Réseau de neurone convolutionnel

Version du 13 avril 2022, 14:29

▷ **Exercice 1** Dans cet exercice il s'agit de comparer les performances de 2 réseaux de neurones. Les tableaux ci desous présente les sorties du réseau de neurones ainsi que les labels associés. L'entropie croisée moyenne $H(y, \hat{y})$ est ici définie par

Table 1 – Réseau 1

0.04	0.2	0.7	0.06	1	0	0	0
0.02	0.06	0.12	0.8	0	0	0	1
0.6	0.1	0.2	0.1	1	0	0	0
0.13	0.1	0.7	0.07	0	0	1	0
0.53	0.18	0.17	0.12	1	1	1	0

Table 2 – Réseau 2

0.3	0.1	0.51	0.09	1	0	0	0
0.16	0.02	0.02	0.8	0	0	0	1
0.4	0.1	0.2	0.3	1	0	0	0
0.07	0.02	0.9	0.01	0	0	1	0
0.6	0.2	0.1	0.1	1	1	1	0

$$H(y, \hat{y}) = -\frac{1}{5} \sum_{j=1}^{5} \sum_{i=1}^{4} y_{ij} log_2(\hat{y}_{ij})$$

pour N données (ici $N = 5$) avec y_k le label et $\hat{y}_k$ la probabilité pour d'une classe k. Ici nous avons 4 classes.

1. Calculer l'entropie croisée pour les 2 réseaux. Quel algorithme possède la meilleure performance au regard de ce critère ?
2. Reprendre l'étude pour la MSE (mean square error) et la MAE (Mean absolute error)
3. En supposant que l'on attribue :
   - le niveau 1 à une valeur de sortie du réseau supérieure à 0,5
   - le niveau 0 à une sortie du réseau inférieure à 0,5

   en déduire le nombre de vrais positifs, de faux positifs, la précision, et le rappel

	Tableau 1		Tableau 2		MAE1	MSE1	MAE2	MSE2
	0,04	-4,6439	0,3	-1,737	0,96	0,9216	0,7	0,49
	0,2	0	0,1	0	0,2	0,04	0,1	0,01
	0,7	0	0,51	0	0,7	0,49	0,51	0,2601
	0,06	0	0,09	0	0,06	0,0036	0,09	0,0081
	0,02	0	0,16	0	0,02	0,0004	0,16	0,0256
	0,06	0	0,02	0	0,06	0,0036	0,02	0,0004
	0,12	0	0,02	0	0,12	0,0144	0,02	0,0004
	0,8	-0,3219	0,8	-0,3219	0,2	0,04	0,2	0,04
	0,6	-0,737	0,4	-1,3219	0,4	0,16	0,6	0,36
	0,1	0	0,1	0	0,1	0,01	0,1	0,01
	0,2	0	0,2	0	0,2	0,04	0,2	0,04
	0,1	0	0,3	0	0,1	0,01	0,3	0,09
	0,13	0	0,07	0	0,13	0,0169	0,07	0,0049
	0,1	0	0,02	0	0,1	0,01	0,02	0,0004
	0,7	-0,5146	0,9	-0,152	0,3	0,09	0,1	0,01
	0,07	0	0,01	0	0,07	0,0049	0,01	0,0001
	0,53	-0,9159	0,6	-0,737	0,47	0,2209	0,4	0,16
	0,18	-2,4739	0,2	-2,3219	0,82	0,6724	0,8	0,64
	0,17	-2,5564	0,1	-3,3219	0,83	0,6889	0,9	0,81
	0,12	0	0,1	0	0,12	0,0144	0,1	0,01
Entropie croisée		**-2,4327**		**-1,9827**	**0,3725**	**0,21575**	**0,3375**	**0,18563**

	Tableau 1		Tableau 2	
VP	**4**	**VP**	**3**
FP	**1**	**FP**	**1**
VN	**10**	**VN**	**10**
FN	**3**	**FN**	**4**
Précision	**0,80**		**0,75**
Recall	**0,57**		**0,42857**

Figure 1 – Correction exercice 1

Conclusion : Le second algorithme est meilleur selon tous les critères.

▷ **Exercice 2** Dans un réseau de neurone convolutif il est nécessaire de paramétrer le padding et le stride.

Dans la bibliothèque Python theano (http ://deeplearning.net/software/theano/tutorial/conv_arithmetic.h plusieurs configurations sont paramétrables.

Par exemple en fixant un padding tel que $Zero\_padding = \frac{K-1}{2}$ avec K la taille du filtre et un pas (stride) de 1 alors l'entrée et la sortie du filtre ont la même dimension.

