Juliette Rengot Deep Learning

Miniproject #1: Classication and Regression

PART 2 : SIMPLE CLASSIFICATION

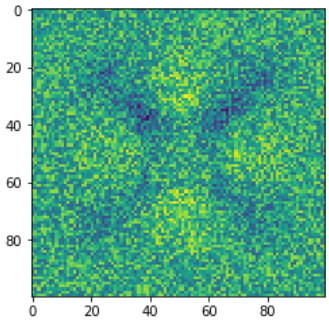
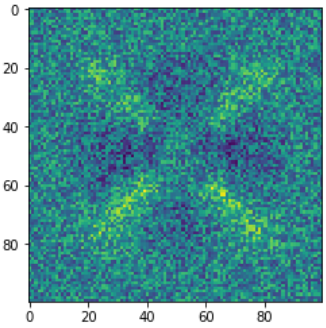
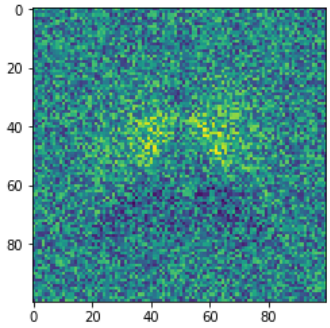
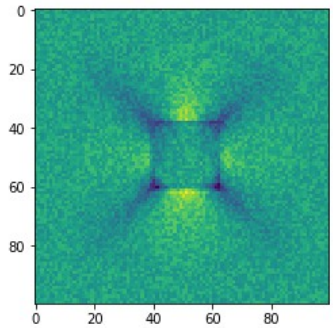
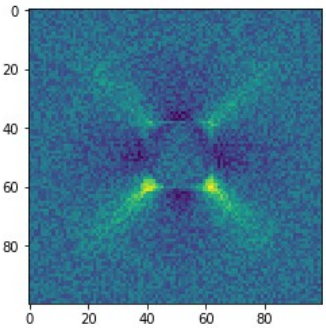
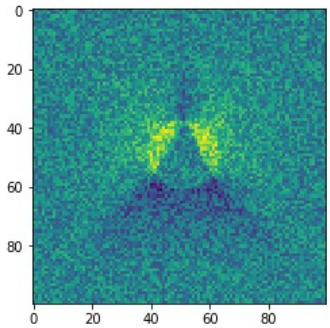
Q1) We build a model with the Keras fonction *Sequential*. Then we add a *Dense* layer using a *softmax* activation function. The input dimension is 10000 (because of the image size) and the output dimension is 3 (because we want to classify the images into three classes : 'rectangle', 'disk' and 'triangle'). We train the model on the training data. Our model has 30003 parameters.

First we use a stochastic gradient descent optimiser (*'sgd'*). We aim at decreasing the *mean_squared_error* loss function. At the end of the training, we ave loss: 0.0075 - acc: 0.9933 - categorical_accuracy: 0.9933. It is good results. When we test our model on testing data, the output is a (1x3) vector where Y_i=0 means that the data doesn't belong to the class i and Y_i=1 means that the data belongs to the class i. We see that we are able to put the rectangle data in the class 0, the disk data in the class 1 and the triangle data in the class 2, even if there is some noise. The classification is correct. The network has generalised what he has learned from the training data to correctly classify new data.

Next we use an Adam optimiser (*'adam'*) with *batch_size=32*. We aim at decreasing the *categorical_crossentropy* loss function. At the end of the training, we have loss: 0.0018 - acc: 1.0000 - categorical_accuracy: 1.0000. It is even better results. When we test our model on testing data,we obtain the same correct classification.

PART 3 : VISUALIZATION OF THE SOLUTION

Q2) We can visualise the weights as color images.

	Column 1	Column 2	Column 3
Optimiser 'sgd'			
Optimiser 'adam'			

The weights of the first column enable the network to find the rectangle data, the second column, the disk data and the last one, the triangle data. Their distribution in the above images evokes these geometrical forms.

PART 4 : A MORE DIFFICULT CLASSIFICATION PROBLEM

Q3) Now, the shapes are allowed to move within the images and change dimensions. With the previous model, the results are not satisfying. The model evaluation gives a loss value of 5.0755513890584307, an accuracy of 0.56666666686534883, and a categorical accuracy of 0.56666666686534883. To improve these results, we need a more sophisticated network.

