Sous Windows: Installer les modules h5py et numpy avec l'"Invite de commandes" et non directement dans le Notebook!

In [1]:

```
!pip install h5py
import h5py
import numpy as np
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: h5py in /home/uzwolfram/.local/lib/python3.8/site-packages
(3.8.0)
Requirement already satisfied: numpy>=1.14.5 in /home/uzwolfram/.local/lib/python3.8/site
-packages (from h5py) (1.23.5)
```

Chargement des training data et des test data

In [2]:

```
def load_data():
    train_dataset = h5py.File('datasets/trainset.hdf5', "r")
    X_train = np.array(train_dataset["X_train"][:]) # your train set features
    y_train = np.array(train_dataset["Y_train"][:]) # your train set labels

    test_dataset = h5py.File('datasets/testset.hdf5', "r")
    X_test = np.array(test_dataset["X_test"][:]) # your train set features
    y_test = np.array(test_dataset["Y_test"][:]) # your train set labels

    return X_train, y_train, X_test, y_test
X, y, Xtest, yt = load_data()
```

In [3]:

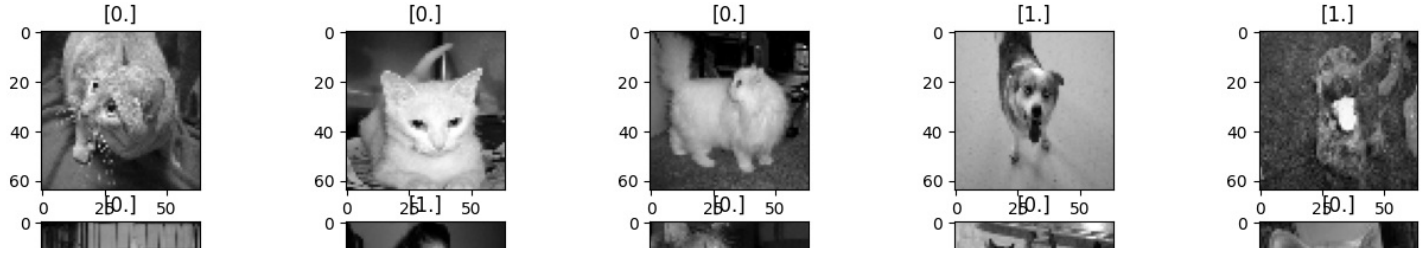
```
print(np.unique(y, return_counts = True)) # 500 photos de chats, 500 photos de chiens
print(X.shape, y.shape) # On a 1000 photos de 64*64 pixels
print(Xtest.shape, yt.shape) # plus 200 photos de test
```

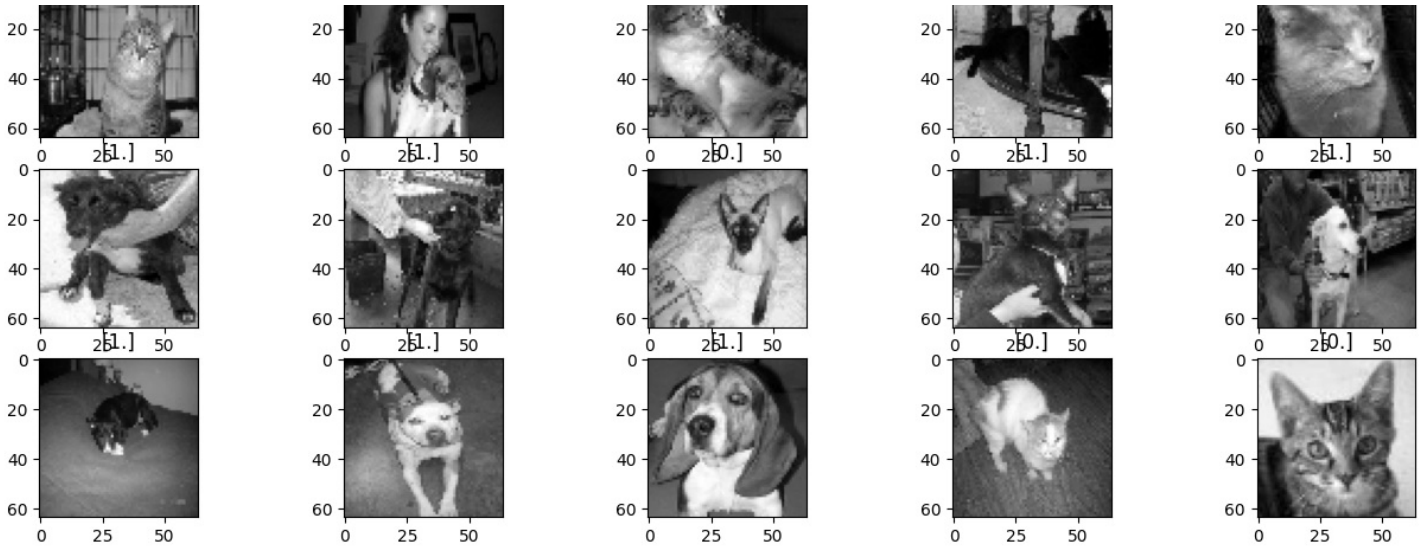
```
(array([0., 1.]), array([500, 500]))
(1000, 64, 64) (1000, 1)
(200, 64, 64) (200, 1)
```

In [4]:

```
# Affichons quelques photos
import matplotlib.pyplot as plt
plt.figure(figsize = (16, 8))
for i in range(1, 21):
    plt.subplot(4, 5, i)
    plt.imshow(X[i], cmap = 'gray')
    plt.title(y[i])
    #plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/matplotlib/text.py:1279: FutureWarning: elementwis
e comparison failed; returning scalar instead, but in the future will perform elementwise
comparison
  if s != self._text:
```





Une fois aplatie, chaque photo est un vecteur de 4096 pixels, donc 4096 variables qui prennent des valeurs entre 0 et 255 (1 pixel = 8 bits). On commence donc par applatir nos images pour en faire un vecteur de 1000 lignes et 4096 caractéristiques.

