + by).

   Cette fonction est utilisée essentiellement pour des tâches de classification et puisque notre réseau estime des sorties probabilistes et la somme doit être égale à 1, cette fonction permet de construire des réseaux avec des sorties normalisées.

3. D'une couche cachée h (de taille nh) connectée en entrée à chacun des neurones de la couche précédente x, et en sortie à chaque neurone de la couche suivante y.

   - Wh : la matrice de poids (nh*nx),
   - bh : le vecteur de biais (nh),
   - une fonction d'activation non linéaire tanh : h= tanh (Wh * x + bh).

Nous avons testé notre réseau sur un dataset jouet, composé d'une base d'apprentissage (donnée écrite dans la matrice Xtrain avec Ytrain les labels associés) et d'une base de test (Xtest, Ytest).

Une descente de gradient classique a été utilisée pour la rétro-propagation. Une descente de gradient stochastique n'a pas été nécessaire dans la mesure ou le nombre de données d'entrainement est très réduit (200 vecteurs). On a donc calculé la dérivée de la fonction coût sur tout l'ensemble d'apprentissage (train). On présente ci-dessous les données synthétisées avec leur vérité terrain :

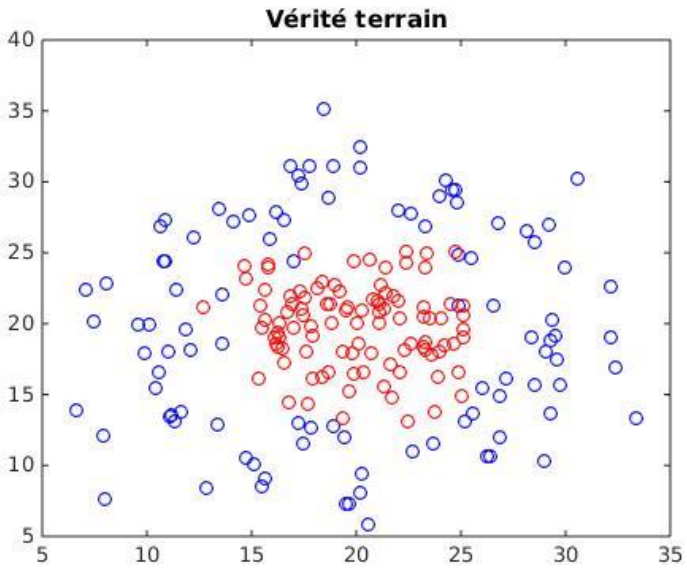
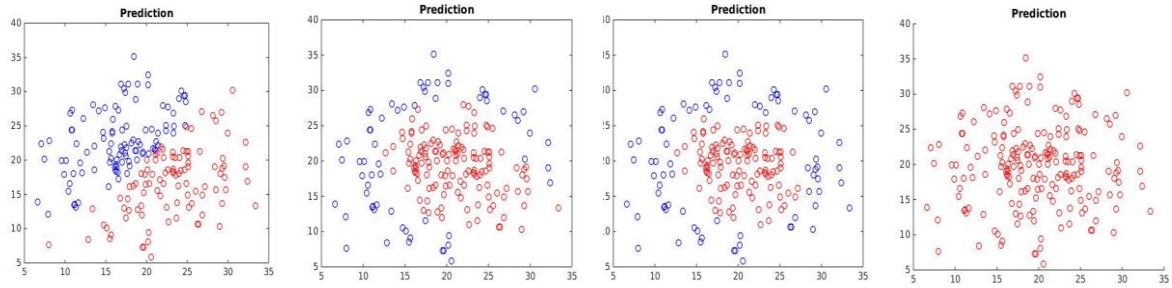


Figure 2 Données d'entrainements et vérité terrain

En prenant des tailles nh différentes de la couche cachée, (le nombre de neurone de la couche cachée). Voici différents résultats pour nh = 6, 50, 200, 250 :



Pour un nombre réduit de neurones, le perceptron ne peut faire qu'une séparation linéaire des données. On observe aussi qu'à partir d'un nombre de l'ordre de la centaine, le réseau est bien appris. Cependant à partir d'un certain nombre (240 neurones), on fait du sur apprentissage. On eut s'attendre à ce résultat car le nombre de paramètre du réseau devient conséquent fasse au nombre des données d'apprentissages.

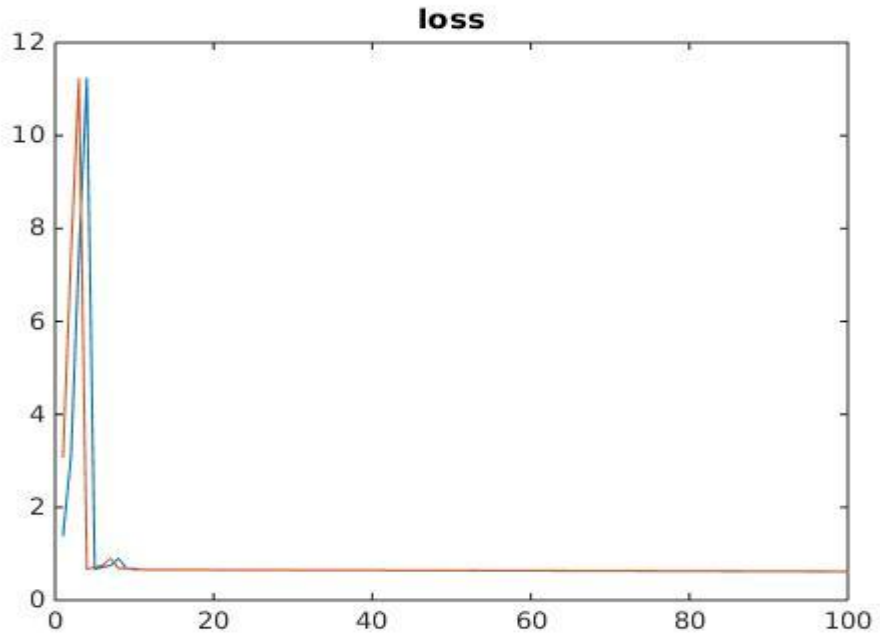


Figure 3 Evolution de la loss à chaque itération

En règle générale, le nombre optimal d'unités cachées varie et il n'a y pas de règles pour le fixer. On peut choisir la taille moyenne des couches d'entrée et de sortie et on peut essayer plusieurs tailles jusqu'à trouver le meilleur résultat. Si on a beaucoup d'exemples de formation, on peut utiliser plusieurs unités cachées, mais parfois juste 2 unités cachées fonctionnent mieux avec peu de données.

Les réseaux de neurones à une couche cachée ont fait leurs preuves et ont été largement utilisés ces dernières années. Cependant ils demandent une base d'apprentissage conséquente pour des problèmes complexes et donc un temps de travail humain considérable.

De plus, l'algorithme de rétro-propagation souffre de deux inconvénients majeurs :

- L'expérience montre que le temps d'entraînement d'un RN croît rapidement lorsque le nombre de neurones augmente,
- Les réseaux de neurones n'échappent pas au problème central du Machine Learning : le sur apprentissage (overfitting).