Soit l'entrée $M = \begin{bmatrix} 1 & 4 & 2 & 3 & 2 \\ 1 & 2 & 4 & 1 & 6 \\ 2 & 2 & 4 & 0 & 1 \\ 2 & 5 & 1 & 3 & 4 \\ 3 & 5 & 4 & 3 & 0 \end{bmatrix}$ et le filtre $K = \begin{bmatrix} 2 & 0 & 2 \\ 3 & 5 & 1 \\ 0 & 1 & 4 \end{bmatrix}$

1. Quelle doit être la valeur de $Zero\_padding$ pour que les données d'entrée (M) et la sortie ($M \star K$) ont la même dimension ($\star$ est l'opérateur de convolution) avec un stride=1.
   Correction :$Zero\_padding = \frac{K-1}{2} = 1$
2. Déterminez le résultat de la convolution de la matrice M par le filtre K, pour les 3 cas suivants :
   — a. stride = 1 , no padding
   Correction : $M \star K = \begin{bmatrix} 41 & 45 & 35 \\ 39 & 45 & 52 \\ 65 & 43 & 35 \end{bmatrix}$
   — b. stride = 2 , padding = 2
   Correction : $M \star K = \begin{bmatrix} 4 & 12 & 11 & 0 \\ 11 & 41 & 35 & 22 \\ 18 & 65 & 35 & 14 \\ 6 & 14 & 8 & 0 \end{bmatrix}$
   — c. stride = 1 , half padding (nombre de colonnes fitre)//2)
   Correction : $M \star K = \begin{bmatrix} 18 & 43 & 33 & 48 & 25 \\ 25 & 41 & 45 & 35 & 40 \\ 38 & 39 & 45 & 52 & 11 \\ 42 & 65 & 43 & 35 & 29 \\ 30 & 44 & 54 & 37 & 15 \end{bmatrix}$

▷ **Exercice 3** LeNet5 de la figure ci-dessous est un réseau de neurones convolutionnel comportant :

— 3 couches de convolution (C1, C2, C3) utilisant des matrices de convolution 5*5
— 2 couches (S2, S4) de sous-échantillonnage de facteur 2
— Un MLP totalement connecté (F6)

Ce réseau permet de classifier des chiffres manuscrits variant de 0 à 9. Les imagettes d'entrée possèdent 32*32 pixels.