In [5]:

```
Xr = X.reshape(X.shape[0], X.shape[1] * X.shape[2])
Xt = Xtest.reshape(Xtest.shape[0], Xtest.shape[1] * Xtest.shape[2])
```

In [6]:

```
print(Xr.shape, y.shape)
```

```
(1000, 4096) (1000, 1)
```

Il n'y a plus qu'à entraîner notre modèle avec ces données. On initialise les paramètres du perceptron.

L'initialisation de W est aléatoire.

In [7]:

```
import random
def initParam(n):
    W = np.array([random.uniform(0, 1) for a in range(n)])
    b = random.uniform(0, 1)
    return (W, b)
```

In [8]:

```
# On définit notre modèle
def perceptron(X, W, b):
    Z = np.dot(X, W) + b
    A = 1 / (1 + np.exp(-Z))
    return A
```

In [9]:

```
def produit_scalaire (U, V):
    return np.dot(U.T, V)
```

In [10]:

```
# On évalue la performance du modèle
def cout(A, y):
    return 0.5 * np.mean((A - y)**2)
```

In [11]:

```
W, b = initParam(Xr.shape[1])
```

```
print(W.shape, b)
print(W)
```

```
(4096,) 0.5805906023137344
[0.5992259  0.17766996 0.82225042 ... 0.63876397 0.94240129 0.72296083]
```

In [12]:

```
P = perceptron(Xr[0: 20], W, b)
score = 0
#print(y[0])
for i in range(0,20):
    if(P[i] == y[i]):
        score +=1
print(score/20)
print(P)
```

```
0.5
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

On s'aperçoit que ce ne sont que des 1, il faut donc normaliser nos pixels.

In [13]:

```
print(Xr[0][0])
Xn = Xr/255
Xnt = Xt/255
print(Xn)
```

```
164
[[0.64313725 0.68235294 0.63921569 ... 0.34117647 0.70588235 0.15294118]
 [0.16470588 0.16862745 0.15294118 ... 0.21568627 0.20392157 0.21568627]
 [0.10588235 0.10196078 0.11372549 ... 0.52941176 0.55294118 0.4745098 ]
 ...
 [0.18431373 0.26666667 0.55686275 ... 0.79215686 0.79215686 0.79607843]
 [0.98823529 0.98823529 0.98823529 ... 0.25098039 0.25882353 0.23137255]
 [0.47843137 0.49411765 0.50980392 ... 0.63137255 0.62745098 0.62745098]]
```

In [14]:

```
lot = Xn[:20]
ylot = y[:20]
```

In [15]:

```
P = perceptron(lot, W,b)
print(cout(P, ylot))
```

```
0.25
```

In [16]:

```
# On détermine le gradient, on ne fait qu'un seul pas
def gradient(A, X, y):
    samples, features = X.shape
    dW = - (1 / samples) * produit_scalaire(Xr, y - A)
    db = - (1 / samples) * np.sum((y - A))
    return (dW, db)
```

In [24]:

```
print(cout(P, ylot))
t = gradient(P, lot, y)
P2 = perceptron(lot, t[0], t[1])

print(cout(P2, ylot))
t = gradient(P2, lot, y)
P3 = perceptron(lot, t[0], t[1])
```

```
print(cout(P3, ylot))
t = gradient(P3, lot, y)
P4 = perceptron(lot, t[0], t[1])

print(cout(P4, ylot))
```

```
0.20028826348692533
0.25
```

```
/tmp/ipykernel_6709/2997745446.py:4: RuntimeWarning: overflow encountered in exp
  A = 1 / (1 + np.exp(-Z))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[24], line 6
      3 P2 = perceptron(lot, t[0], t[1])
      5 print(cout(P2, ylot))
----> 6 t = gradient(P2, lot, y)
      7 P3 = perceptron(lot, t[0], t[1])
      9 print(cout(P3, ylot))

Cell In[23], line 4, in gradient(A, X, y)
      2 def gradient(A, X, y):
      3     samples, features = X.shape
----> 4     dW = - (1 / samples) * produit_scalaire(Xr, y - A)
      5     db = - (1 / samples) * np.sum((y - A))
      6     return (dW, db)

ValueError: operands could not be broadcast together with shapes (1000,1) (20,20)
```

In [20]:

```
# on met à jour les paramètres par descente de gradient
def mAj(dW, db, W, b, alpha):
    return (W - alpha * dW,  b - alpha * db)
```

In [41]:

```
# On définit notre fonction d'apprentissage
def learnRN(X, y, alpha = 0.1, iter = 100):
    W, b = initParam(X.shape[1])
    #lots = [X[i*m : (i+1)*m] for i in range((len(X) // m) - 1)]
    #ylots= [y[i*m : (i+1)*m] for i in range((len(y) // m) - 1)]
    couts = []
    for i in range(iter):
        #lot = lots[i % len(lots)]
        #ylot = ylots[i % len(ylots)]
        P = perceptron(X, W, b)
        couts.append(cout(P, y))
        dW, db = gradient(P, X, y)
        W, b = mAj(dW, db, W, b, alpha)
    plt.plot(couts)  # On trace lacourbe de descente des couts
    plt.show()
    return W, b
```

In [45]:

```
W, b = learnRN(Xr, y)
```

```
/tmp/ipykernel_5108/2362300140.py:4: RuntimeWarning: overflow encountered in exp
  A = 1 / (1 + np.exp(-Z))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[45], line 1
----> 1 W, b = learnRN(Xr, y)

Cell In[41], line 13, in learnRN(X, y, alpha, iter)
     11     couts.append(cout(P, y))
     12     dW, db = gradient(P, X, y)
---> 13     W, b = mAj(dW, db, W, b, alpha)
     14 plt.plot(couts)  # On trace lacourbe de descente des couts
```