Le challenge est donc de réduire au maximum le nombre d'exemples d'apprentissage.

Dans la suite nous allons plus loin dans les méthodes à base d'apprentissage. Nous présentons des méthodes basées sur les réseaux de neurones artificiels appelé réseaux de neurones convolutionnels (CNN). Nous introduisons dans la suite le sujet du Deep Learning et abordons ensuite le sujet des CNN. Nous verrons en quoi l'utilisation de réseaux profonds est un avantage face aux réseaux classiques type perceptrons.

Les réseaux convolutionnels

Généralités sur l'apprentissage profond ou deep learning

Nous avons présenté précédemment les structures de réseaux de neurones pour la classification et la segmentation. On appel réseau de neurones profonds un réseau qui possède plus d'une couche cachée. Le but théorique est alors d'approximer n'importe quelle fonction.

Cette famille d'algorithmes a permis de faire des progrès importants dans les domaines de la reconnaissance d'images et du traitement des images en général, notamment en segmentation.

Les modèles de réseaux de neurones profonds sont bâtis sur le même modèle que les perceptrons précédemment décrits. Cependant, on fait intervenir des couches intermédiaires plus nombreuses.

Chacune des couches intermédiaires cachées va être subdivisée en sous partie, traitant un sous problème, plus simple et fournissant le résultat à la couche suivante et ainsi de suite.

Cette manière d'ordonner les déductions amènent à ajouter au fur et à mesure un contexte de plus en plus précis au sujet sur lequel le modèle opère. [3]

Il existe différents algorithmes de deep learning. Nous pouvons ainsi citer :

- Les réseaux de neurones convolutionnels (CNN ou Convolutional Neural Networks). Le problème est divisé en sous parties et pour chaque partie, un « cluster » de neurones sera créé afin d'étudier une portion spécifique. Par exemple, pour une image en couleur, il est possible de diviser l'image sur la largeur, la hauteur et la profondeur (les couleurs). [4]
- La machine de Boltzmann profonde (Deep Belief Network, G. Hinton) : Ces algorithmes fonctionnent suivant une première phase non supervisée, suivi de l'entrainement classique supervisé. Cette étape d'apprentissage non-supervisée, permet, en outre, de faciliter l'apprentissage supervisé. [5]

Nous détaillons dans la suite les réseaux de neurones convolutionnels (il existe cependant une différence au niveau de l'architecture par rapport au CNN classique que nous mettrons en avant).

Les réseaux convolutionnels (ou ConvNets) sont des types de réseaux de neurones artificiels dans lesquels le motif de connexion entre les neurones est inspiré par le cortex visuel des animaux (du chat particulièrement). Les neurones de cette région du cerveau sont arrangés de sorte à ce qu'ils correspondent à des régions qui se chevauchent lors du pavage du champ visuel. Leur fonctionnement est inspiré par les processus biologiques, ils consistent en un empilage multicouche de perceptrons, dont le but est de prétraiter de petites quantités d'informations. [6]

Un des intérêts particuliers des réseaux convolutifs est leur capacité et apprendre les représentations elles-mêmes à partir des données.

Dans les architectures classiques de réseaux de neurones convolutionnels, les couches de convolution sont suivies par des couches de sous-échantillonnage. Une couche de sous-échantillonnage réduit la taille des entrés convoluées et introduit de l'invariance aux faibles rotations et translations pouvant apparaître en entrée.

Une couche de convolution est paramétrée par son nombre N de cartes de convolution, la taille des noyaux de convolution (souvent carrée), et le schéma de connexion à la couche précédente. Chaque carte de convolution est le résultat d'une somme de convolution des cartes de la couche précédente par son noyau de convolution respectif. Un biais b est ensuite ajouté et le résultat est passé à une fonction d'activation. [7]

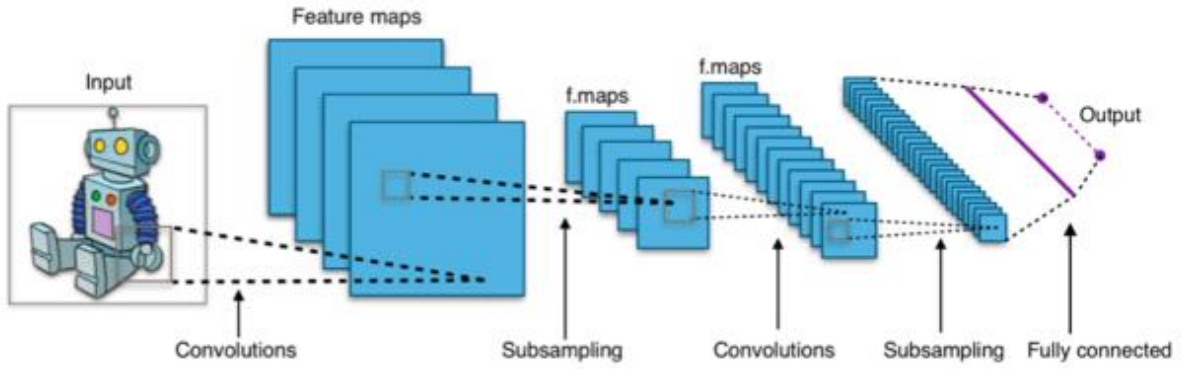


Figure 4 Les premières étapes d'un réseau convolutionnel (https://upload.wikimedia.org/wikipedia/commons/6/63/Typical_cnn.png)

Le début de l'architecture consiste donc en une succession de couches de convolution, sous échantillonnage (subsampling) et fonctions d'activation. Ces enchaînements sont dédiés à l'extraction automatique de caractéristiques, tandis que la seconde partie, composée de couches de neurones complètement connectés (fully connected), est dédiée à la classification.

Intérêt du deep learning

L'approche deep learning est pertinente dans la mesure où elle permet d'approximer des fonctions complexes avec la même précision qu'un réseau de neurone classique en utilisant plus de couche mais finalement beaucoup moins de nœuds neuronaux. On fait donc intervenir moins de paramètres donc moins de degrés de libertés. Outre les avantages computationnels, un model avec moins de degrés de libertés nécessite moins de données d'apprentissage. Or on a vu que le nombre d'élément d'une base d'entrainement peut être un facteur limitant. [8]

Expérimentations

Dans la suite des TP, nous avons pris en main la bibliothèque MatConvNet et étudier le réseau VGG.

Notre réseau est représenté par un empilement de couches :

- Couche de convolution
- Couche de pooling

- Couche ReLu
- Couche de normalisation
- Couche SoftMax

Convolution

Une couche de convolution est paramétrée par son nombre N de cartes de convolution, la taille des noyaux de convolution (souvent carrée), et le schéma de connexion à la couche précédente. Chaque carte de convolution est le résultat d'une somme de convolution des cartes de la couche précédente par son noyau de convolution respectif. Les convolutions du réseau en U sont au nombre de 64 par couche et ont une taille fixe de 3x3.

