CR 4

Premiers pas sur les réseaux de neurones.

La première étape est de bien comprendre comment fonctionnent les réseaux de neurones, voici un petit résumé de plusieurs guides trouvables sur internet.

Les réseaux de neurones convolutifs comportent quatre types de couches différentes, qui ont chacune différentes entrées et sorties. Le réseau est alors une combinaison de ces différentes couches.

-> Couche de convolution: (particularité aux RNN convolutifs cad pour les images)
Extraire les features des images: filtrage par convolution, traitement par fenêtre .
On calcule le produit de convolution entre la fenêtre et chaque portion de l'image.

feature <-> filtre.

La couche de convolution reçoit plusieurs images et calcule la convolution de chacune d'entre elles avec chaque filtre.

On obtient alors une paire (image,filtre)=(feature,map) qui indiquent où se trouvent les features dans l'image.

---> Comment choisir les features ? Apprises par le réseau !

En entrée:

Une image W x H x D

4 HYPERPARAMÈTRES:
1- Nbr de filtres **K**
2- Taille des filtres **F**: (F x F x D)
3- le pas **S**. (de la fenêtre)
4- le zero-padding **P**: contour de l'image en entrée d'épaisseur P?

En sortie:

matrice de dimensions Wc x Hc x Dc où
Wc=(W-F+2P)/S+1
Hp=(H-F+2p)/S+1
Dp=K.

On peut par exemple choisir P=(F-1)/2 et S=1 pour avoir des feature maps de même largeur et hauteur que celles reçues en entrée.

-> Couche de pooling:

reçoit plusieurs feature maps et applique à chacune d'entre elles un pooling -> réduire la taille des images en préservant les caractéristiques importantes.

→ On découpe l'image en cellules régulières (de même taille), on garde pour chaque cellule la valeur maximale (de quoi ?) (2x2 ou 3x3 avec pas de 2px).

On diminue le nombre de calculs et on évite le sur-apprentissage.

En entrée:
Deux Hyperparamètre:
1- la taille des cellules **F**
2- Le pas **S**

En sortie:
Une matrice de dimensions Wp x Hp x Dp
avec Wp= (W-F)/2+1
Hp=(H-F)/S+1
Dp=D.

-> Couche de correction ReLu:

(Rectified Linear Units) ReLU(x)=max(0,x)

remplace les valeurs négatives par des 0.

-> La couche fully-connected:

pas caractéristique d'un CNN,

reçoit un vecteur en entrée et produit un vecteur en sortie en appliquant une combinaison linéaires (et éventuellement une fonction d'activation)

permet de classifier l'image: renvoie un vecteur de taille N (nombre de classes ex: chien, chat) chaque élément indique la probabilité pour l'image en entrée d'appartenir à une classe. le reste c'est du réseau de neurones classique.

Maintenant que nous savons comment fonctionne un réseau de neurones il faut l'implémenter (en cours), utiliser une base pré entrainée (par exemple VGG16) et analyser les résultats.

Pour le moment, nous pensons suivre le réseaux de

https://www.researchgate.net/publication/332949646_A_Passive_Approach_for_Detecting_Image_Splicing_using_Deep_Learning_and_Haar_Wavelet_Transform

qui demande ensuite d'appliquer les ondelletes de Har et la SVM mais c'est possible qu'on utilise plutôt
https://www.researchgate.net/publication/338957097_Deep_Learning_Local_Descriptor_for_Image_Splicing_Detection_and_Localization