Rapport LABO4 Learning with Artificial Neural Networks

ARN

Auteur:

Bastien Pillonel

Loïc Brasey

Question 1:

What is the learning algorithm being used to optimize the weights of the neural networks? What are the parameters (arguments) being used by that algorithm? What cost function is being used ? please, give the equation(s)

L'algorithm utilisé est le RMSprop de la librairie Keras.

Les paramètres utilisés sont :

```
tf.keras.optimizers.RMSprop(
    learning_rate=0.001,
    rho=0.9,
    momentum=0.0,
    epsilon=1e-07,
    centered=False,
    weight_decay=None,
    clipnorm=None,
    clipvalue=None,
    global_clipnorm=None,
    use_ema=False,
    ema_momentum=0.99,
    ema_overwrite_frequency=100,
    jit_compile=True,
    name="RMSprop",
    **kwargs
)
```

L'equation utilisé est :

$v_{dw} = \beta * v_{dw} + (1 - \beta) * dw^2$
$v_{db} = \beta * v_{dw} + (1 - \beta) * db^2$
$W = W - a * \frac{dw}{\sqrt{v_{dw}} + \epsilon}$
$b = b - a * \frac{db}{\sqrt{v_{db}} + \epsilon}$

Selon article de site web towardsdatascience

Et la loss function utilisé est la categorical_crossentropy.

Question 2

Model complexity: for each experiment (shallow network learning from raw data, shallow network learning from features, CNN, and Fashion MNIST), select a neural network topology and describe the inputs, indicate how many are they, and how many outputs. Compute the number of weights of each model (e.g., how many weights between the input and the hidden layer, how many weights between each pair of layers, biases, etc..) and explain how do you get to the total number of weights.

Shallow network learning from raw data

Nbr d'entrés : 784
Nbr de sorties : 10 classes
Nbr de couches cachée : 1
Nbr de neurones cachés : 300
Nbr de poid dans couche caché : 784 * 300 + 300 = 235 500 Nbr de poid à la sortie : 300 * 10 + 10 = 3010 Nbr de poid total : 235 500 + 3010 = 238 510

Shallow network learning from features

CNN

Fashion MNIST

Question 3

Do the deep neural networks have much more "capacity" (i.e., do they have more weights?) than the shallow ones? explain with one example

Question 4

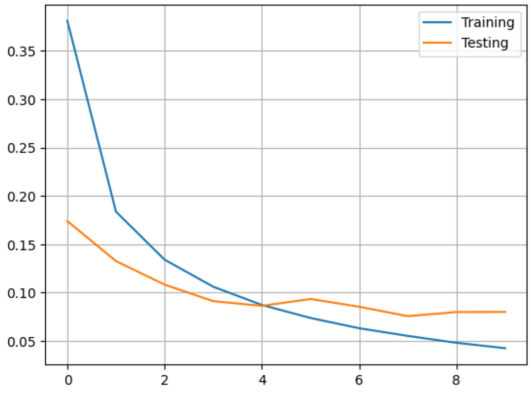
Test every notebook for at least three different meaningful cases (e.g., for the MLP exploiting raw data, test different models varying the number of hidden neurons, for the feature-based model, test pix_p_cell 4 and 7, and number of orientations or number of hidden neurons, for the CNN, try different number of neurons in the feed-forward part) describe the model and present the performance of the system (e.g., plot of the evolution of the error, final evaluation scores and confusion matrices). Comment the differences in results. Are there particular digits that are frequently confused?

Shallow network learning from features

Pour ce notebook nous avons principalement essayé de faire varier le nombre de neurones dans la couches cachée.

Résultat avec 100 neurones:

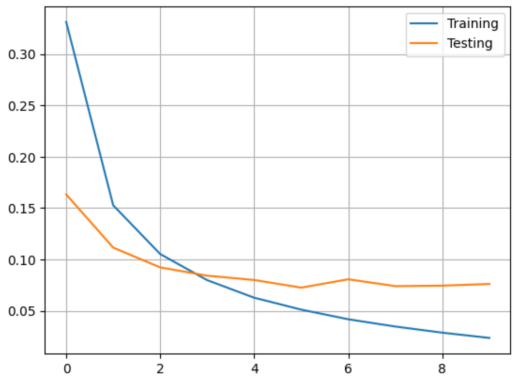
```
Test score: 0.08035963028669357
Test accuracy: 0.9768999814987183
```




```
313/313 [==============================] - 1s 2ms/step
array([[ 969,    0,    1,    2,    1,    1,    2,    1,    1,    2],
       [   0, 1125,    4,    0,    0,    1,    2,    0,    3,    0],
       [   4,    4, 1005,    1,    2,    0,    3,    7,    6,    0],
       [   0,    0,    6,  983,    0,    5,    0,    5,    2,    9],
       [   0,    0,    5,    0,  958,    0,    3,    1,    1,   14],
       [   3,    0,    0,    9,    2,  864,    4,    2,    5,    3],
       [   7,    3,    0,    0,    1,    3,  943,    0,    1,    0],
       [   1,    6,    7,    4,    1,    0,    0,  999,    1,    9],
       [   5,    0,    4,    5,    4,    2,    6,    2,  940,    6],
       [   2,    5,    0,    5,    9,    1,    0,    4,    0,  983]])
```