Max-Pooling

On a vu précédemment que dans les architectures classiques, les couches convolés sont suivit par une étape de sous échantillonnage. Dans la méthode en U, une autre approche est proposée. C'est une technique existante qui a fait ses preuves [9]. Une couche de sous échantillonnage est alors remplacé par la couche de max–pooling. La sortie d'une couche de max–pooling est donnée par la valeur maximale d'activation au sein de la couche d'entrée pour différentes régions.

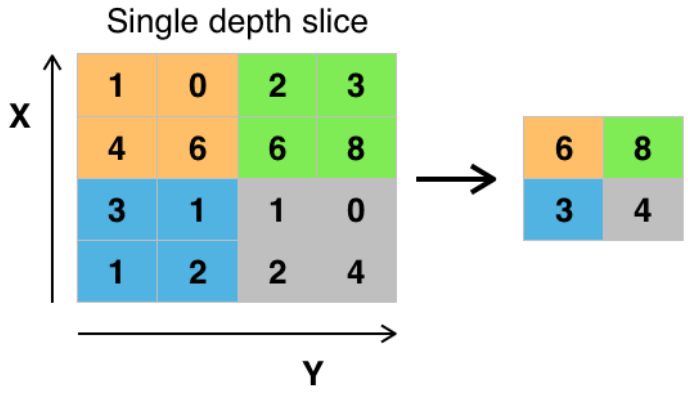


Figure 5 https://upload.wikimedia.org/wikipedia/commons/e/e9/Max_pooling.png

ReLu

La fonction ReLu est la fonction d'activation utilisée après chaque convolution. Elle est définie par :

$$f(x) = \max(0, x)$$

Avec x, l'entrée d'un neurone. Elle a pour représentation graphique :

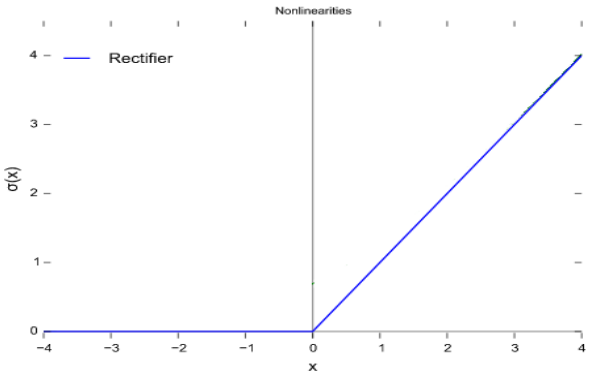


Figure 6
https://upload.wikimedia.org/wikipedia/en/6/6c/Rectifier_and_softplus_functions.svg

Cette fonction, étant aussi appelée "fonction d'activation non saturante", augmente les propriétés non linéaires de la fonction de décision et de l'ensemble du réseau sans affecter les champs récepteurs de la couche de convolution. Elle sature les valeurs négatives à zéro et introduit de la non-linéarité.

Pour chaque image de la base, nous calculons la sortie de chaque couche du réseau à partir des couches précédentes grâce à la fonction **vl_simplenn** de MatConvNet permettant de calculer la prédiction du modèle.

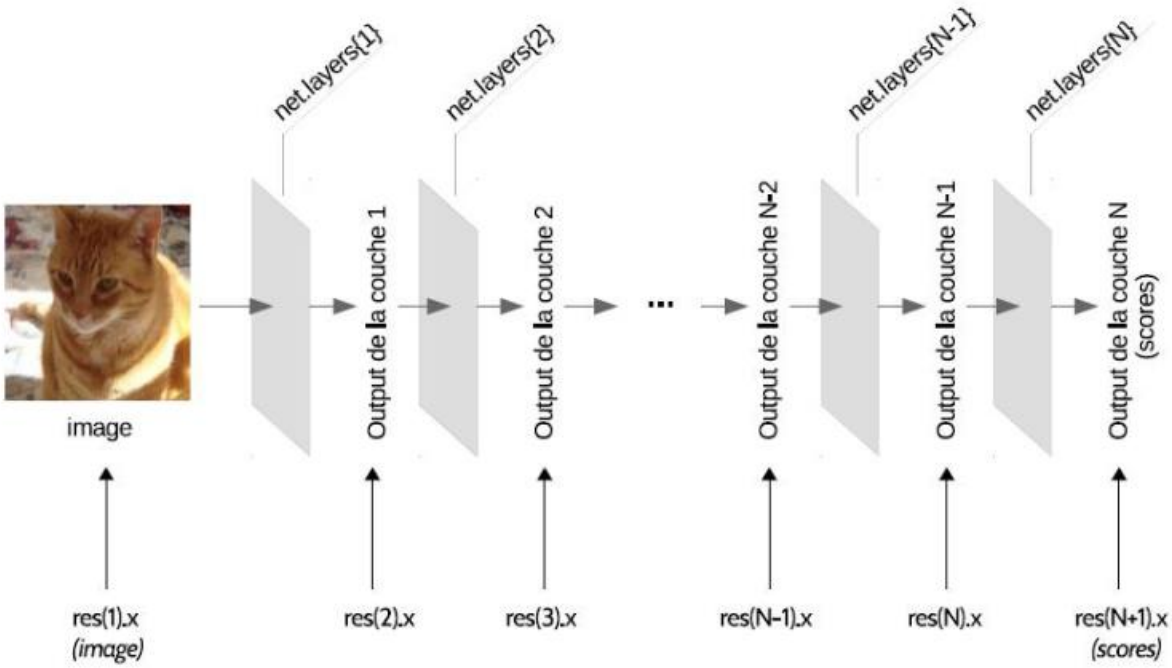


Figure 7 Accès aux différentes couches

Nous étudions le réseau pré-entraîné CNN-F (Fast Convolutionnel Neural Network) de Chatfield et al [9]. Il comporte 22 couches d'enchainement de convolution/ReLU/norm2/MaxPool, une couche « fully conected » et la fonction de cout optimisé est la fonction SoftMax (par une descente de gradient stochastique ici). Le réseau prend en entrée une image en couleur (RGB) de taille 224*224. Un traitement rapide est assuré par un stride de 4 pixels dans la première couche convolutionnelle. Ce réseau a été pré entrainé sur la base d'image *ImageNet* contenant actuellement dix millions d'images annotées pour 1000 classes.