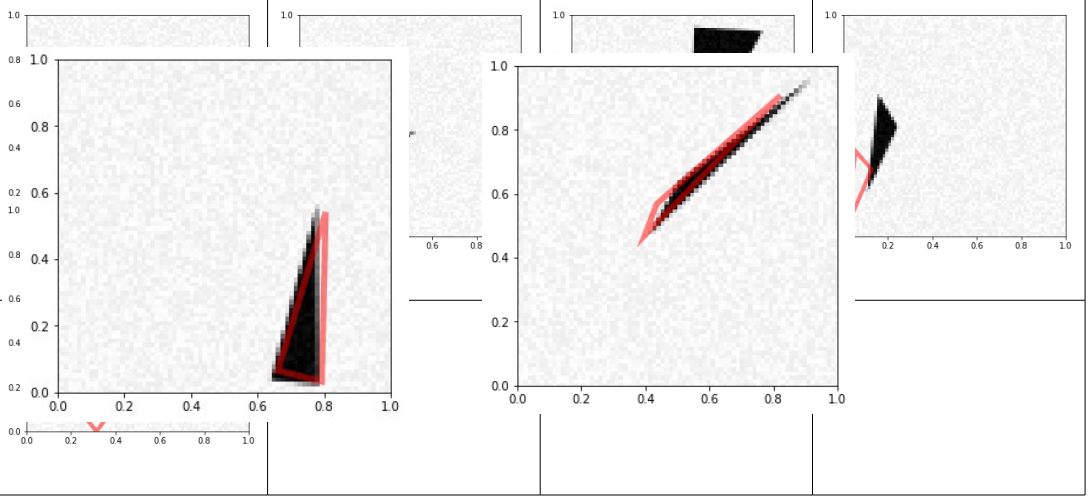
This time, the two first layer of our model are an 1D convolution of 16 filters with kernel size (5x5). Then, we add a max-pooling layer. We flatten the output and we apply a Dense layer with softmax activation. We use Adam optimiser and categorical_crossentropy loss function. The model has 240003 parameters. The model evaluation gives a loss value of 0.0025, an accuracy of 1, and a categorical accuracy of 1. We are not able to do the classification correctly.

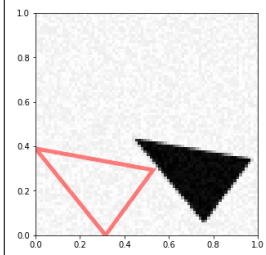
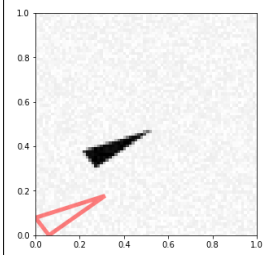
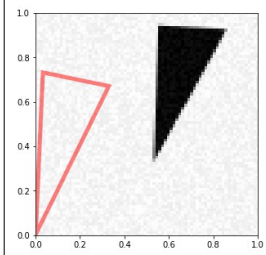
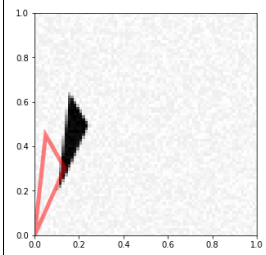
PART 5 : A REGRESSION PROBLEM

Q4) We use a model with more layers. The first layer of our model is an 1D convolution of 16 filters with kernel size (5x5) with relu activation. Then we randomly set 10% of the obtained units to 0 in order to prevent overfitting. It is a dropout layer. Then, we add a 1D convolution, a max-pooling layer, a new dropout layer, another 1D convolution and again a max-pooling layer. We flatten the output and we apply two times a Dense layer with softmax activation. We use Adam optimiser and mean squared error as a loss function. The model has 641782 parameters.

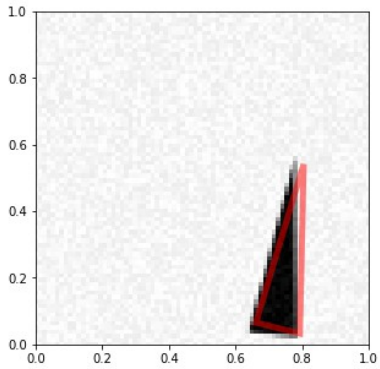
We normalise both Y_train and Y_test by apply a translation so that the triangle is not deformed but placed in the bottom left hand corner. Our train set is made of 3000 samples. The fitting takes 100 epochs. At the end on the training, we reach a loss of 0.0243 and an accuracy 0.8750 on the training set. We evaluate the model on the testing set and we get a loss of 0.026789866785208383 and an accuracy of 0.8799999992052714.

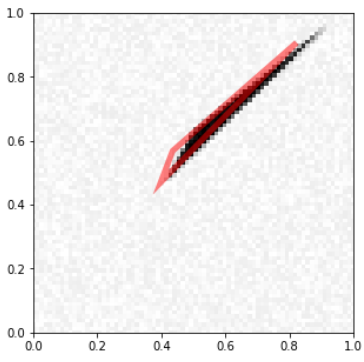
From a test data, we can make a prediction. Here are some of the visualisations.

	Sample 80	Sample 169	Sample 161	Sample 219
Predicted Y				
Real Y after normalisation				

Real Y after normalisation				

We are able to predict, with more or less accuracy the shape of the triangles but not their position in the image. To do that, we have to store the translation apply during the normalisation and apply on the result the opposite one.





The issue of that approach is that we use information on Y_test that should not be available. So, we add the translation parameters in two additional dimantions of Y_train and Y_test and we train our model to learn them too. We evaluate the model on the testing set and we get a loss of 0.05003923654556274 and an accuracy of 0.7233333325386048.

The visualisation is not satisfying. We may be closer to the real triangle but the shape is not really well recognised.

	Sample 169	Sample 139	Sample 217	Sample 219