Analyse de textures d'images par réseaux de neurones

Adama Abdoul BA et **Christian Manzala Kumbi**
Travail dirigé par le Professeur
Frédéric J.P. RICHARD

Aix-Marseille Université,
Master Mathématique appliquée, Statistiques
Parcours : Data Science

20 avril 2023

Plan

Aix*Marseille université
Initiative d'excellence

1. Introduction

- Analyse de textures d'images
- Réseau neuronal convolutif
- Classification d'images

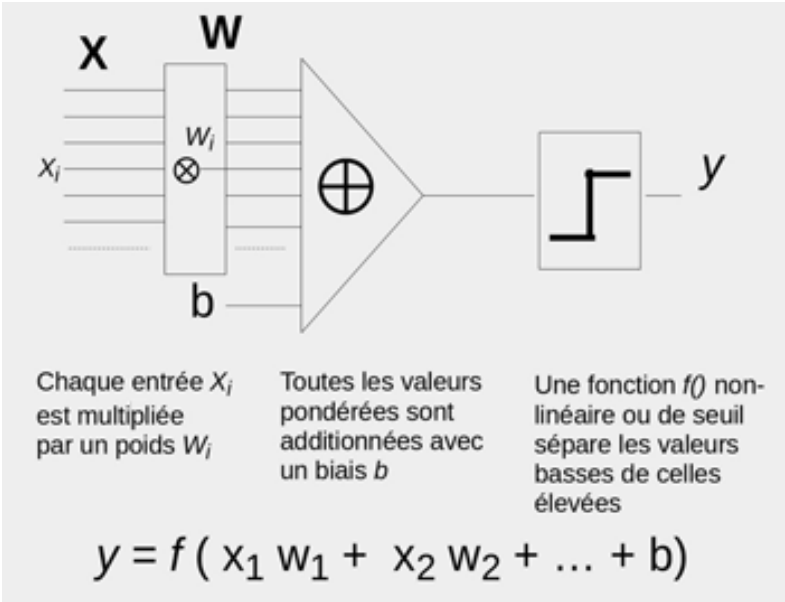
2. Réseau de neurones

Definition

Un réseau de neurones est un assemblage de neurones formels (appelé aussi perceptron). Un neurone formel peut être schématisé par l'equation :

$$Y = f\left(\omega_0 + \sum_{j=1}^{p} \omega_j x^j\right)$$

Représentation graphique

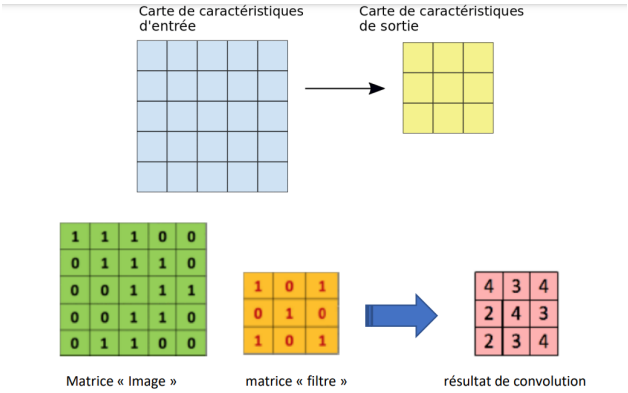


Réseau neuronal convolutif

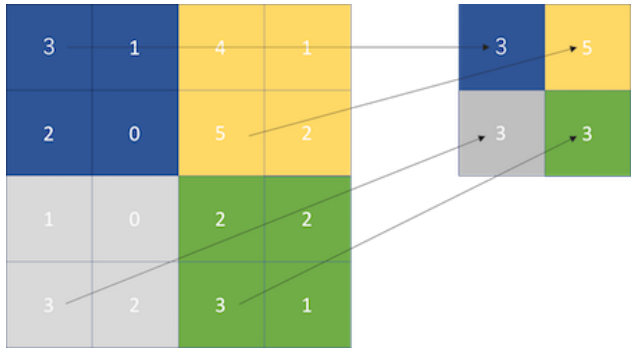
Definition

Un réseau neuronal convolutif (CNN) est une architecture de réseau de neurones specialement conçue pour le traitement d'images.

