Université Mohammed V
Faculté des Sciences
Rabat

L'apprentissage profond pour la vision par ordinateur

Mehdi Bouskri

mehdi_bouskri@um5.ac.ma

L'opération de convolution

La différence fondamentale entre une couche dense et une couche de convolution est la suivante :

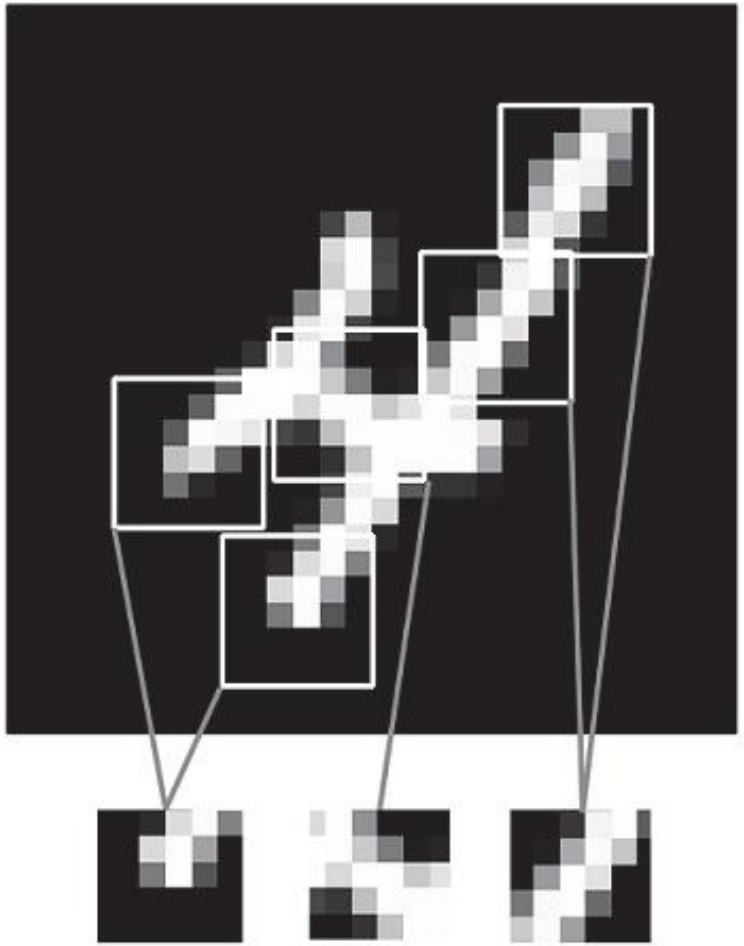
les couches denses apprennent des motifs globaux dans l'espace des caractéristiques d'entrée, tandis que les couches de convolution apprennent des motifs locaux.

L'opération de convolution

C'est à dire, des motifs découverts dans de petites fenêtres 2D des entrées.

Les images peuvent être décomposées en motifs locaux tels que les bordures, les textures, etc.

Dans notre exemple, ces fenêtres étaient toutes 3x3.