```
     15 plt.show()

Cell In[43], line 3, in mAj(dW, db, W, b, alpha)
      2 def mAj(dW, db, W, b, alpha):
----> 3     return (W - alpha * dW,  b - alpha * db)

ValueError: operands could not be broadcast together with shapes (4096,) (4096,1000)
```

In [23]:

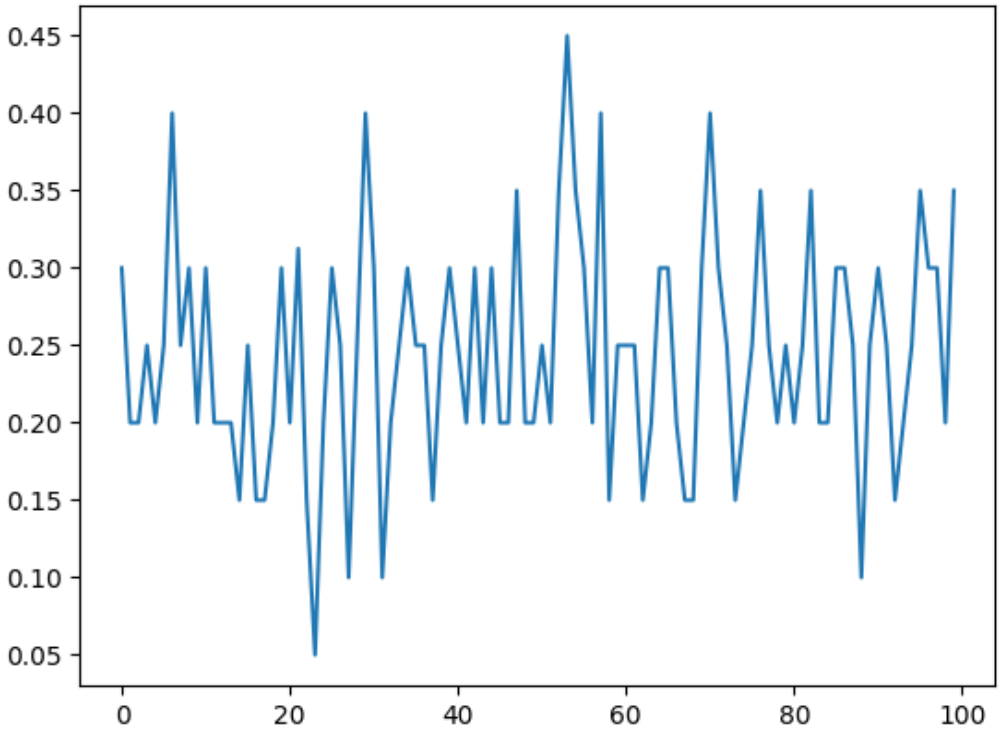
```
print(W, b)
```

```
[ 0.50830662 -0.49132795  0.51038073 ... -0.49241645  0.50796948
 -0.49305831] [0.52385411]
```

In [30]:

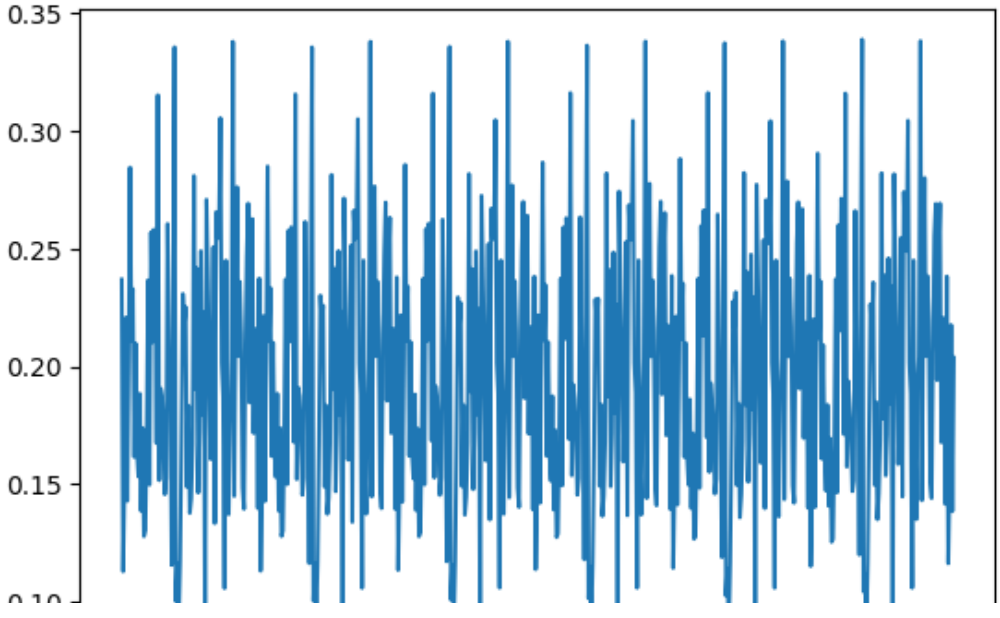
```
W, b = learnRN(Xr, y, alpha = 0.01)
```

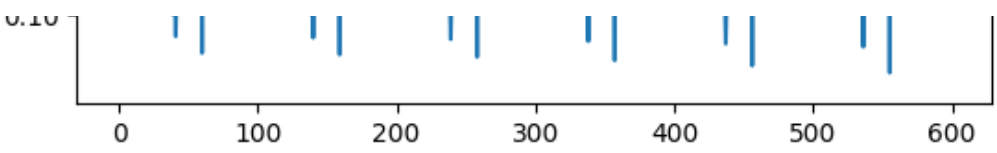
```
/tmp/ipykernel_5108/2362300140.py:4: RuntimeWarning: overflow encountered in exp
  A = 1 / (1 + np.exp(-Z))
```



In [31]:

```
# Voilà qui est mieux
W, b = learnRN(Xn, y, alpha = 0.001, iter = 600)
```





In [28]:

```
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings('ignore')
```

In [29]:

```
# Nous y voilà! Quelle est alors la performance du modèle?
print(W, b)
# [ 0.50663628 -0.49281757  0.50713973 ... -0.49181645  0.50860527  -0.49209327] [0.5195
226]
```

```
[ 0.5 -0.5  0.5 ... -0.5  0.5 -0.5] 0.5
```

In [33]:

```
def predict(X, W, b):
    return perceptron(X, W, b)
```

In [30]:

```
# On modifie la fonction d'apprentissage pour afficher les courbes de coûts et les perfor
mances du modèle
def learnRN(X, y, alpha = 0.1, iter = 100):
    W, b = initParam(X.shape[1])
    couts = []
    accuracy = []
    for i in range(iter):
        A = predict(X, W, b)
        couts.append(cout(A, y))
        accuracy.append(accuracy_score(y, A.round()))
        dW, db = gradient(P, X, y)
        W, b = mAj(dW, db, W, b, alpha)

    plt.figure(figsize = (12, 4))
    plt.subplot(1, 2, 1)
    plt.plot(Loss) # On trace la courbe de descente des couts
    plt.subplot(1, 2, 2)
    plt.plot(Acc) # et l'accuracy
    plt.show()
    return W, b
```

In [34]:

```
W, b = learnRN(Xn, y, alpha = 0.01, iter = 1000)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[34], line 1
----> 1 W, b = learnRN(Xn, y, alpha = 0.01, iter = 1000)

Cell In[30], line 12, in learnRN(X, y, alpha, iter)
     10     accuracy.append(accuracy_score(y, A.round()))
     11     dW, db = gradient(P, X, y)
---> 12     W, b = mAj(dW, db, W, b, alpha)
     14 plt.figure(figsize = (12, 4))
     15 plt.subplot(1, 2, 1)

Cell In[20], line 3, in mAj(dW, db, W, b, alpha)
      2 def mAj(dW, db, W, b, alpha):
----> 3     return (W - alpha * dW,  b - alpha * db)

ValueError: operands could not be broadcast together with shapes (4096,) (4096,20)
```

On serait tenté d'aller plus loin car l'accuracy continue d'augmenter...

In [ ]:

```
W, b = learnRN(Xn, y, alpha = 0.01, iter = 10000)
```

Ne serait-on pas en overfitting? Il est temps de regarder ce qui se passe sur les datas de test:

In [ ]:

```
def learnRN(X, y, Xt, yt, alpha = 0.1, iter = 100): # Cette fois, on ajoute les données
de test
    couts = []
    accuracy = []
    couts_T = []
    accuracy_T = []
    for i in range(iter):
        A = predict(X, W, b)
        couts.append(cout(A, y))
        accuracy.append(accuracy_score(y, A.round()))
        At = predict(Xt, W, b)
        couts_T.append(cout(At, yt))
        accuracy_T.append(accuracy_score(yt, At.round()))

        dW, db = gradient(P, X, y)
        W, b = mAj(dW, db, W, b, alpha)

    plt.figure(figsize = (12, 4))
    plt.subplot(1, 2, 1)
    plt.plot (Loss, label = 'Données d`entraînement')  # On trace la courbe de descente d
es couts
    plt.plot (LossT, label = 'Données de test')
    plt.legend()
    plt.subplot(1, 2, 2)
    plt.plot(Acc,label = 'Données d`entraînement')
    plt.plot(AccT,label = 'Données de test')
    plt.legend()
    plt.show()
    return W, b
```

In [ ]:

```
W, b = learnRN(Xn, y, Xtn, yt, alpha = 0.01, iter = 4000)
```

On n'a pas assez de données pour le nombre de caractéristiques. C'est le Fléau de la dimension. Il faut réduire le nombre de variables ou augmenter le nombre de données. Mais surtout... augmenter le nombre de neurones! Nous avons donc besoin d'un réseau de neurones multicouche.

A l'aide d'un PMC

In [83]:

```
from sklearn.model_selection import train_test_split
Xa, ya, Xt, yt = load_data() # chargement des training data et des test data
```

In [84]:

```
Xa = Xa.reshape(Xa.shape[0], Xa.shape[1] * Xa.shape[2]) # On applatit et on normalise le
s données
Xt = Xt.reshape(Xt.shape[0], Xt.shape[1] * Xt.shape[2])
```

In [85]:

```
print(Xa.shape, ya.shape)
```

```
print(Xt.shape, yt.shape)
```

```
(1000, 4096) (1000, 1)
(200, 4096) (200, 1)
```

In [86]:

```
Xa = Xa / 255
Xt = Xt / 255
```

On commence avec les paramètres par défaut

apprentissage avec les deux dernières variables

In [87]:

```
!pip install scikit-learn
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
```

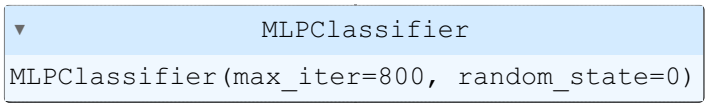
```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: scikit-learn in /home/uzwolfram/.local/lib/python3.8/site-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in /home/uzwolfram/.local/lib/python3.8/site-packages (from scikit-learn) (1.23.5)
Requirement already satisfied: joblib>=1.1.1 in /home/uzwolfram/.local/lib/python3.8/site-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: scipy>=1.3.2 in /home/uzwolfram/.local/lib/python3.8/site-packages (from scikit-learn) (1.10.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /home/uzwolfram/.local/lib/python3.8/site-packages (from scikit-learn) (3.1.0)
```

In [88]:

```
clf = MLPClassifier(max_iter = 800, random_state = 0)
clf.fit(Xa, ya)
```

```
/home/uzwolfram/.local/lib/python3.8/site-packages/sklearn/neural_network/_multilayer_perceptron.py:1098: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[88]:



On calcule les scores d'apprentissage et de test

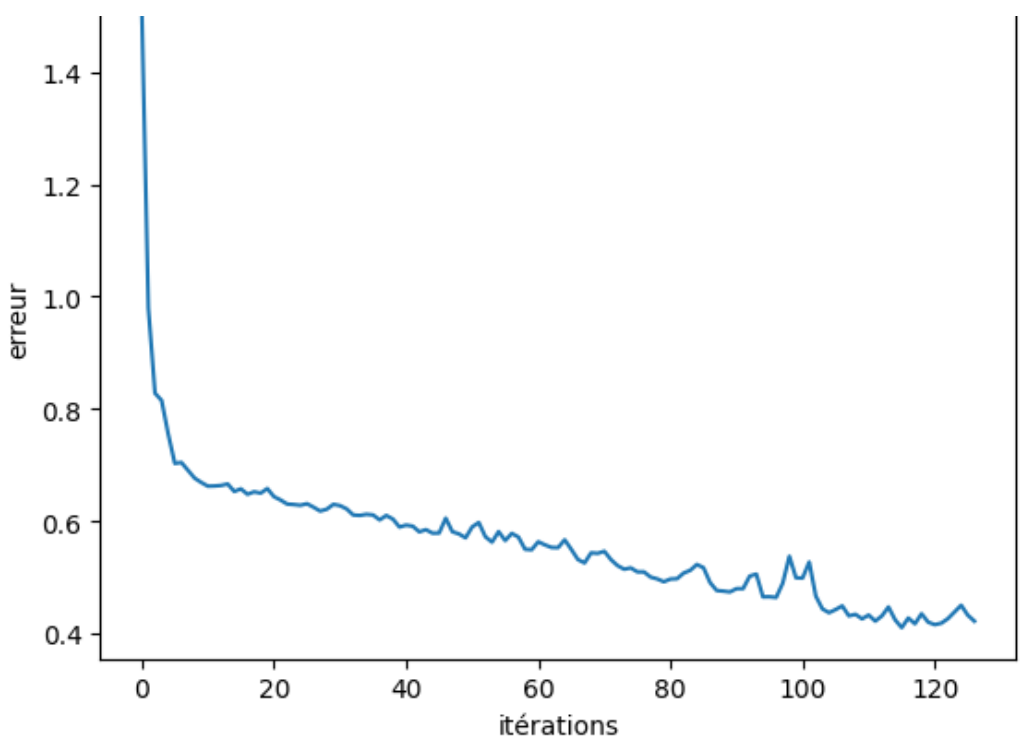
In [89]:

```
train_score = clf.score(Xa, Ya)
test_score = clf.score(Xt, Yt)
print("Le score sur les données d'apprentissage est {}".format(train_score))
print("Le score sur les données de test est {}".format(test_score))
```

```
Le score sur les données d'apprentissage est 0.781
Le score sur les données de test est 0.535
```

In [90]:

```
import matplotlib.pyplot as plt
plt.figure()
plt.xlabel('itérations')
plt.ylabel('erreur')
plt.plot(clf.loss_curve_)
plt.show()
```



1 chance sur 2, vraiment pas terrible! Et ce n'est pas un problème de nombre d'itérations. Regardons comment cela varie avec le nombre de neurones dans une couche intermédiaire

In [55]:

```
(800, 64, 64) (800, 1)
(200, 64, 64) (200, 1)
(800, 4096)
(200, 4096)
```

Et avec le nombre de couches (10 neurones dans chaque couche). D'abord aplatir Xa2:

In [96]:

```
clf2 = MLPClassifier(hidden_layer_sizes = (10, 10, 2), activation = 'relu', solver = 'ad
am',
                     max_iter = 5000, random_state = 100)
clf2.fit(Xa, ya)
train_score_2 = clf2.score(Xa, ya)
test_score_2 = clf2.score(Xt, yt)
print("Le score sur les données d'apprentissage est {}".format(train_score_2))
print("Le score sur les données de test est {}".format(test_score_2))
```

```
/home/uzwolfram/.local/lib/python3.8/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:1098: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
Le score sur les données d'apprentissage est 0.5
Le score sur les données de test est 0.5
```

In [97]:

```
# On a du mal à dépasser 60% sur les données de test, c'est notre challenge.
clf3 = MLPClassifier(hidden_layer_sizes = (10, 20, 20, 20, 20, 2), activation = 'relu',
solver = 'adam',
                     max_iter = 5000, random_state = 50)
clf3.fit(Xa, ya)
train_score_3 = clf3.score(Xa, ya)
test_score_3 = clf3.score(Xt, yt)
print("Le score sur les données d'apprentissage est {}".format(train_score_3))
print("Le score sur les données de test est {}".format(test_score_3))
```