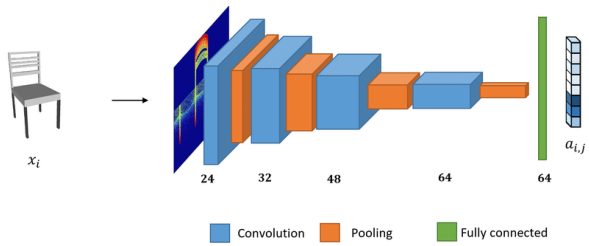
Les caractéristiques du CNN



Convolution



Pooling



CNN

3. Matériels

- 20000 images simulées par le modèle mathématique **Anisotropic Fractional Brownian Fields** (AFBF)
- Les 20000 images et leurs caractéristiques mathématiques sont disponibles sur kaggle sur le lien suivant : https ://www.kaggle.com/datasets/frdricrichard/pyafbf-textures-set-002 ?select=all-images

Les caractéristiques des images

	# Hurst index	argmin set length	argmin set center	Hmax - Hmin
0	0.501390	1.738323	1.010166	0.393828
1	0.303077	1.328428	0.043223	0.167634
2	0.493284	1.387437	0.490528	0.184184
3	0.213095	1.744495	0.906263	0.113027
4	0.050133	3.039050	-0.947203	0.878647

On a consideré comme isoptropiques, les images ayant une variation (Hmax-Hmin) inférieure 0.5 et les images anisotropiques pour une variation supérieure 0.5