L'opération de convolution

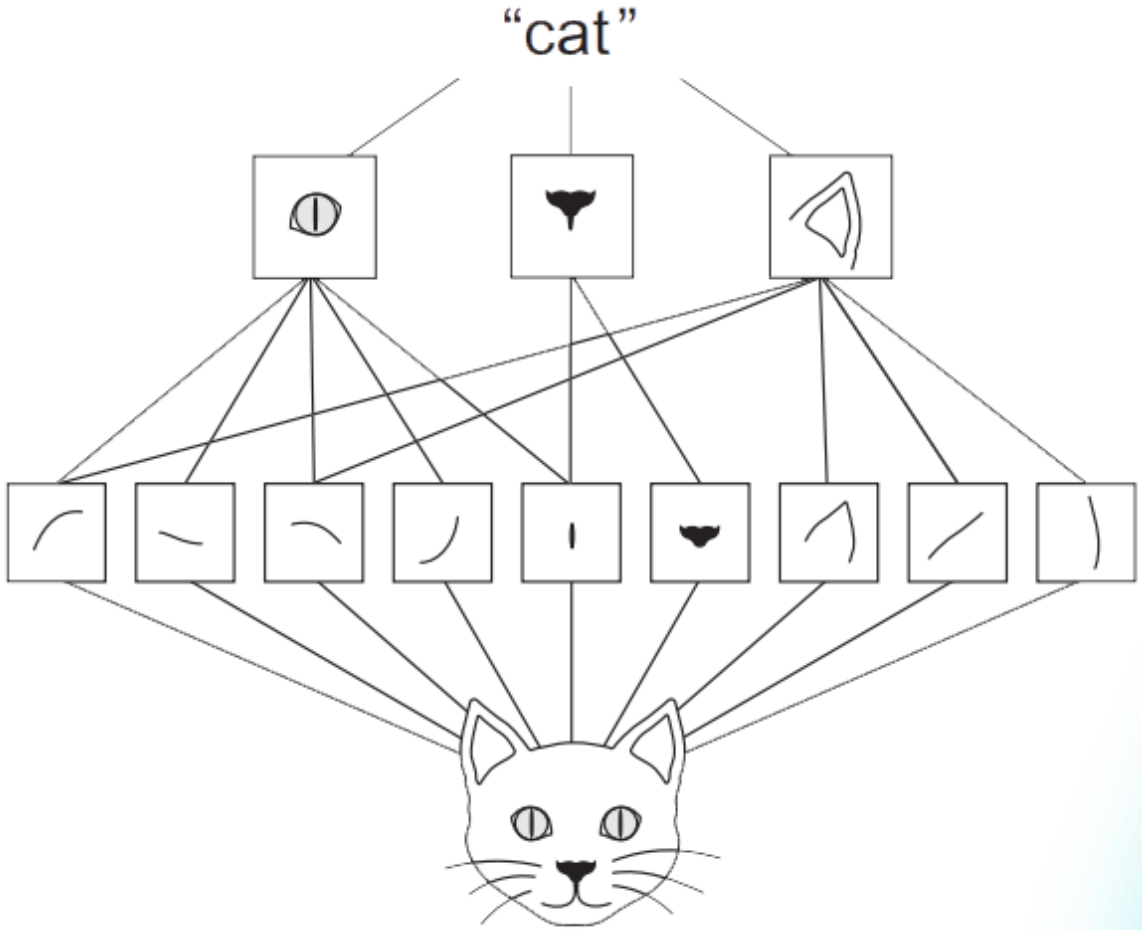
Cette caractéristique essentielle donne aux convnets deux propriétés importantes :

- Les motifs qu'ils apprennent sont invariants en fonction de la localisation. Cela rend les convnets très efficaces en termes de traitement des images.

- Ils peuvent apprendre des hiérarchies spatiales de motifs.

hiérarchies spatiales de motifs

Une première couche de convolution apprendra de petits motifs locaux tels que les bords, mais une deuxième couche de convolution apprendra des motifs plus grands constitués des caractéristiques de la première couche et ainsi de suite.