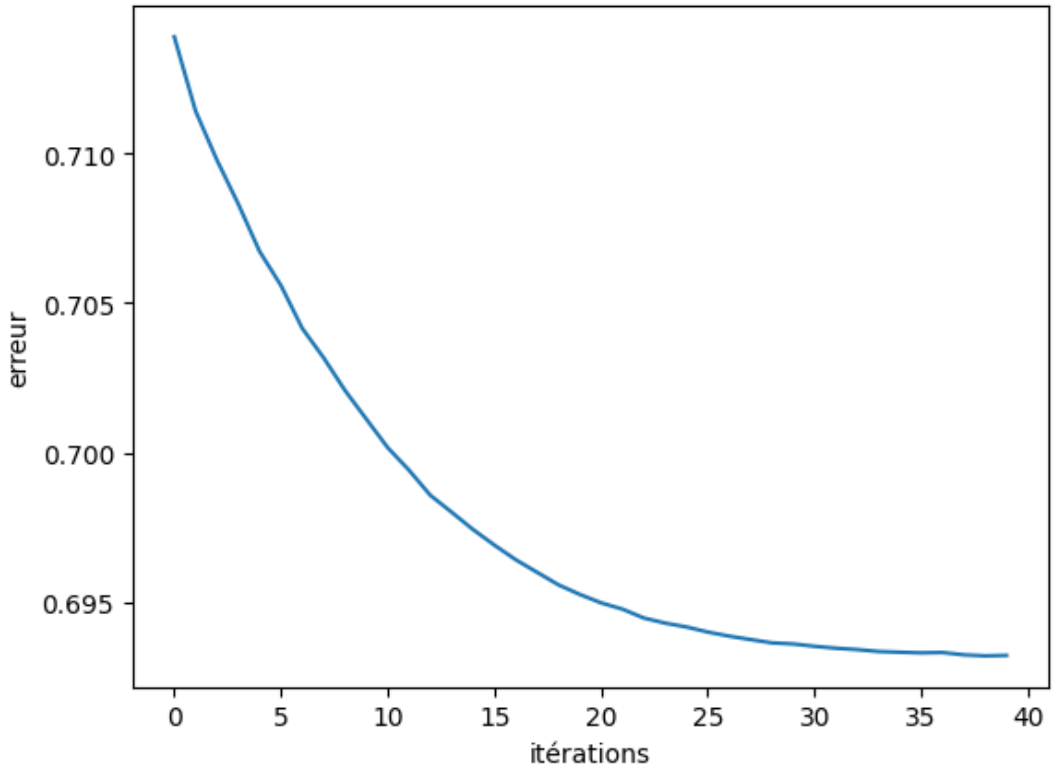
```
/home/uzwolfram/.local/lib/python3.8/site-packages/sklearn/neural_network/_multilayer_per
```

```
ceptron.py:1098: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
Le score sur les données d'apprentissage est 0.5
Le score sur les données de test est 0.5
```

In [95]:

```
# C'est pas trop mal!
plt.figure()
plt.xlabel('itérations')
plt.ylabel('erreur')
plt.plot(clf2.loss_curve_)
plt.show()
```



In [ ]:

```
# Essayons avec une activation sigmoïde.
```

Difficile d'avoir mieux que 60% Il faudrait augmenter le nombre d'itérations mais c'est long... à moins de commencer par réduire le nombres de variables... à suivre.

In [ ]:

```

```

In [ ]:

```
plt.figure()
plt.xlabel('itérations')
plt.ylabel('erreur')
plt.plot(clf.loss_curve_)
plt.show()
```