L'architecture du réseau est simplifiée dans la figure suivante **:**

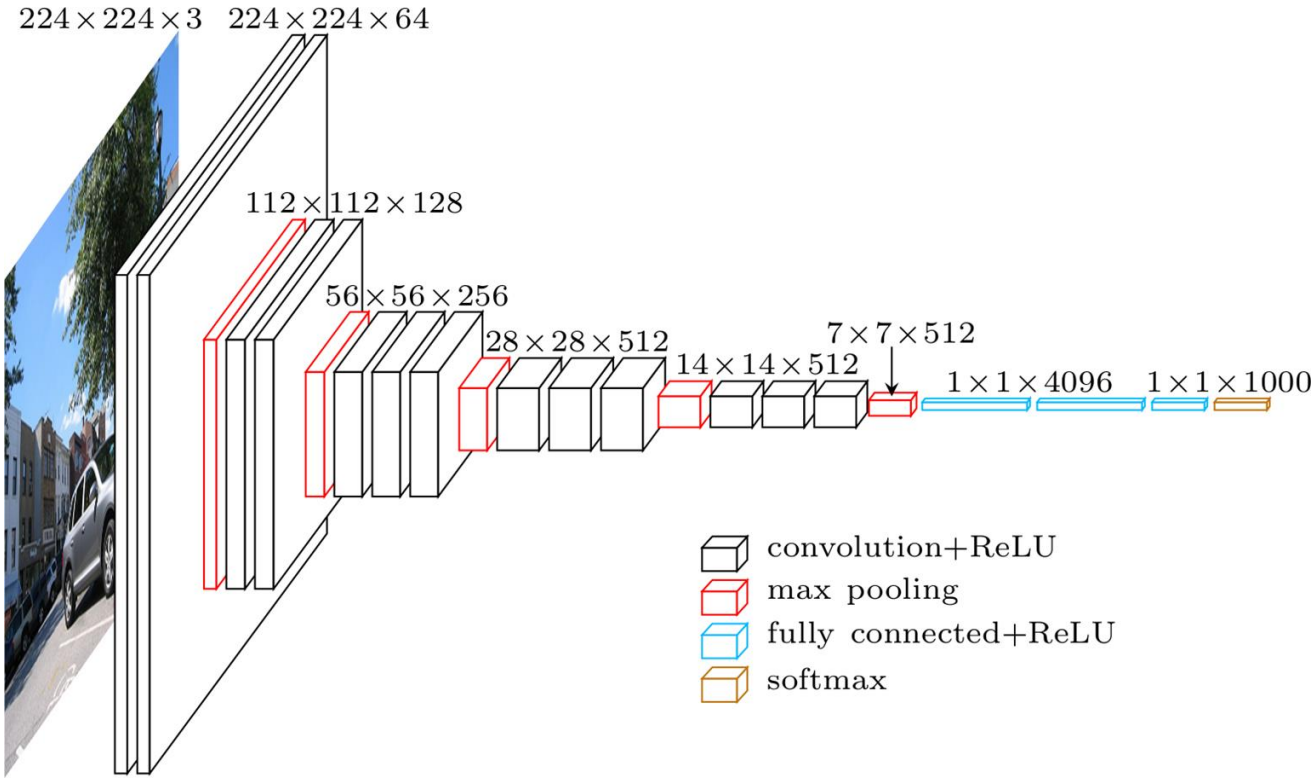


Figure 8 Description simplifiée de l'architecture

En appliquant la fonction vl_simplenn : l'algorithme nous affiche le score et la classe prédite parmi les 1000 classes disponibles et le résultat res est une structure 1*22 : chaque élément i+1 contient la sortie de la couche i de réseau **net.**

Prenons l'exemple de l'élément res (2) =54*54*64 avec un stride de 4 : 224/4 = 56 avec une perte de 2 pixels au bord de chaque côté.

La relation liant les tailles des entrées et des sorties des couches de convolutions s'écrit sous la forme suivante :

$$Ts = Te/s - 2$$

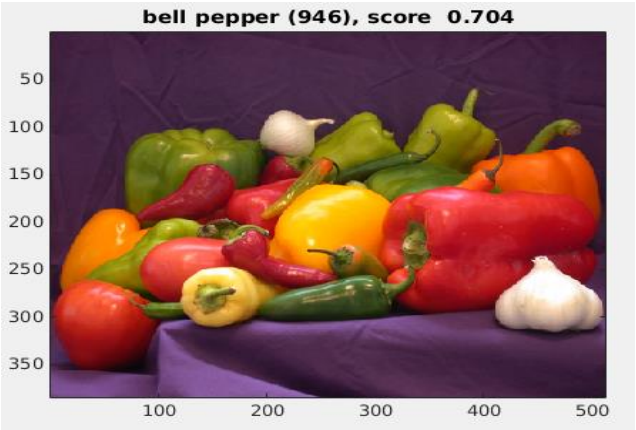
Avec :

Ts : taille de la sortie

Te : taille de l'entrée

s : le pas de convolution (stride).

On a ainsi pu tester le réseau sur l'image « pepper »



Parmi les 1000 classes, le VGG a pour sortie le score de 0.704 pour la classe « bell pepper ». On peut observer l'ensemble des probabilités calculée pour les 1000 classes :

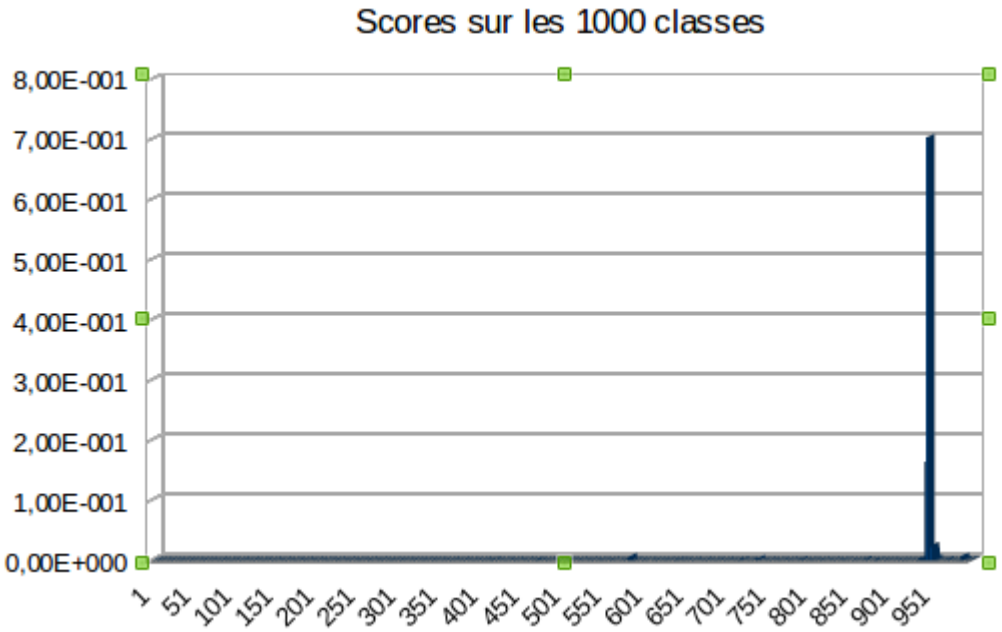


Figure 9 Les différent scores calculé (le max correspond à la classe correspondante)

Nous avons ensuite affiché la sortie de la couche correspondante à la première convolution. Ceci permet de prendre compte de l'effet des filtres de convolution qui ont été appris lors de l'entrainement.

La figure ci-dessous illustre les images correspondant aux différentes cartes de convolutions. On observe alors que les filtres cherchent à mettre en évidence les changements d'intensité et le passage d'une couleur à une autre.