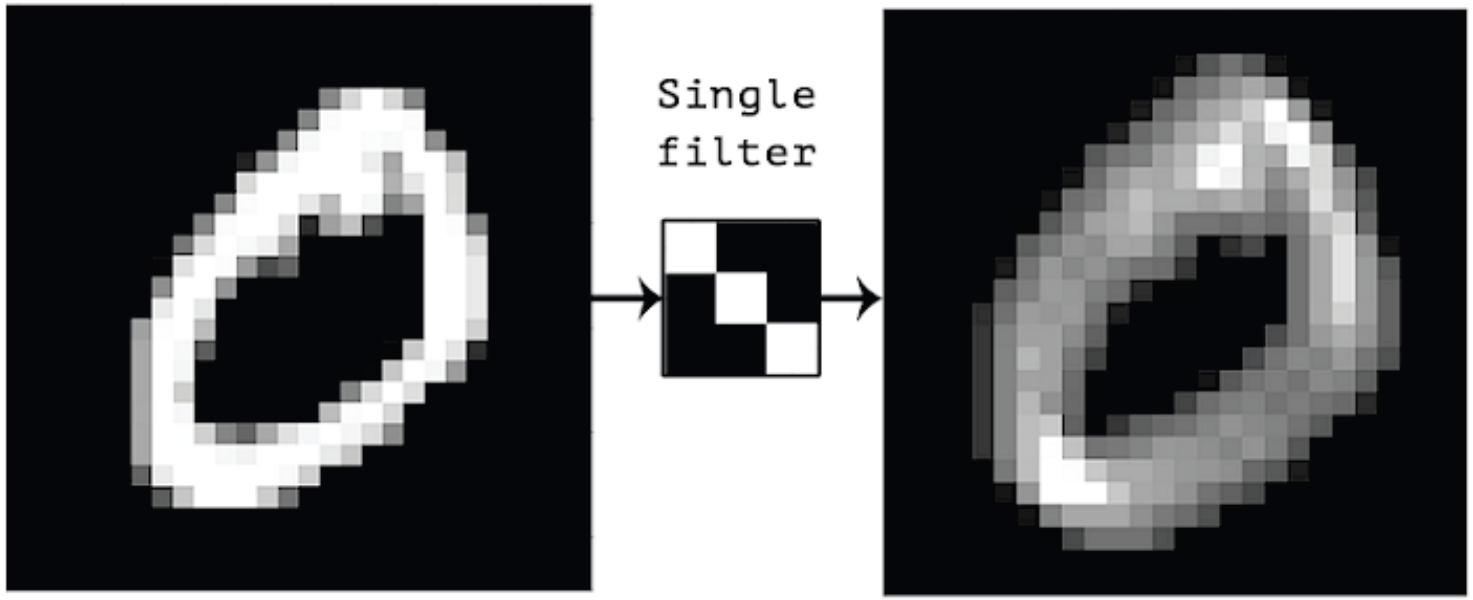
L'opération de convolution

Les convolutions opèrent sur des tenseurs 3D, appelés cartes de caractéristiques, elles extraient des blocs de ces cartes de caractéristiques d'entrée et appliquent une même transformation à tous ces blocs , produisant ainsi une carte de caractéristiques de sortie.

Cette carte de caractéristiques de sortie est toujours un tenseur 3D, elle a une largeur et une hauteur, mais sa profondeur est arbitraire, puisqu'elle est un paramètre de la couche, et les différentes dimensions de cet axe de profondeur ne représentent plus des couleurs spécifiques, mais plutôt ce que nous appelons des filtres.

Les filtres

Les filtres encodent des aspects spécifiques des données d'entrée. De manière générale, un seul filtre peut encoder le concept de "présence d'un visage dans l'entrée".



L'opération de convolution

Les convolutions sont définies par deux paramètres clés :

- La taille des blocs qui sont extraits des entrées (typiquement 3x3 ou 5x5). Dans notre exemple, il s'agissait toujours de 3x3, ce qui est un choix très courant.
- La profondeur de la carte des caractéristiques de sortie, c'est-à-dire le nombre de filtres calculés par la convolution.