Aix*Marseille université
Initiative d'excellence

Les classes d'images

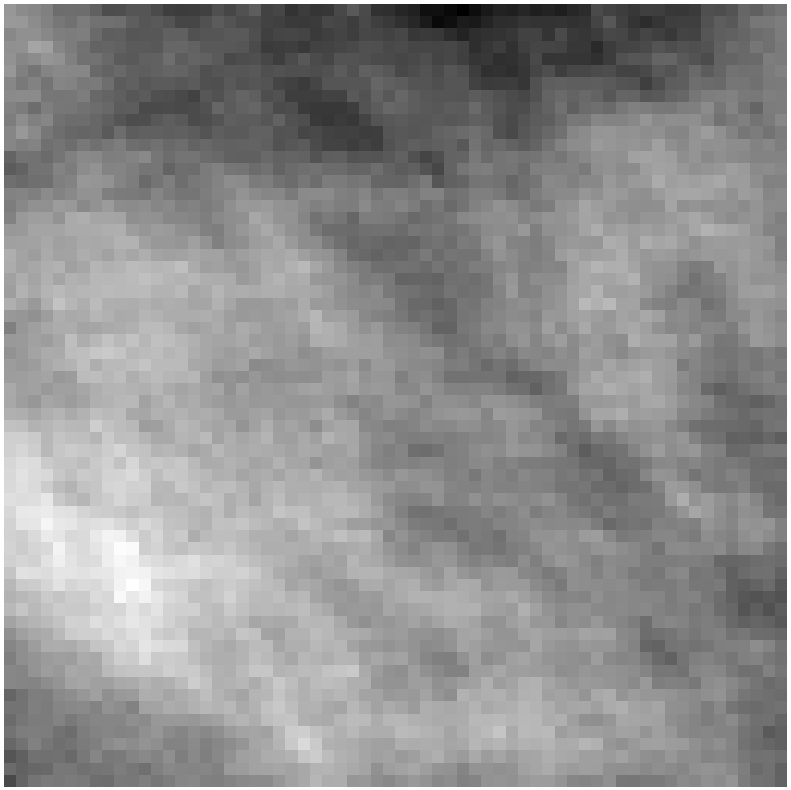


Image isotropique

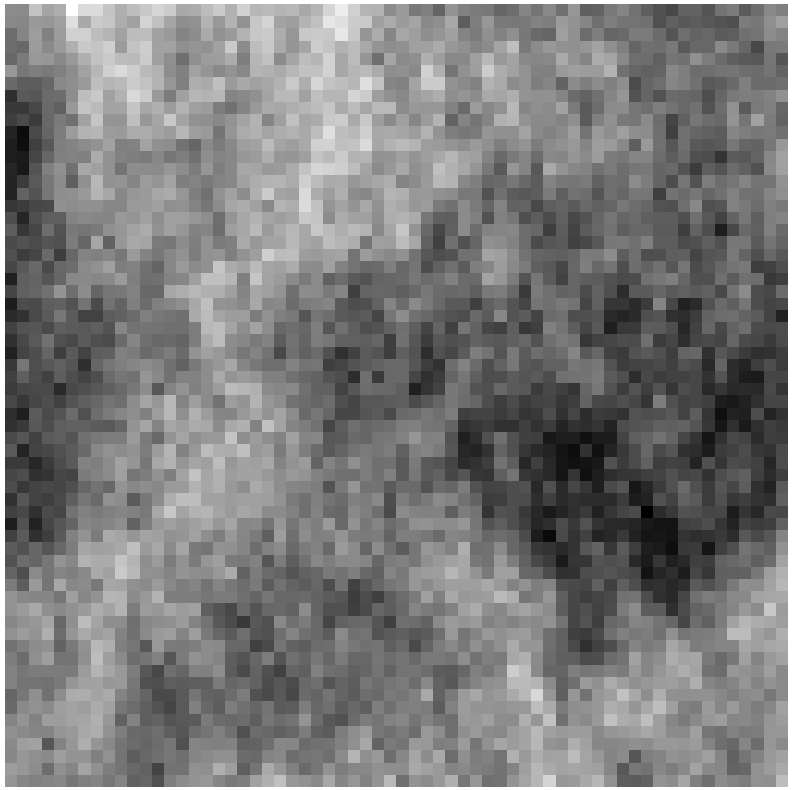


Image anisotropique

- A l'aide de la validation croisée, nous avons fixé 18000 images pour l'entraînement et 2000 images pour la validation

Séparation d'images

```
from sklearn.model_selection import train_test_split #Cross-validation splitting method

dataset = np.array(list(zip(images,classes)), dtype = object)

np.random.shuffle(dataset)

img, labels = zip(*dataset)

t_img,v_img,t_lab,v_lab = train_test_split(img,labels,train_size = 0.9,
                                           random_state = 0)
```

4. Méthodologie

- Dans le cadre de ce travail, nous avons opté pour l'apprentissage par transfert à partir du modèle VGG16

Le modèle **VGG16**

Le modèle **TER_model**

```
TER_model = Sequential()

# Load the pre-trained model from Keras
pretrained_model = VGG16(weights='imagenet',
                         include_top=False,
                         input_shape=(64, 64, 3))

# Freeze the layers of the pre-trained model
for layer in pretrained_model.layers:
    layer.trainable = False


num_classes = 2 # number of classes

TER_model.add(pretrained_model)
TER_model.add(Flatten())
TER_model.add(Dense(128, activation='relu'))
TER_model.add(Dense(128, activation = 'relu'))
TER_model.add(Dense(num_classes, activation='softmax'))
```

Aix*Marseille université
Initiative d'excellence

L'architecture du TER_model

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 2, 2, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 128)               262272
_________________________________________________________________
dense_1 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 258
=================================================================
Total params: 14,993,730
Trainable params: 279,042
Non-trainable params: 14,714,688
_________________________________________________________________
```

Aix*Marseille université
Initiative d'excellence

La fonction d'activation ReLU

Definition

La fonction Rectified Linear Unit (ReLU) permet d'introduire une non-linéarité dans le modèle et d'éviter le phénomène de disparition du gradient qui peut se produire avec d'autres fonctions d'activation.

$$f(x) = \max(0, x) = \begin{cases} 0 & \text{si } x < 0 \\ x & \text{si } x \geq 0 \end{cases} \tag{1}$$

La fonction softmax

Definition

La fonction softmax renvoie un vecteur de probabilités normalisées pour chaque classe. La somme des probabilités de chaque classe est toujours égale à 1.

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{pour } j = 1, \ldots, K \tag{2}$$

Apprentissage