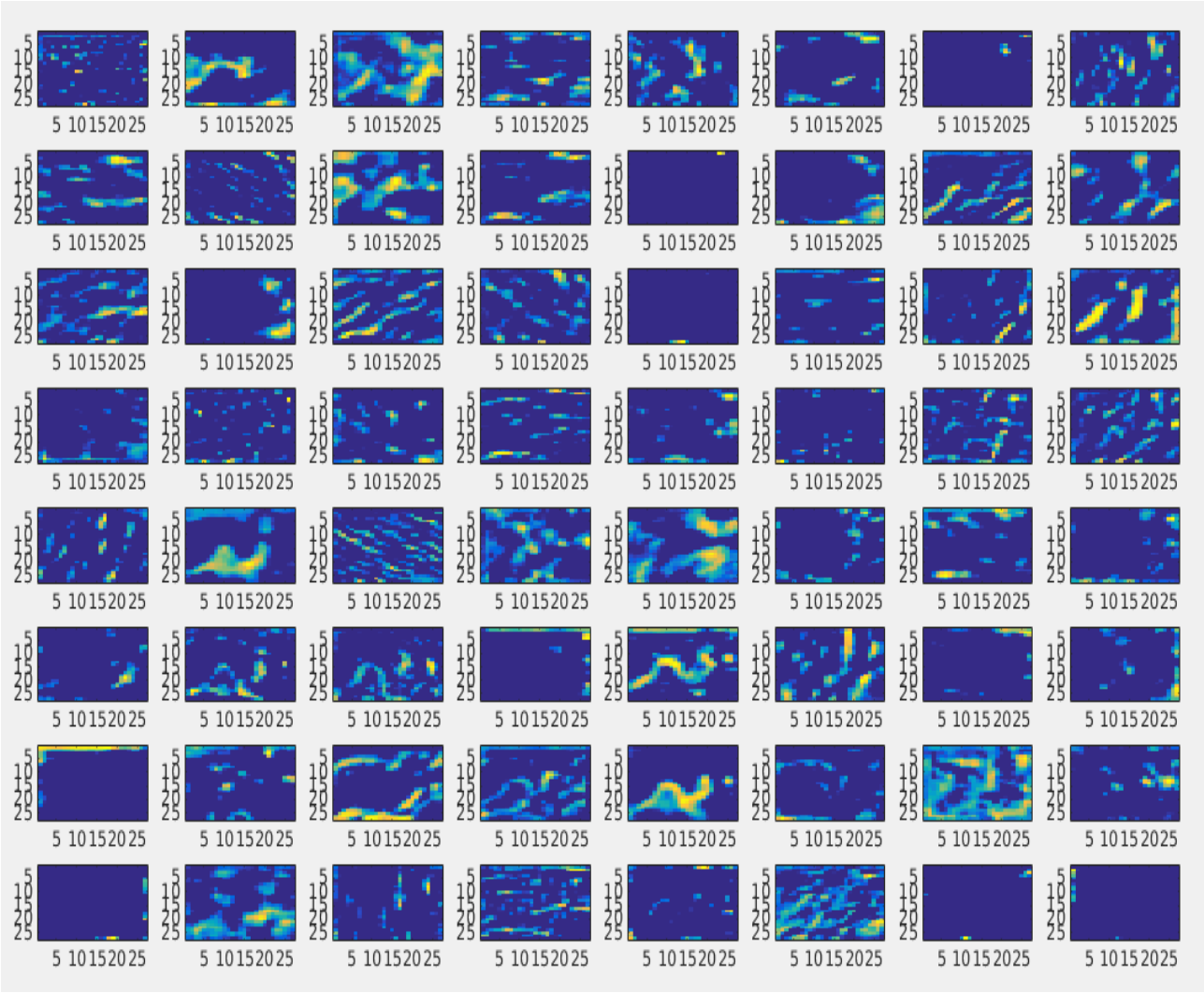


Figure 10 Cartes en sortie de la première couche de convolution

Ces cartes de convolutions seront concaténées en un vecteur de caractéristiques.

La raison pour laquelle un réseau moins large en termes de nombre de neurones par couche, mais plus profond est plus efficace qu'un réseau moins profond à taille égale (en nombre de neurones) est qu'un réseau profond réduit la quantité de travail redondant effectué.

Pour une image contenant par exemple des formes géométriques dont nous aimerions détecter et identifier le type, les premières couches d'un réseau profond effectuerons des tâches de bas niveaux tel que l'extraction de caractéristiques tel que les gradients, les formes élémentaires etc. pour la transformé en une image plus « concrète » et exploitable au regard du réseau et de la prise de décision. On fait donc un apprentissage des caractéristiques à différents niveaux d'abstractions. Les couches supérieures peuvent alors effectuer la détection et la classification à partir de la représentation plus simple de l'image.

Pour un réseau classique, type perceptron, les tâches de bas niveau devront être effectuées à plusieurs reprises, car il y a peu de propagation de résultats intermédiaires. Cela entraîne de la redondance. [10]

Utilisation du VGG pour l'extraction de caractéristiques

On a pu constater au TP suivant que les caractéristiques apprises par le réseau peuvent être utilisé pour le SVM (support Vector Machine).

Le principe de la démarche se compose de deux parties : on a utilisé un réseau de neurones pré-appris pour faire de la feature extraction à la suite de laquelle on entraine un modèle de classification. La première partie peut se voir comme une alternative à l'approche Bag of Words (BoW). L'objectif a donc été d'obtenir pour chaque image une représentation vectorielle de l'image décrivant son contenu, que l'on a par la suite utiliser pour la classification d'image (base *15-scenes*).

Nous avons observé que les performances de classification ne sont pas les même selon quelle sortie de couche de convolution était considérée. Pour les premières couches, les caractéristiques ou carte de convolutions, sont très localisées, symétriquement, en fin du réseau les caractéristiques sont trop abstraites ou trop « spécialisées » pour la base d'apprentissage. La couche 19 du VGG nous permet de classer au mieux les images de tests. Il a été intéressant d'observer que les caractéristiques sont souvent corrélées à cause du chevauchement lors des convolutions (due au stride). On peut décolérer les données et ne garder que les plus significative pour faire notre classification en utilisant un algorithme de réduction dimensionnelle type PCA (Principal Composant Analysis). On gagne alors énormément en temps d'exécution tout en gardant une accuracy satisfaisante.

Nous nous sommes concentré ici sur le réseau CNN-F, il existe cependant d'autres types dont voici les caractéristiques :

Arch.	conv1	conv2	conv3	conv4	conv5	full6	full7	full8
CNN-F	64x11x11 st. 4, pad 0 LRN, x2 pool	256x5x5 st. 1, pad 2 LRN, x2 pool	256x3x3 st. 1, pad 1 -	256x3x3 st. 1, pad 1 -	256x3x3 st. 1, pad 1 x2 pool	4096 drop- out	4096 drop- out	1000 soft- max
CNN-M	96x7x7 st. 2, pad 0 LRN, x2 pool	256x5x5 st. 2, pad 1 LRN, x2 pool	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 x2 pool	4096 drop- out	4096 drop- out	1000 soft- max
CNN-S	96x7x7 st. 2, pad 0 LRN, x3 pool	256x5x5 st. 1, pad 1 x2 pool	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 x3 pool	4096 drop- out	4096 drop- out	1000 soft- max

Le **CNN-M** (Medium Convolutionnel Neural Network) est caractérisé par la diminution du pas de convolution (stride) et du champ réceptif plus petit de la première couche convolutive.