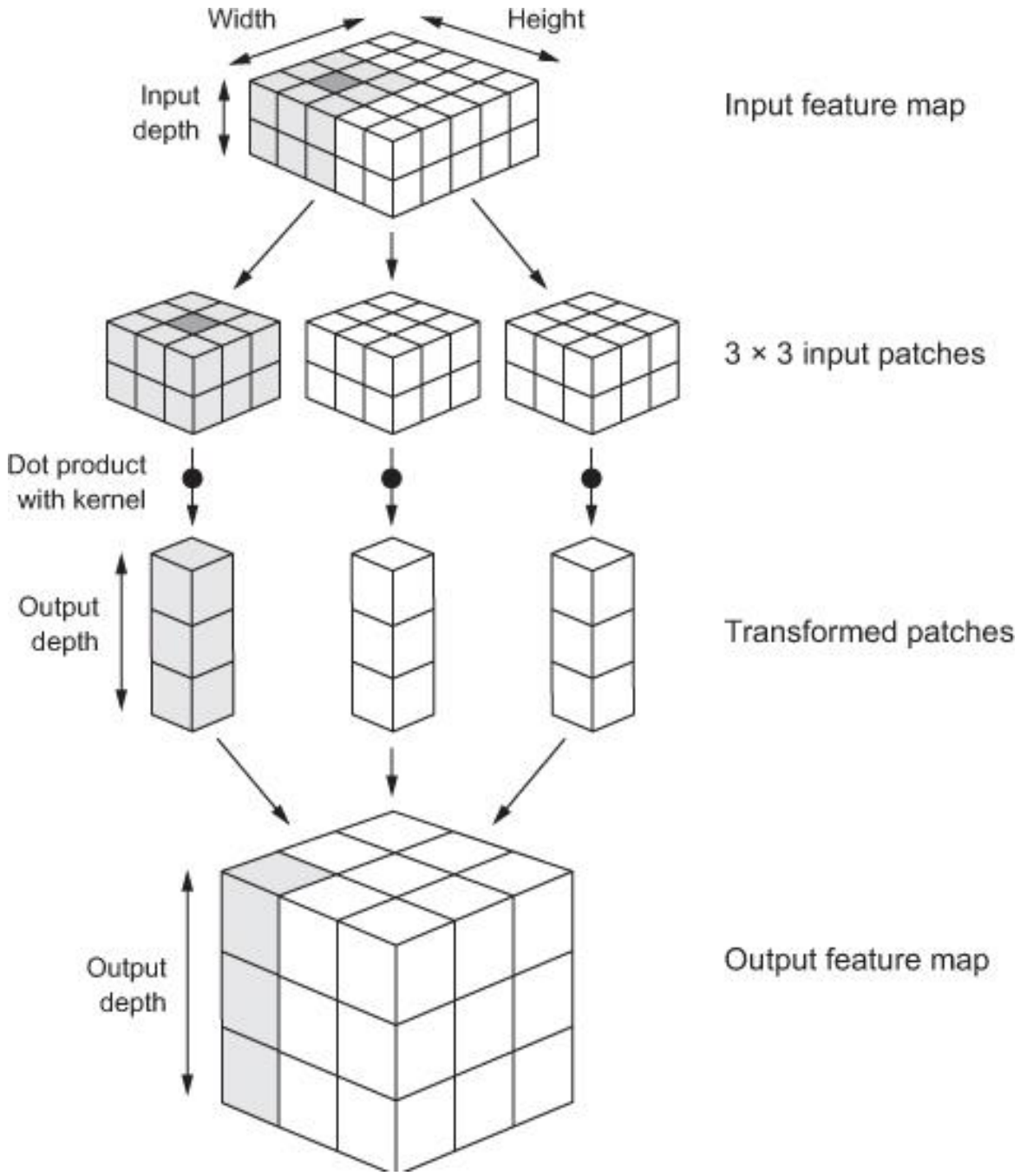
Dans les couches Conv2D de Keras, ces paramètres sont les premiers arguments passés à la couche :

```
Conv2D(output_depth, (window_height, window_width))
```

L'opération de convolution

Une convolution fonctionne en:

1. Glissant des fenêtres de taille 3x3 ou 5x5 sur la carte de caractéristiques 3D d'entrée, en s'arrêtant à chaque position possible, et en extrayant le bloc 3D des caractéristiques qui l'entourent.
2. Chaque zone 3D est ensuite transformée en un vecteur 1D via un produit tensoriel avec la matrice de poids ou le noyau de convolution.
3. Tous ces vecteurs sont ensuite réassemblés spatialement en une carte de sortie 3D.