```
# Apprentissage du modèle

TER_model.compile(optimizer=Adam(learning_rate=0.001),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
```

L'optimiseur ADAM

Definition

L'optimiseur Adam pour "Adaptive Moment Estimation" est un algorithme d'optimisation qui combine le RMSprop et la descente de gradient stochastique avec moment. Le fonctionnement de l'optimiseur Adam consiste à calculer et stocker une moyenne mobile des gradients de la fonction de coût.

$$
\begin{aligned}
m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\
v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\
\hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\
\theta_{t+1} &= \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t
\end{aligned}
\tag{3}
$$

L'entropie croisée binaire pour l'erreur

Definition

L'entropie croisée binaire (Binary Cross Entropy) est une mesure de l'erreur utilisée en apprentissage automatique pour la classification binaire.L'entropie croisée binaire mesure la différence entre la distribution de probabilité réelle des classes et la distribution de probabilité prédite par le modèle.

$$BCE = -\frac{1}{N}\sum_{i=1}^{N}(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)) \quad (4)$$

Apprentissage

```
history = TER_model.fit(t_img, t_lab,
          batch_size= 32,
          epochs = 200,
          validation_data =(v_img,v_lab)
                          )
```

Un epoch correspond à un passage à travers toutes les données d'apprentissage. Ici le batchsize = **32** = **18000/562.5**, donc un epoch correspond à un apprentissage sur **562.5** lots d'images.