Réseaux de neurones convolutifs avec Keras

On importe les bibliothèques nécessaires puis on réinitialise le dataset

In [1]:

```
!pip install h5py
```

```
import h5py
!pip install tensorflow
import tensorflow as tf
from tensorflow import keras
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: tensorflow in /home/uzwolfram/.local/lib/python3.8/site-pa
ckages (2.12.0)
Requirement already satisfied: absl-py>=1.0.0 in /home/uzwolfram/.local/lib/python3.8/sit
e-packages (from tensorflow) (1.4.0)
Requirement already satisfied: numpy<1.24,>=1.22 in /home/uzwolfram/.local/lib/python3.8/
site-packages (from tensorflow) (1.23.5)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.2
1.5,<5.0.0dev,>=3.20.3 in /home/uzwolfram/.local/lib/python3.8/site-packages (from tensor
flow) (4.22.3)
Requirement already satisfied: termcolor>=1.1.0 in /home/uzwolfram/.local/lib/python3.8/s
ite-packages (from tensorflow) (2.2.0)
Requirement already satisfied: libclang>=13.0.0 in /home/uzwolfram/.local/lib/python3.8/s
ite-packages (from tensorflow) (16.0.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /home/uzwolfram/.local/lib/pyt
hon3.8/site-packages (from tensorflow) (4.5.0)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /home/uzwolfram/.local/lib/python3.
8/site-packages (from tensorflow) (0.4.0)
Requirement already satisfied: tensorflow-estimator<2.13,>=2.12.0 in /home/uzwolfram/.loc
al/lib/python3.8/site-packages (from tensorflow) (2.12.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /home/uzwolfram/.local/lib/python3.
8/site-packages (from tensorflow) (1.53.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /home/uzwolfram/.local/lib/python3.8/
site-packages (from tensorflow) (3.3.0)
Requirement already satisfied: tensorboard<2.13,>=2.12 in /home/uzwolfram/.local/lib/pyth
on3.8/site-packages (from tensorflow) (2.12.2)
Requirement already satisfied: six>=1.12.0 in /usr/lib/python3/dist-packages (from tensor
flow) (1.14.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/lib/python3/dist-packages (fro
m tensorflow) (1.11.2)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /home/uzwolfram/.l
ocal/lib/python3.8/site-packages (from tensorflow) (0.32.0)
Requirement already satisfied: astunparse>=1.6.0 in /home/uzwolfram/.local/lib/python3.8/
site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: jax>=0.3.15 in /home/uzwolfram/.local/lib/python3.8/site-p
ackages (from tensorflow) (0.4.8)
Requirement already satisfied: h5py>=2.9.0 in /home/uzwolfram/.local/lib/python3.8/site-p
ackages (from tensorflow) (3.8.0)
Requirement already satisfied: packaging in /usr/lib/python3/dist-packages (from tensorfl
ow) (20.3)
Requirement already satisfied: flatbuffers>=2.0 in /home/uzwolfram/.local/lib/python3.8/s
ite-packages (from tensorflow) (23.3.3)
Requirement already satisfied: google-pasta>=0.1.1 in /home/uzwolfram/.local/lib/python3.
8/site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: setuptools in /usr/lib/python3/dist-packages (from tensorf
low) (45.2.0)
Requirement already satisfied: keras<2.13,>=2.12.0 in /home/uzwolfram/.local/lib/python3.
8/site-packages (from tensorflow) (2.12.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/lib/python3/dist-packages (from
astunparse>=1.6.0->tensorflow) (0.34.2)
Requirement already satisfied: ml-dtypes>=0.0.3 in /home/uzwolfram/.local/lib/python3.8/s
ite-packages (from jax>=0.3.15->tensorflow) (0.1.0)
Requirement already satisfied: scipy>=1.7 in /home/uzwolfram/.local/lib/python3.8/site-pa
ckages (from jax>=0.3.15->tensorflow) (1.10.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in /home/uzwolfram/.local/lib/python
3.8/site-packages (from tensorboard<2.13,>=2.12->tensorflow) (2.17.3)
Requirement already satisfied: werkzeug>=1.0.1 in /home/uzwolfram/.local/lib/python3.8/si
te-packages (from tensorboard<2.13,>=2.12->tensorflow) (2.2.3)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /home/uzwolfram/.
local/lib/python3.8/site-packages (from tensorboard<2.13,>=2.12->tensorflow) (0.7.0)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/lib/python3/dist-packages (fro
m tensorboard<2.13,>=2.12->tensorflow) (2.22.0)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /home/uzwolfram/.local/li
b/python3.8/site-packages (from tensorboard<2.13,>=2.12->tensorflow) (1.8.1)
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in /home/uzwolfram/.local/l
ib/python3.8/site-packages (from tensorboard<2.13,>=2.12->tensorflow) (1.0.0)
Requirement already satisfied: markdown>=2.6.8 in /home/uzwolfram/.local/lib/python3.8/si
```

```
te-packages (from tensorboard<2.13,>=2.12->tensorflow) (3.4.3)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /home/uzwolfram/.local/lib/pytho
n3.8/site-packages (from google-auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow) (5.3
.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /home/uzwolfram/.local/lib/python3.8/site
-packages (from google-auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow) (4.9)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /home/uzwolfram/.local/lib/python
3.8/site-packages (from google-auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow) (0.2.
8)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /home/uzwolfram/.local/lib/pyt
hon3.8/site-packages (from google-auth-oauthlib<1.1,>=0.5->tensorboard<2.13,>=2.12->tenso
rflow) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in /home/uzwolfram/.local/lib/pyth
on3.8/site-packages (from markdown>=2.6.8->tensorboard<2.13,>=2.12->tensorflow) (6.3.0)
Requirement already satisfied: MarkupSafe>=2.1.1 in /home/uzwolfram/.local/lib/python3.8/
site-packages (from werkzeug>=1.0.1->tensorboard<2.13,>=2.12->tensorflow) (2.1.2)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.8/dist-packages (from
importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.13,>=2.12->tensorflow) (3.14.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /home/uzwolfram/.local/lib/python3
.8/site-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.13,>=2
.12->tensorflow) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/lib/python3/dist-packages (from re
quests-oauthlib>=0.7.0->google-auth-oauthlib<1.1,>=0.5->tensorboard<2.13,>=2.12->tensorfl
ow) (3.1.0)
```

```
2023-04-16 15:59:49.746427: I tensorflow/tsl/cuda/cudart_stub.cc:28] Could not find cuda
drivers on your machine, GPU will not be used.
2023-04-16 15:59:50.873555: I tensorflow/tsl/cuda/cudart_stub.cc:28] Could not find cuda
drivers on your machine, GPU will not be used.
2023-04-16 15:59:50.883591: I tensorflow/core/platform/cpu_feature_guard.cc:182] This Ten
sorFlow binary is optimized to use available CPU instructions in performance-critical ope
rations.
To enable the following instructions: AVX2, in other operations, rebuild TensorFlow with
the appropriate compiler flags.
2023-04-16 15:59:56.064581: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TR
T Warning: Could not find TensorRT
```

In [6]:

```
class generator:
    def __init__(self, file):
        self.file = file

    def __call__(self):
        with h5py.File(self.file, 'r') as hf:
            for im in hf["train_img"]:
                yield im
```

In [11]:

```
hdf5_path = '/home/uzwolfram/Documents/L3_IntelligenceArtificielle/datasets'

ds = tf.data.Dataset.from_generator(
    generator(hdf5_path),
    tf.uint8,
    tf.TensorShape([427, 561, 3]))
```

In [14]:

```
def read_examples_hdf5(filename, label):
    with h5py.File(filename, 'r') as hf: # read frames from HDF5 and decode them from JP
G
        return frames, label

filenames = glob.glob(os.path.join(hdf5_path, "*.h5"))
labels = [0] * len(filenames) # ... can we do this more elegantly?

dataset = tf.data.Dataset.from_tensor_slices((filenames, labels))
dataset = dataset.map(
    lambda filename, label: tuple(tf.py_func(
        read_examples_hdf5, [filename, label], [tf.uint8, tf.int64])))
```

```
)

dataset = dataset.shuffle(1000 + 3 * BATCH_SIZE)
dataset = dataset.batch(BATCH_SIZE)
iterator = dataset.make_one_shot_iterator()
next_batch = iterator.get_next()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[14], line 5
      2     with h5py.File(filename, 'r') as hf: # read frames from HDF5 and decode them
from JPG
      3         return frames, label
----> 5 filenames = glob.glob(os.path.join(hdf5_path, "*.h5"))
      6 labels = [0] * len(filenames) # ... can we do this more elegantly?
      8 dataset = tf.data.Dataset.from_tensor_slices((filenames, labels))

NameError: name 'glob' is not defined
```

In [ ]:

```
value = ds.make_one_shot_iterator().get_next()
while True: # Example on how to read elements
    try:
        data = sess.run(value)
        print(data.shape)
    except tf.errors.OutOfRangeError:
        print('done.')
        break
```