Width
Height
Input depth
Input feature map
3 × 3 input patches
Dot product with kernel
Output depth
Transformed patches
Output depth
Output feature map

L'opération de convolution

Notez que la largeur et la hauteur de la sortie peuvent différer de celles de l'entrée pour deux raisons :

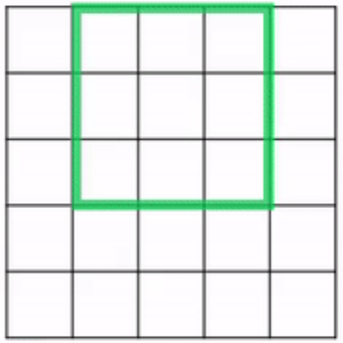
- Les effets de bords, qui peuvent être contrés par le remplissage de la carte de caractéristiques d'entrée.

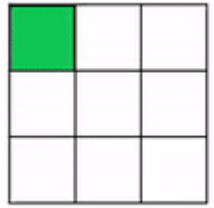
- L'utilisation de strides .

Effets de bord et le remplissage

Considérons une carte de caractéristiques 5x5 (25 cases au total). Il n'y a que 9 carreaux différents autour desquels on peut centrer une fenêtre 3x3.

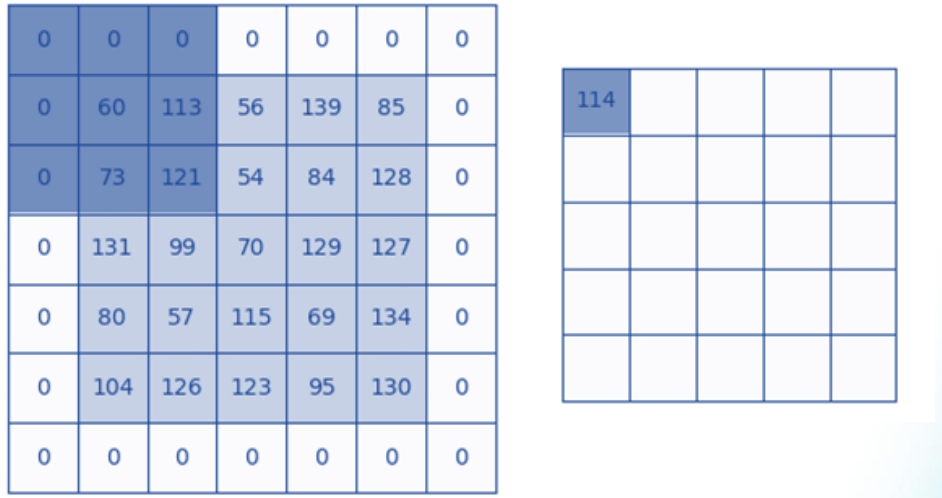
En conséquence, la carte de caractéristiques de sortie sera de 3x3. Elle est réduite par deux cases le long de chaque dimension dans ce cas.





Effets de bord et le remplissage

Le remplissage consiste à ajouter un nombre approprié de lignes et de colonnes de chaque côté de la carte de caractéristiques d'entrée afin de permettre l'insertion des fenêtres de convolution centrales autour de chaque carré d'entrée.



Effets de bord et le remplissage

Dans les couches Conv2D, le remplissage est configurable via l'argument `padding`, qui prend deux valeurs : "`valid`", qui signifie qu'il n'y a pas de remplissage (seuls les emplacements de fenêtre "valides" seront utilisés), et "`same`", qui signifie qu'il faut remplir de manière à avoir une sortie avec la même largeur et hauteur que l'entrée. La valeur par défaut de l'argument `padding` est "`valid`".

Strides de convolution

Le second facteur qui peut influencer la taille de la sortie est la notion de " stride". Dans notre description de la convolution jusqu'à présent, nous avons supposé que les carreaux centraux de la convolution étaient tous adjacents.