Résultat avec 200 neurones:

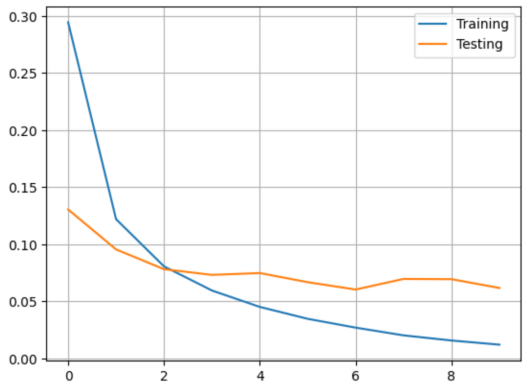
```
Test score: 0.06774899363517761
Test accuracy: 0.9793000221252441
```




```
313/313 [==============================] - 1s 1ms/step
array([[ 968,    0,    3,    0,    1,    0,    4,    1,    2,    1],
       [   0, 1130,    1,    0,    0,    1,    2,    0,    1,    0],
       [   3,    1, 1012,    3,    4,    0,    2,    5,    2,    0],
       [   0,    0,    4,  993,    1,    4,    0,    3,    3,    2],
       [   0,    1,    1,    1,  967,    0,    6,    1,    0,    5],
       [   2,    1,    0,    9,    1,  865,    8,    2,    3,    1],
       [   0,    3,    1,    1,    4,    3,  946,    0,    0,    0],
       [   0,    8,    8,    1,    0,    0,    0, 1009,    1,    1],
       [   5,    2,    7,    7,    4,    4,    4,    4,  935,    2],
       [   1,    7,    1,    6,    9,    2,    2,    8,    5,  968]])
```

Résultat avec 400 neurones:

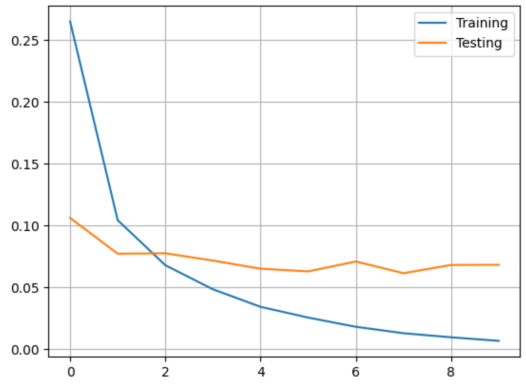
```
Test score: 0.06080867350101471
Test accuracy: 0.9817000031471252
```




```
313/313 [==============================] - 1s 2ms/step
array([[ 971,    1,    1,    0,    1,    0,    3,    1,    2,    0],
       [   0, 1124,    3,    1,    0,    1,    2,    2,    2,    0],
       [   3,    1, 1012,    0,    3,    0,    1,    8,    3,    1],
       [   0,    0,    4,  993,    0,    2,    0,    3,    3,    5],
       [   1,    0,    2,    1,  960,    0,    3,    4,    0,   11],
       [   2,    0,    0,   11,    0,  871,    3,    0,    2,    3],
       [   5,    2,    3,    1,    4,    2,  938,    1,    2,    0],
       [   0,    2,    7,    2,    0,    0,    0, 1013,    1,    3],
       [   3,    0,    3,    5,    4,    2,    1,    4,  949,    3],
       [   2,    2,    0,    5,    4,    2,    0,    8,    0,  986]])
```

Résultat avec 800 neurones:

```
Test score: 0.06336165219545364
Test accuracy: 0.9817000031471252
```




```
313/313 [==============================] - 1s 2ms/step
array([[ 969,    1,    2,    0,    0,    0,    3,    1,    4,    0],
       [   0, 1126,    2,    1,    0,    0,    2,    2,    2,    0],
       [   2,    1, 1010,    4,    3,    0,    1,    6,    5,    0],
       [   0,    0,    3,  991,    0,    4,    0,    2,    5,    5],
       [   0,    1,    3,    0,  970,    0,    3,    1,    1,    3],
       [   2,    0,    0,    6,    1,  874,    3,    1,    3,    2],
       [   3,    3,    0,    1,    2,    3,  946,    0,    0,    0],
       [   0,    4,    6,    1,    1,    0,    0, 1010,    3,    3],
       [   2,    0,    3,    3,    3,    1,    2,    4,  955,    1],
       [   3,    7,    0,    5,   16,    3,    0,    7,    2,  966]])
```

Interprétation des résultats:

A partir de 400 neurones, on commence à observer un overfitting du modèle qui