Le **CNN-S** (Slow Convolutionnel Neural Network) quant à lui, utilise des filtres 7*7 avec un stride de 2 dans la couche conv1.Contrairementà CNN-M le stride dans conv2 est plus petit (1 pixel) mais le max-pool est plus grand dans conv1 et conv5 (3*3) pour compenser la résolution spatiale accrue.

Notre mini réseau et ses régularisations

D'autre type de régularisation et optimisations peuvent être misent en œuvre pour les réseaux convolutionelles. Ainsi, lors du dernier TP, nous avons construit notre propre architecture. Nous l'avons testé puis nous avons mis en œuvre certaines techniques de régularisation.

Nous avons donc appris un petit réseau de convolution sur la base d'images 15-Scenes L'apprentissage et l'évaluation a été fait avec la bibliothèque MatConvNet.

Le prétraitement d'images contient le passage en simple précision et le redimensionnement. Le fait de réduire la taille des images (64*64) permet d'accélérer l'apprentissage car pour une taille donnée de filtre, on fait moins d'opération de convolution sur une image de taille réduite.

Notre réseau construit, est composé des couches suivantes :

1. conv1 : couche de convolution avec 10 filtres de taille 9 × 9, un stride de 1 et pas de padding ;
2. pool1 : couche de max pooling sur des voisinages de taille 7 × 7, un stride de 7 sans padding ;
3. relu1 : couche de fonction de transfert non-linéaire de type ReLU ;
4. fc2 : couche totalement connectée.

Avec les l'hyper-paramétrage suivant :

1. Nombre d'epochs : entre 100 et 500 (selon les tests effectués),
2. Pas d'apprentissage (learning rate) : 0.0001,
3. Nombre d'image par batch : 50.

Le learning rate est un paramètre de l'apprentissage qui contrôle la variation de la valeur des poids et biais du réseau. Il joue sur la vitesse d'apprentissage. La taille de mini batch influe sur la mémoire mise en jeux lors de l'apprentissage.
Il admet donc environ 10 000 paramètres pour 1500 images d'entrainement. Lors des tests on se rend vite compte l'effet du nombre largement supérieur des paramètres. En effet on observe sur la loss qu'il y a « overfitting » ou sur-apprentissage :

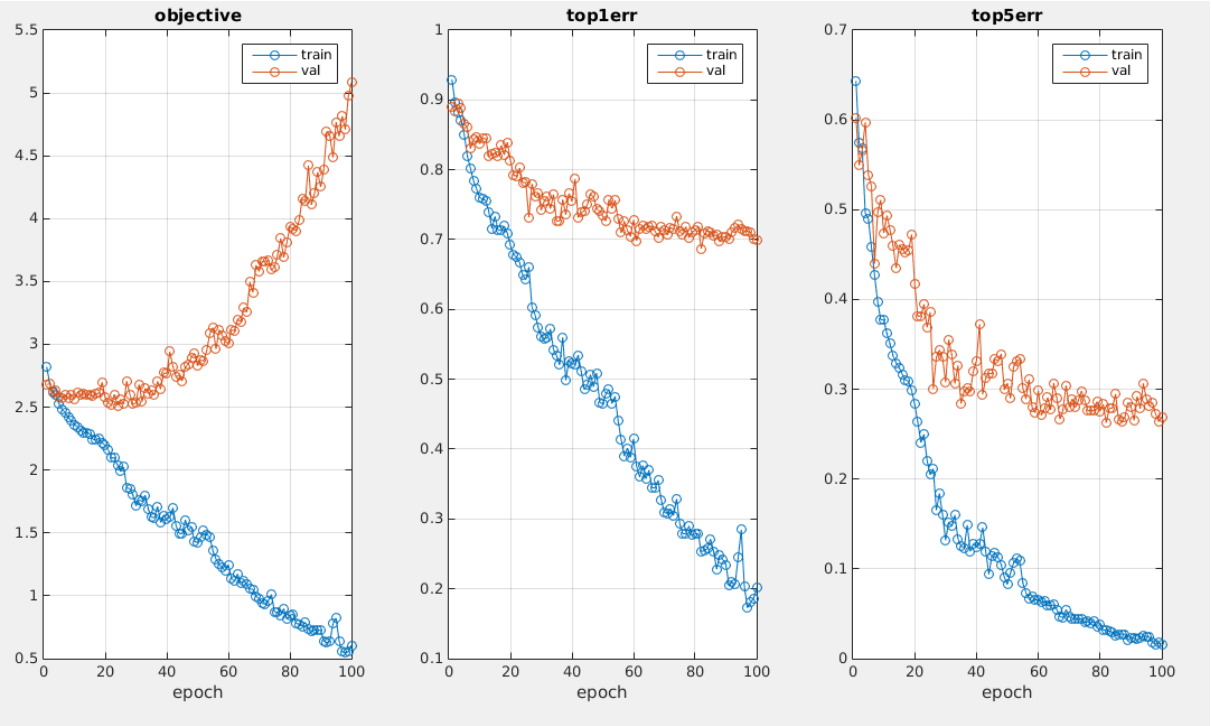


Figure 11 Aucune optimisation : overfitting, précision = 72%

Une première idée d'optimisation est **l'augmentation de données**.

Augmentation de données
La formation d'un grand réseau nécessite beaucoup d'échantillons. Pour augmenter nos données, l'idée est de procédé à des transformations sur les échantillons disponibles. On peut lister les techniques de transformations suivantes, qui sont largement utilisées :

- Les translations/rotations
- Les changements d'échelle
- Les réflexions miroirs
- Les distorsions photométriques
- Les transformations élastiques.