Cependant, la distance entre deux fenêtres successives est en fait un paramètre de la convolution appelé son "stride", qui par défaut est égal à un.

Strides de convolution

L'utilisation de ‘stride’ égale à 2 signifie que la largeur et la hauteur de la carte de caractéristiques sont réduites par un facteur de 2, en plus des changements induits par l’effet de bords.

Pour réduire la taille des cartes de caractéristiques, au lieu des ‘strides’, on utilise généralement l'opération "max pooling", que nous avons vue en action dans notre premier exemple de convnet.

L'opération de max pooling

Le rôle du max pooling est de diminuer agressivement la taille des cartes de caractéristiques, un peu comme les convolutions avec strides.

Le max pooling consiste à extraire des fenêtres des cartes de caractéristiques d'entrée et de sortir la valeur maximale de chaque canal. Le concept est similaire à celui de la convolution, sauf qu'au lieu de transformer les blocs locaux via une transformation linéaire apprise, ils sont transformés via une opération tensorielle programmée par défaut.

L'opération de max pooling

Deux choses sont problématiques dans une configuration sans maxpooling2D:

- Elle n'est pas adaptée à l'apprentissage d'une hiérarchie spatiale des caractéristiques.

- La carte de caractéristiques finale comporte 22*22*64 = 31 000 coefficients au total par élément. Si on aplatit la carte pour y ajouter une couche Dense de taille 512, cette couche aurait 15,8 millions de paramètres . Ce nombre est énorme pour un petit modèle, et entraînerait un surajustement intense.

L'opération de max pooling

La raison qui justifie la réduction des tailles des cartes de caractéristiques est simplement de réduire le nombre de coefficients à traiter, ainsi que d'induire des hiérarchies spatiales en faisant en sorte que les couches de convolution successives traitent des fenêtres de plus en plus grandes, en termes de proportion de l'entrée originale qu'ils couvrent.

L'opération de max pooling

Notez que le max pooling n'est pas le seul moyen de réaliser une telle opération. Vous pouvez également utiliser des strides dans la couche de convolution, et vous pouvez aussi utiliser average pooling au lieu du max pooling, où chaque bloc d'entrée local est transformé en prenant la valeur moyenne.

Cependant, le max pooling a tendance à fonctionner mieux que ces solutions alternatives.

L’entraînement d'un convnet sur un petit ensemble de données

La nécessité d'entraîner un modèle de classification d'images en utilisant peu de données est une situation courante, que vous rencontrerez probablement dans la pratique si vous faites de la vision par ordinateur.

Avoir "peu" de données peut signifier de quelques centaines à quelques dizaines de milliers d'images. Comme exemple pratique, on va essayer de classifier des images en "chiens" ou "chats" d’un ensemble de données contenant 4000 images de chats et de chiens (2000 chats, 2000 chiens). Nous utiliserons 2000 images pour l'apprentissage, 1000 pour la validation, et enfin 1000 pour le test.

Téléchargement des données

L'ensemble de données ‘cats vs dogs’ que nous allons utiliser n'est pas fourni avec Keras. Il a été mis à disposition par Kaggle.com dans le cadre d’une compétition de vision par ordinateur en 2013.

http://www.kaggle.com/c/dogs-vs-cats/data

Le jeu de données original contient 25 000 images de chiens et de chats. On va créer un nouvel ensemble de données contenant trois sous-ensembles : un ensemble pour l'apprentissage avec 1000 échantillons de chaque classe, un ensemble de validation avec 500 échantillons de chaque classe et enfin un ensemble de test avec 500 échantillons de chaque classe.

Construction du réseau

Puisque nous traitons des images plus grandes et un problème plus complexe que celui de MNIST, on va élargir notre réseau pour à la fois augmenter la capacité du réseau, et réduire encore la taille des cartes de caractéristiques avant d'atteindre la couche d'aplatissement.