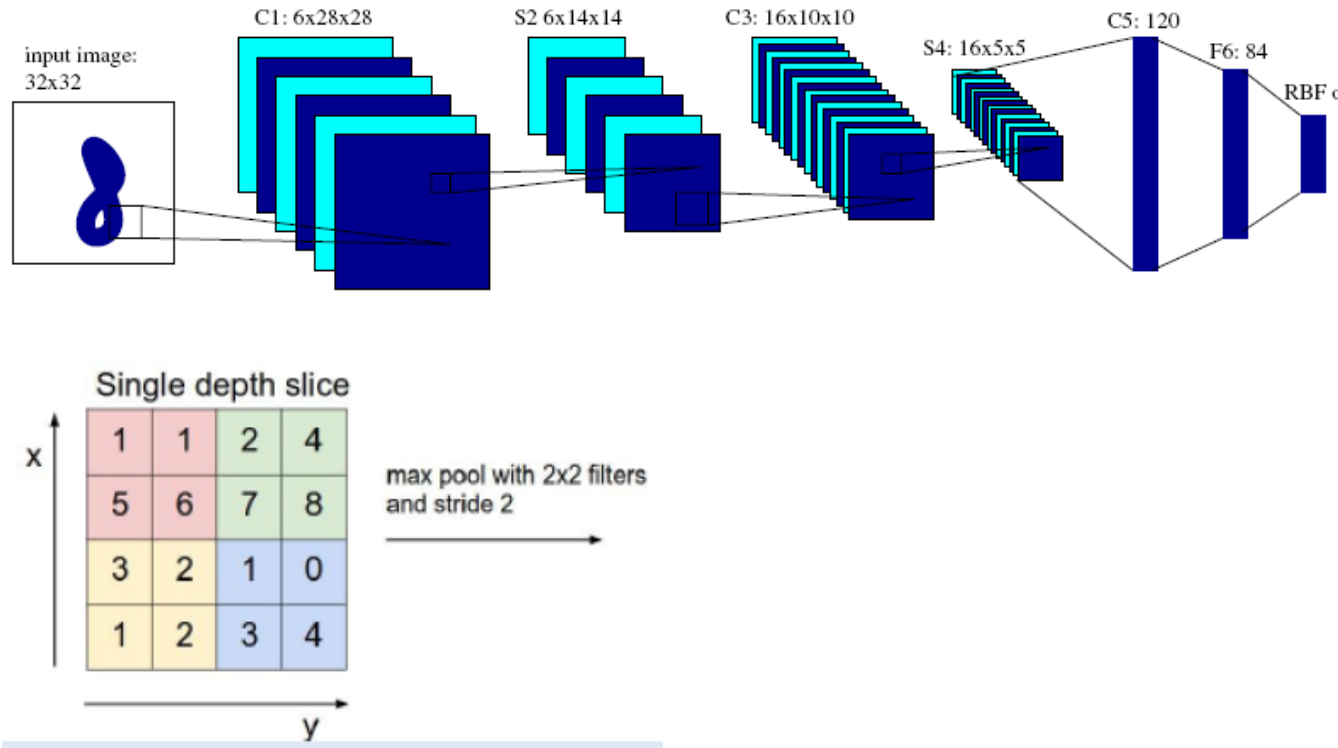

FIGURE 2 – Pooling

1. Quels sont les avantages d'un réseau de neurone convolutionnel par rapport à un réseau multicoucle "classique" qui comporterait 32*32=1024 entrées ?

   Correction : un réseau convolutionnel comporte beaucoup moins de paramètres que le réseau multicouches. Par exemple pour la couche C1 il y a 156 paramètres pour le réseau convolutionnel tandis qye le résau multicouche comprend 32*32*128+128 =131200 paramètres. Ensuite une couche de convolution permet d'exhiber une "feature" par noyau de convolution et de hiérarchiser le niveau sémantique des "features" un peu comme opère le cerveau.

2. La première couche du réseau (C1) est de taille 6*28*28. Expliquer pourquoi la taille de chaque caractéristique est de 28*28. **Correction :** En l'absence de padding, et en présence d'un noyau 5*5, deux lignes sont supprimmées en haut et en bas de l'image ainsi que deux colonnes à gauche et à droite. Il en résulte une ilage de taille (32-4)*(32-4)

3. Donner le nombre total de paramètres de la couche C1. **Correction :** 5*5*6+6= 156 paramètres (https ://engmrk.com/lenet-5-a-classic-cnn-architecture/)

4. Donner le résultat du sous-échantillonnage sur l'exemple de la figure 2.

   Correction :

6	8
3	4

5. En déduire l'intérêt du sous-échantillonage ou pooling. **Correction :** Le pooling

permet de limiter le nombre de paramètres, de retenir les valeurs les plus significatives et aussi de rendre l'algorithme plus robuste aux changements d'orientation et d'échelle.

6. Quel est l'intérêt dropout dans un réseau convolutionnel ? **Correction :** Le dropout permet d'inhiber un certains taux de neuroenes afin d'éviter le problème du surapprentissave ou overfiffing bien connu en machine learning.

7. Expliquer le fonctionnement général de ce réseau en vous aidant de la documentation sur le site : yann.lecun.com/exdb/lenet/