In [12]:

```
ds = tf.data.Dataset.from_tensor_slices(filenames)
# You might want to shuffle() the filenames here depending on the application
ds = ds.interleave(lambda filename: tf.data.Dataset.from_generator(
        generator(),
        tf.uint8,
        tf.TensorShape([427, 561, 3]),
        args = (filename,)),
       cycle_length, block_length)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[12], line 1
----> 1 ds = tf.data.Dataset.from_tensor_slices(filenames)
      2 # You might want to shuffle() the filenames here depending on the application
      3 ds = ds.interleave(lambda filename: tf.data.Dataset.from_generator(
      4         generator(),
      5         tf.uint8,
      6         tf.TensorShape([427,561,3]),
      7         args=(filename,)),
      8        cycle_length, block_length)

NameError: name 'filenames' is not defined
```

In [5]:

```
#xa, ya, xt, yt = load_data() # chargement des training data et des test data

(xa, ya) = keras.utils.image_dataset_from_directory(
    directory = '/home/uzwolfram/Documents/L3_IntelligenceArtificielle/datasets',
    labels = 'inferred',
    label_mode = 'categorical',
    batch_size = 800,
    image_size = (64, 64))
(xt, yt) = keras.utils.image_dataset_from_directory(
    directory = '/home/uzwolfram/Documents/L3_IntelligenceArtificielle/datasets',
    labels = 'inferred',
    label_mode = 'categorical',
    batch_size = 200,
    image_size = (64, 64))
```

```
xa = xa.reshape(-1, 28, 28, 1)
xt = xt.reshape(-1, 28, 28, 1)
print("Données d'apprentissage: ", xa.shape)
print("Données de test: ", xt.shape)
```

```
Found 0 files belonging to 0 classes.

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[5], line 3
      1 #xa, ya, xt, yt = load_data() # chargement des training data et des test data
----> 3 (xa, ya) = keras.utils.image_dataset_from_directory(
      4     directory = '/home/uzwolfram/Documents/L3_IntelligenceArtificielle/datasets',
      5     labels = 'inferred',
      6     label_mode = 'categorical',
      7     batch_size = 800,
      8     image_size = (64, 64))
      9 (xt, yt) = keras.utils.image_dataset_from_directory(
     10     directory = '/home/uzwolfram/Documents/L3_IntelligenceArtificielle/datasets',
     11     labels = 'inferred',
     12     label_mode = 'categorical',
     13     batch_size = 200,
     14     image_size = (64, 64))
     15 xa = xa.reshape(-1, 28, 28, 1)

File ~/.local/lib/python3.8/site-packages/keras/utils/image_dataset.py:297, in image_data
set_from_directory(directory, labels, label_mode, class_names, color_mode, batch_size, im
age_size, shuffle, seed, validation_split, subset, interpolation, follow_links, crop_to_a
spect_ratio, **kwargs)
    293 image_paths, labels = dataset_utils.get_training_or_validation_split(
    294     image_paths, labels, validation_split, subset
    295 )
    296 if not image_paths:
--> 297     raise ValueError(
    298         f"No images found in directory {directory}. "
    299         f"Allowed formats: {ALLOWLIST_FORMATS}"
    300     )
    302 dataset = paths_and_labels_to_dataset(
    303     image_paths=image_paths,
    304     image_size=image_size,
   (...)
    310     crop_to_aspect_ratio=crop_to_aspect_ratio,
    311 )
    312 dataset = dataset.prefetch(tf.data.AUTOTUNE)

ValueError: No images found in directory /home/uzwolfram/Documents/L3_IntelligenceArtific
ielle/datasets. Allowed formats: ('.bmp', '.gif', '.jpeg', '.jpg', '.png')
```

In [ ]:

```
# On normalise les données
xa = xa / xa.max()
xt = xt / xa.max()
```

In [ ]:

```
model = keras.models.Sequential()
model.add(keras.layers.Input((28, 28, 1))) # Couche d'entrée 28*28 pixels monochrome (le
s bords seront coupés)

model.add(keras.layers.Conv2D(8, (3, 3), activation = 'relu')) # Couche de convolution 2
D (noir et blanc)
model.add(keras.layers.MaxPooling2D((2, 2))) # Couche de compression
model.add(keras.layers.Dropout(0.2)) # On demande à 20% tirés au sort des neurones de ne
pas participer à la suite des calculs

model.add(keras.layers.Conv2D(16, (3, 3), activation = 'relu'))
model.add(keras.layers.MaxPooling2D((2, 2)))
model.add(keras.layers.Dropout(0.2))

model.add(keras.layers.Flatten()) # Couche d'aplatissement
model.add(keras.layers.Dense(100, activation = 'relu')) # Couche dense
```

```
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(10, activation = 'softmax'))
```

In [ ]:

```
model.summary()
```

In [ ]:

```
# On le crée
model.compile(optimizer = 'adam', loss = 'sparse_categorical_crossentropy', metrics = ['
accuracy'])
```

In [ ]:

```
# On paramètre et on lance l'apprentissage
batch_size = 512
epochs =  16
verb = 1
history = model.fit(xa, ya, batch_size = batch_size, epochs = epochs, verbose = verb, va
lidation_data = (xt, yt))
```

In [ ]:

```
history.history # Noter bien le contenu de l'historique
```

In [ ]:

```
# Tracé des courbes d'apprentissages sur les données d'apprentissage et de test
hist = history.history
a = hist['accuracy']
b = hist['val_accuracy']
fig, ax = plt.subplots()
ax.plot(a, label = 'Données de test', color = 'blue')
ax.plot(b, label = 'Données d''apprentissage', color = 'red')
plt.xlabel('Epoch')
plt.ylabel('Score')
plt.title('Courbes d''apprentissage')
plt.show()
```