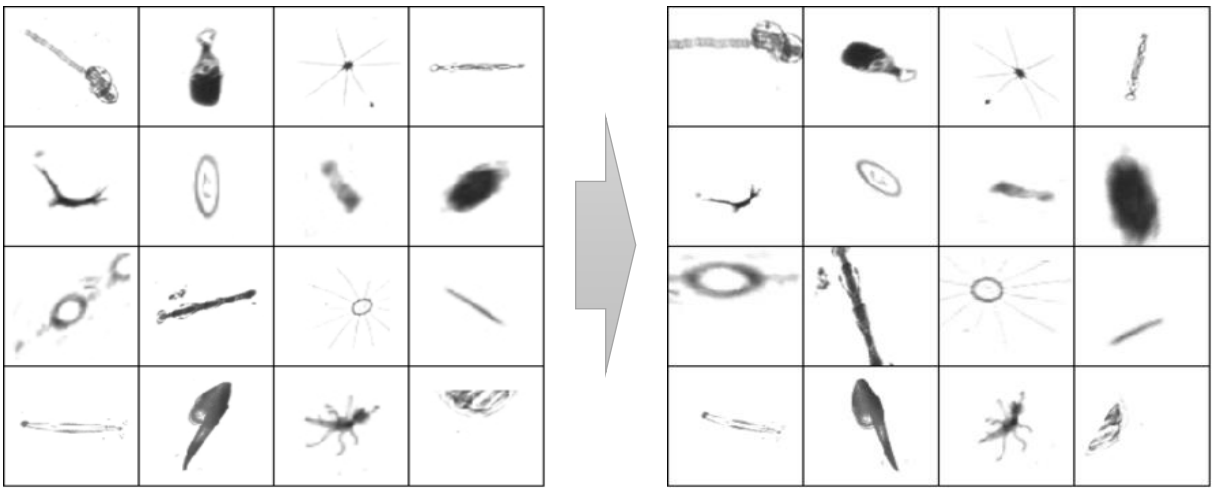


Figure 12 Exemple de transformations pour l'augmentation de données. (https://moodle.technion.ac.il/mod/resource/view.php?id=390948&lang=en)

mais il va être important de choisir des méthodes d'augmentation qui sont pertinentes par rapport au type d'images que l'on a à disposition. Dans le cadre du TP, nous traitons des images de scène naturelles type plage, chambre etc. Nous pourrons donc appliquer des transformation type symétrie horizontales qui semples pertinents. Ces transformation double le nombre des images d'apprentissage.

Ces augmentations d'images sont réalisées de façon dynamique, « à la volée », directement pendant l'apprentissage.

On observe un gain d'environ 10% en précision :

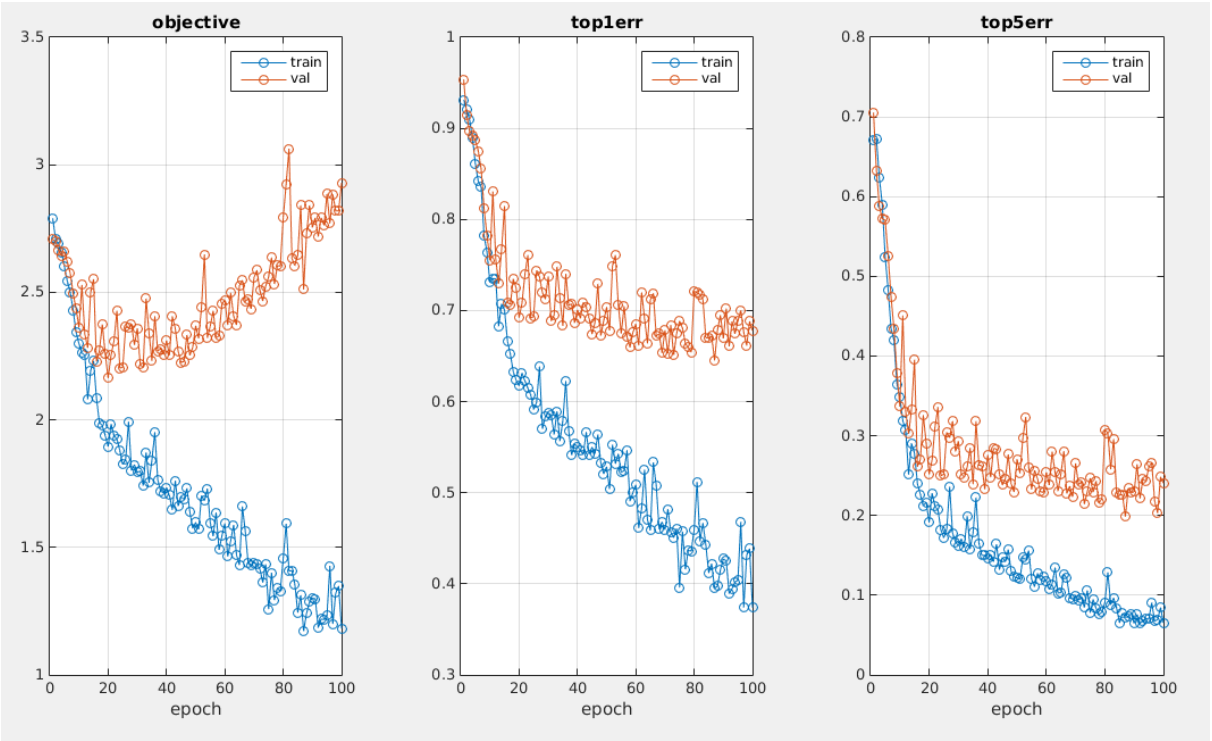


Figure 13 Régularisation par augmentation de donnée : overfitting, précision = 66%

Normalisation des exemples d'apprentissage

Une autre astuce courante en Machine Learning est de normaliser les exemples afin de mieux conditionner l'apprentissage. Pour cela, on modifie la génération des ensembles d'apprentissage et de validation en calculant l'image moyenne sur tous les exemples d'apprentissage et en la soustrayant aux images. Cette opération permet de gagner ici environs 15% de précision en validation :

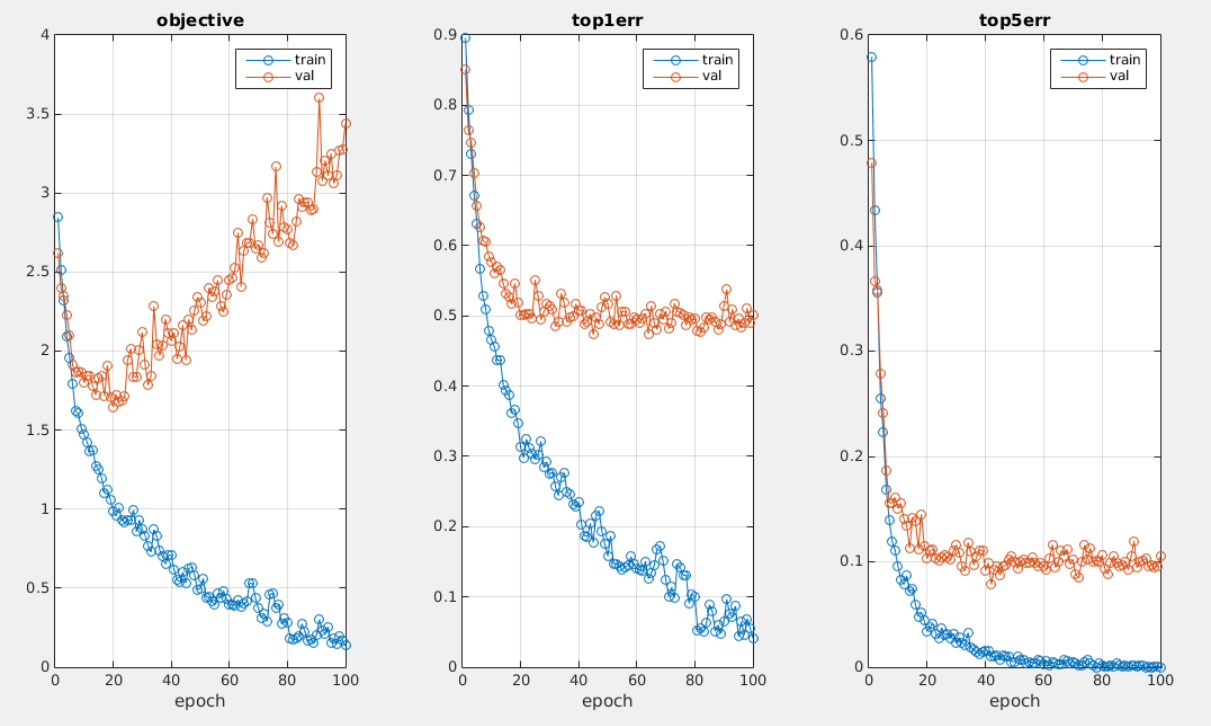


Figure 14 Régularisation par augmentation et normalisation de données : overfitting, précision = 50%