8. Quel est l'intérêt du dropout ? Le dropout permet d'éviter le surapprentissage.

▷ **Exercice 4** Soit l'architecture du réseau convolutionnel décrite sur la figure 3.

```
model = Sequential()
model.add(Conv2D(32, (3,3), strides=(2,2), activation='relu', padding='same',
input_shape=input_shape))
model.add(Dropout(0.25))
model.add(Conv2D(64, (5,5), strides=(1,1), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3,3), strides=(1,1), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(256, (3,3), strides=(1,1), activation='relu', padding='same'))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

FIGURE 3 – Réseau de neurones convolutionnels

1. Combien y a-t-il de paramètres (appris, stockés) dans l'architecture de la figure 3 où le nombre de classes est 10 ($num\_classes = 10$) ?

   Pour information vous trouverez sur ce lien le détail du calcul du nombre de paramètres.

   https ://towardsdatascience.com/understanding-and-calculating-the-number-of-parameters-in-convolution-neural-networks-cnns-fc88790d530d

   La figure 4 donne un exemple pour une image couleur en entrée de taille $32 \times 32 \times 3$.

   — **Pour des données de taille (28, 28, 1)** ?
   Correction :

SL.No		Activation Shape	Activation Size	# Parameters
1.	Input Layer:	(32, 32, 3)	3072	0
2.	CONV1 (f=5, s=1)	(28, 28, 8)	6272	608
3.	POOL1	(14, 14, 8)	1568	0
4.	CONV2 (f=5, s=1)	(10, 10, 16)	1600	3216
5.	POOL2	(5, 5, 16)	400	0
6.	FC3	(120, 1)	120	48120
7.	FC4	(84, 1)	84	10164
8.	Softmax	(10, 1)	10	850

FIGURE 4 – Exemple de calcul de paramètres

Couche	Act. shape	Act. size	Paramètres
Couche entrée	$(28, 28, 1)$	784	0
Couche 1(Conv2D, 32 stride (2,2))	$(14, 14, 32)$	6272	$32 * 9 + 32 = 320$
Couche 2 (dropout :0.25)			
Couche 3 (Conv2D,64)	(14,14,64)	12544	$64 * 32 * 25 + 64 = 51264$
Couche 4 (Poul2D(2,2))	$(7, 7, 64)$	3136	0
Couche 5 (Conv2D,128)	(7,7,128)	6272	$128 * 64 * 9 + 128 = 73856$
Couche 6 (Poul2D(2,2))	$(3, 3, 128)$	1152	0
Couche 7 (Conv2D,256)	(3,3,256)	2304	$256 * 128 * 9 + 256 = 295268$
Couche 8 (dropout :0.25)			
Couche 9 (Dense 256)	$(256, 1)$	256	2304*256+256=590080
Couche 10 (Dense)	$(10, 1)$	10	2570

— **Pour des données de taille (64, 64, 3) ?**
Correction :

Couche	Act. shape	Act. size	Paramètres
Couche entrée	$(64, 64, 3)$	12288	0
Couche 1(Conv2D, 32 stride (2,2))	$(32, 32, 32)$	32768	$32 * 9 * 3 + 32 = 896$
Couche 2 (dropout :0.25)			
Couche 3 (Conv2D,64)	(32,32,64)	65536	$64 * 32 * 25 + 64 = 51264$
Couche 4 (Poul2D(2,2))	$(16, 16, 64)$	16384	0
Couche 5 (Conv2D,128)	(16,16,128)	32768	$128 * 64 * 9 + 128 = 73856$
Couche 6 (Poul2D(2,2))	$(8, 8, 128)$	8192	0
Couche 7 (Conv2D,256)	(8,8,256)	16384	$256 * 128 * 9 + 256 = 295268$
Couche 8 (dropout :0.25)			
Couche 9 (Dense 256)	$(256, 1)$	256	16384*256+256=4594560
Couche 10 (Dense)	$(10, 1)$	10	2570

2. Comment diminuer le nombre de paramètres ?
   Le plus efficace serait de diminuer la taille de l'image d'entrée, qui influence fortement le nombre de paramètres dans la couche 9, ou bien diminuer le nombre de caractéristiques dans les couches de convolution.

▷ **Exercice 5** La documentation concernant le codage d'un réseau convolutionnel est donnée dans le lien : *https* : *//keras.io/api/layers/convolution*$_{l}$*ayers/convolution*2*d/*.

Dans la librairies keras :

— "same" signifie qu'il y a un padding autour de l'image d'entrée de telle sorte que l'image de sortie ait la même taille que l'entrée.

— "valid" : signifie qu'il n'ya pas de padding i.e. padding de dimension (0,0).

1. Démontrer que le nombre de paramètres dans chaque couche du réseau de la figure 5 où le nombre de classes est 10 (no_classes=10) correspond au résultat donné à la figure 6
   (a) Couche 1 (Conv2D) : $32 * 3 * 3 + 32(biais) = 320$
   (b) Couche 2 (Conv2D) : $64 * 32 * 3 * 3 + 64(biais) = 18496$
   (c) Couche 3 (Conv2D) : $128 * 64 * 3 * 3 + 128(biais) = 73856$
   (d) Couche 3 (Dense) : $22 * 22 * 128 * 256 + 256(biais) = 15859968$
   (e) Couche 4 (Dense) :$256 * 10 + 10(biais) = 2570$
2. Que peut-on faire pour réduire le nombre de paramètres ? La couche *d* comprend le plus grand nombre de paramètres. Une solution serait d'ajouter des couches de pooling entre les couches de convolution pour réduire la taille des images d'entrée des couches cachées et ainsi dimiluer le nombre de paramètres dans la couche d.

```
# Create the model
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape,
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu', padding='valid'))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(no_classes, activation='softmax'))
```

FIGURE 5 – Réseau de neurones convolutionnels

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 22, 22, 128)       73856
_________________________________________________________________
flatten_1 (Flatten)          (None, 61952)             0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               15859968
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570
=================================================================
Total params: 15,955,210
Trainable params: 15,955,210
Non-trainable params: 0
```

Figure 6 – Nombre de paramètres du CNN