Comme nous avons à un problème de classification binaire, nous terminons le réseau avec une seule unité et une fonction activation sigmoid.

Prétraitement des données

Actuellement, nos données se trouvent sur un disque dur sous forme de fichiers JPEG, donc les étapes pour les introduire dans notre réseau sont globalement les suivantes :

- Lire les fichiers images et décoder le contenu JPEG en matrices de pixels RBG.
- Convertir celles-ci en tenseurs à virgule flottante.
- Remettre les valeurs des pixels (entre 0 et 255) à l'intervalle [0, 1].
- Redimensionner toutes les images à une résolution de 150x150.

Prétraitement des données

Keras dispose d'un module contenant des outils d'aide au traitement des images, `keras.preprocessing.image`. Il contient notamment la classe ImageDataGenerator qui permet de configurer rapidement des générateurs Python capables de transformer automatiquement des fichiers d'images en lots de tenseurs prétraités.

L’entraînement du modèle

Nous utiliserons la méthode fit, qui supporte les générateurs de données. Les données étant générées de manière infinie, le générateur doit savoir combien d'échantillons il doit tirer de l’ensemble d’apprentissage avant de déclarer une époque terminée. C'est le rôle de l'argument steps_per_epoch.

On peut passer l'argument validation_data, et donc il faut aussi spécifier l'argument validation_steps.

L'utilisation de l'augmentation des données

L'augmentation des données consiste à générer plus de données d'apprentissage à partir de données existantes, en augmentant les échantillons via un certain nombre de transformations aléatoires qui produisent des images d'apparence acceptable.
L'objectif est qu'au moment de son apprentissage, le modèle ne soit jamais confronté deux fois à la même image. Cela permet au modèle d'être exposé à plus d'aspects des données et de généraliser encore mieux.

L'utilisation de l'augmentation des données

Cependant, les entrées que le modèle ‘voit’ sont encore fortement corrélées puisqu'elles proviennent d'un petit nombre d'images originales.

Pour éviter encore le surajustement, il faut ajouter également une couche de Dropout à notre modèle, juste avant le classificateur entièrement connecté.

L’utilisation d'un convnet préentraîné

Un réseau préentraîné est simplement un réseau enregistré préalablement entraîné sur un grand ensemble de données, généralement sur une tâche de classification d'images à grande échelle.
Si cet ensemble de données original est suffisamment grand et général, la hiérarchie de caractéristiques spatiales apprise par le réseau préentraîné peut servir à titre de modèle générique de notre monde visuel, et donc ses caractéristiques peuvent s'avérer utiles pour de nombreux problèmes de vision par ordinateur, même si ces nouveaux problèmes peuvent impliquer des classes complètement différentes de celles de la tâche originale.

L’utilisation d'un convnet préentraîné

Dans notre cas, nous considérerons un grand convnet entraîné sur le jeu de données ImageNet (1,4 million d'images étiquetées et 1000 classes différentes).

ImageNet contient de nombreuses classes d'animaux, y compris différentes espèces de chats et de chiens. On peut donc s'attendre à obtenir de très bons résultats dans notre problème.

Nous utiliserons l'architecture VGG16, une architecture convnet simple et largement utilisée pour ImageNet. Bien qu'il s’agit d'un modèle loin de l'état de l’art actuel, son architecture est similaire à ce que nous avons utilisé jusqu'à présent, et facile à comprendre sans introduire de nouveaux concepts.

L'extraction de caractéristiques

L'extraction de caractéristiques consiste à utiliser les représentations apprises par un réseau préalable pour extraire des caractéristiques intéressantes de nouveaux échantillons. Ces caractéristiques sont ensuite soumises à un nouveau classificateur, qui est entraîné de manière normale.

L'extraction de caractéristiques

Il y a deux façons possibles de procédé:

- L'exécution de la base convolutive sur notre ensemble de données, l'enregistrement de sa sortie dans un tableau Numpy, puis on utilise ces données comme entrée d'un classificateur indépendant.