Régularisation du réseau par dropout

Une dernière technique de régularisation a été mise en place. Il s'agit du dropout. Le dropout est une technique de régularisation qui permet de réduire la sur apprentissage des réseaux neuronaux en empêchant des coadaptations complexes sur les données d'entraînement. C'est un moyen efficace permettant d'effectuer la moyenne des modèles avec des réseaux de neurones. La technique repose sur la désactivation de certains neurones. Cela force l'apprentissage du réseau sur différents "chemins".

Cette régularisation permet d'éviter le sur-apprentissage et permet un gain de temps d'exécution. Cela est vrai car on a moins de paramètres à apprendre pour un nombre constant de donnée d'apprentissage.

On désactive ici 50% des neurones et obtient les résultats suivant :

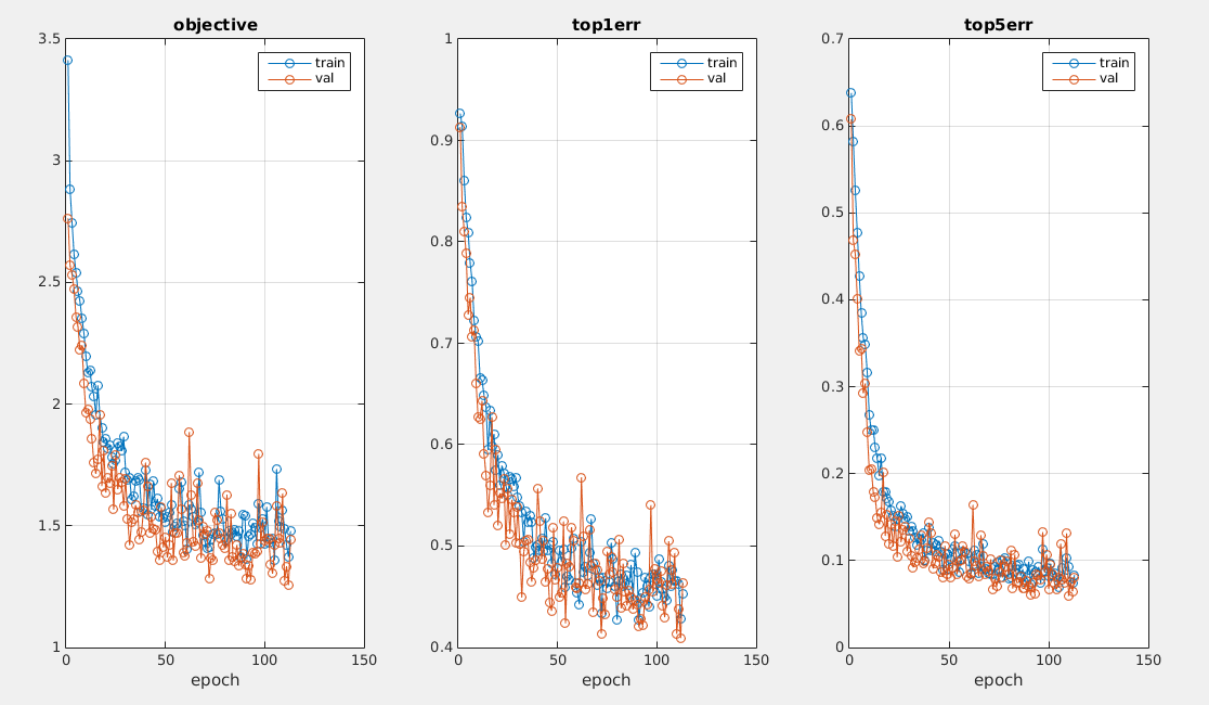


Figure 15 Régularisation par augmentation de donnée et DropOut : pas d'overfitting, précision = 42%

Cette opération permet de stabiliser l'erreur à 40%.

Toutes ces optimisations nous ont permis d'éviter l'overfitting et de passer d'environ 75% d'erreur a 40%. Il existe cependant d'autres technique d'optimisation qui n'ont pas été mise en place ici. On peut noter les techniques de DropConnect, de pooling stochastique.

Conclusion:

Dans la vie réelle, nous faisons face à des situations ou les données changent continuellement. Les systèmes d'apprentissage, eux aussi, font face au dilemme de stabilité (les informations déjà apprises ne se perdent pas à cause de nouvelles informations) et de plasticité (le système est capable de s'adapter aux changements de l'environnement et apprendre de nouvelles informations). Les réseaux de neurones sont, donc, conçus pour contourner ce dilemme. Au cours de ces Tps, nous avons essayé de se familiariser avec des différentes architectures des réseaux de neurones convolutionnels, de les apprendre pour les adapter à des problèmes de classification d'images et de modifier les paramètres internes pour améliorer les performances et surmonter les problèmes de sur-apprentissage .

References

[1] Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, Vol 65(6), Nov 1958, 386-408.

[2] P. FOUCHER, P. REVOLLON, B. VIGOUROUX. Segmentation d'images en couleurs par réseau de neurones : application au domaine végétal.

[3] https://blogs.msdn.microsoft.com/mlfrance/2016/04/28/une-premiere-introduction-au-deep-learning/

[4] Y. LeCun and Y. Bengio: Convolutional Networks for Images, Speech, and Time-Series, in Arbib, M. A. (Eds), The Handbook of Brain Theory and Neural Networks, MIT Press, 1995,

[5] Teh, Y. and Hinton, G. E. (2001). Rate-coded restricted Boltzmann machines for face recognition. In Advances in Neural Information Processing Systems, volume 13.

[6] Yann LeCun, « LeNet-5, convolutional neural networks »

[7] Pierre Buyssens, Marinette Revenu, Olivier Lepetit. Réseau de Neurones Convolutionnels pour la Reconnaissance Faciale Infrarouge. GRETSI'09, Sep 2009, Dijon, France. 4 p., 2009. <hal-00812603>

[8] Matthew Lai, Daniel. Deep Learning for Medical Image Segmentation. Rueckert Apr 29, 2015

[9] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In ICANN, pages 92–101, 2010.

[10] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details : Delving deep into convolutional nets. In Proceedings of the British Machine Vision Conference (BMVC), 2014.