5. Résultat

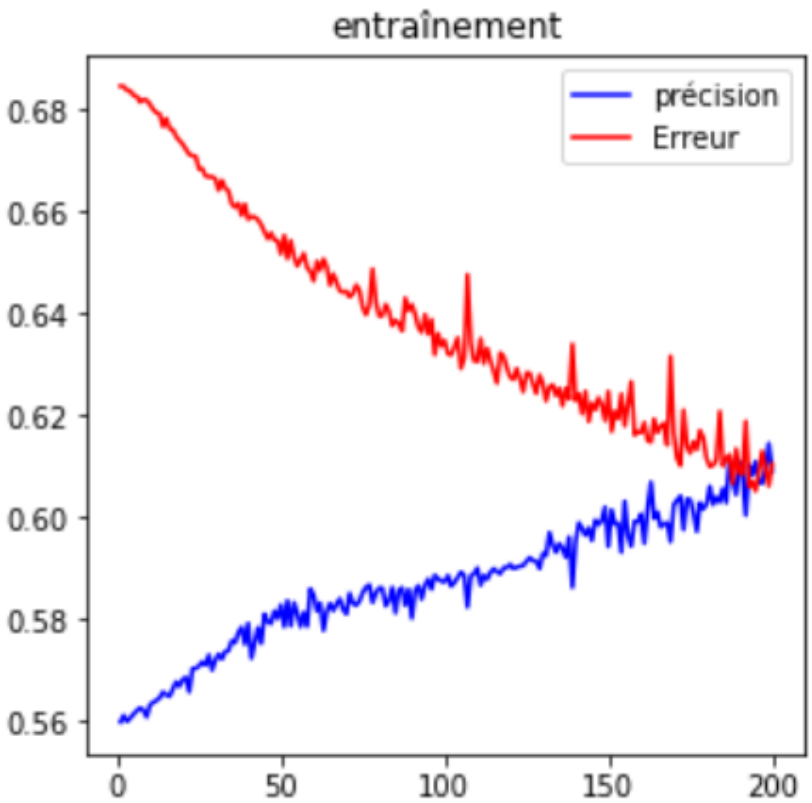
```
Epoch 194/200
563/563 [==============================] - 6s 11ms/step - loss: 0.6070 - accuracy: 0.6083 - val_loss: 1.1736 - val_accuracy: 0.4945
Epoch 195/200
563/563 [==============================] - 6s 11ms/step - loss: 0.6052 - accuracy: 0.6109 - val_loss: 1.1408 - val_accuracy: 0.5135
Epoch 196/200
563/563 [==============================] - 6s 11ms/step - loss: 0.6090 - accuracy: 0.6069 - val_loss: 1.1795 - val_accuracy: 0.5060
Epoch 197/200
563/563 [==============================] - 6s 11ms/step - loss: 0.6130 - accuracy: 0.6067 - val_loss: 1.1556 - val_accuracy: 0.5165
Epoch 198/200
563/563 [==============================] - 7s 12ms/step - loss: 0.6095 - accuracy: 0.6094 - val_loss: 1.0912 - val_accuracy: 0.5155
Epoch 199/200
563/563 [==============================] - 6s 11ms/step - loss: 0.6061 - accuracy: 0.6145 - val_loss: 1.1268 - val_accuracy: 0.5250
Epoch 200/200
563/563 [==============================] - 6s 11ms/step - loss: 0.6105 - accuracy: 0.6091 - val_loss: 1.0979 - val_accuracy: 0.5110
```

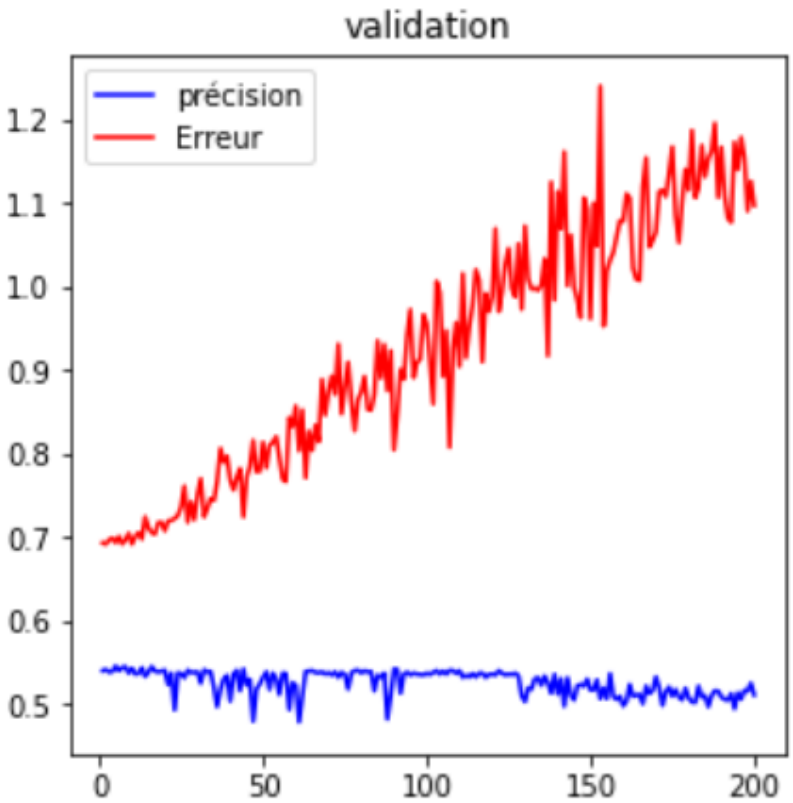
Précisions :

- **60.91%** pour l'entraînement
- **51.10%** pour la validation

Aix*Marseille université
Initiative d'excellence

Evolution de l'apprentissage





6. Discussion

- Un modèle de réseau neuronal de convolution pré-entraîné sur un ensemble d'images similaires aux nôtres pourrait être un choix plus judicieux.
- Augmentation des échantillons