- L'extension du modèle que nous avons en ajoutant des couches denses, et en exécutant le modèle entier sur les données d'entrée.

Fine-tuning

Le fine-tuning consiste à débloquer quelques couches supérieures d'un modèle de base utilisé pour l'extraction de caractéristiques, et à entraîner à la fois la partie du modèle nouvellement ajoutée et ces couches débloquées.

Fine-tuning

les étapes du fine-tuning d'un réseau sont les suivantes:

1. Ajouter un réseau personnalisé à un réseau de base déjà entraîné.
2. Bloquer le réseau de base.
3. Entraîner la partie ajoutée.
4. Débloquer certaines couches du réseau de base.
5. Entraîner conjointement ces couches avec la partie ajoutée.

Fine-tuning

On peut affiner plus de couches ou l'ensemble de la base convolutive, mais nous devons considérer que :

- Les premières couches de la base convolutive encodent des caractéristiques plus génériques et réutilisables, tandis que les couches supérieures encodent des caractéristiques plus spécialisées.
- Plus nous entraînons de paramètres, plus nous risquons de tomber dans le surajustement.

Conclusion

Ce que vous devez retenir de cette partie:

- Les convnets sont le meilleur type de modèles d'apprentissage automatique pour les tâches de vision par ordinateur.
- Sur un petit ensemble de données, le surajustement sera le principal problème. L'augmentation des données est un moyen puissant pour réduire le surajustement lorsque l'on travaille avec des images.
- Il est facile de réutiliser un convnet existant sur un nouveau jeu de données, via l'extraction de caractéristiques.
- En complément de l'extraction de caractéristiques, on peut utiliser le fine-tuning, qui adapte à un nouveau problème certaines des représentations précédemment apprises par un modèle existant

Visualisation des informations apprises par les convnets

Nous allons aborder trois des plus utiles techniques de visualisation;

1. Visualisation des sorties intermédiaires de convnet. Ceci est utile pour comprendre comment les couches successives de convnet transforment leur entrée, et pour avoir une idée de la signification des filtres individuels de convnet.
2. Visualisation des filtres convnets. Ceci est utile pour comprendre précisément à quels motif ou concept visuels chaque filtre d'un convnet est réceptif.
3. Visualisation des heatmaps de l'activation des classes dans une image. Ceci est utile pour comprendre quelle partie d'une image a été identifiée comme appartenant à une classe donnée, et permet ainsi de localiser des objets dans des images.

Visualisation des activations intermédiaires

La visualisation des activations intermédiaires consiste à visualiser les cartes de caractéristiques produites par les différentes couches de convolution et de pooling d'un réseau, pour une certaine entrée.
Cela permet de voir comment une entrée est décomposée en différents filtres appris par le réseau

Visualisation des activations intermédiaires

Quelques éléments importants à noter:

1. La première couche agit comme une collection de divers détecteurs de bords.
2. Au fur et à mesure que l'on progresse, les activations deviennent de plus en plus abstraites et moins visuellement interprétables.
3. La dispersion des activations augmente avec la profondeur de la couche.

Visualisation des filtres convnet

Une autre façon simple d'inspecter les filtres appris par les convnets est d'afficher le motif visuel auquel chaque filtre est supposé de répondre.
On va construire une fonction de perte qui maximise la valeur d'un filtre dans une couche de convolution, puis on va utiliser la descente de gradient stochastique pour ajuster les valeurs de l'image d'entrée de façon à maximiser cette valeur d'activation.

Visualisation des heatmaps de l'activation des classes

Cette technique est utile pour comprendre quelles parties d'une image donnée ont conduit un convnet à sa décision finale de classification.

Cette catégorie générale de techniques est appelée visualisation de la "carte d'activation de classe" (CAM), et consiste à produire des heatmaps d'activation de classe sur les images d'entrée.