Miniproject #1: Classification and Regression
MVA - CentraleSupélec

Vincent Lepetit

- The goal of this assignment is to learn how to implement simple image classification and regression in Keras.
- You can refer to the introductory slides on Keras:
  `https://www.labri.fr/perso/vlepetit/teaching/deep_learning_mva/intro_keras.pdf`
  and of course the online documentation on Keras `https://keras.io/`.
- You have to send your answers as a pdf by email to `vincent.lepetit@u-bordeaux.fr` by February 9, 2018. Please put 'MVA-MP1' in the subject of your email.
- You have to answer only the questions marked with the † symbol. You need to provide your code (with comments) and explain your solution.

1 Getting Started

Download the code provided at
`https://www.labri.fr/perso/vlepetit/teaching/deep_learning_mva/mp1.py`

Take some time to read it and understand it.

2 Simple Classification

You can generate a training set of images of simple geometric shapes (rectangle, disk, triangle) centered in the images by calling the function:

```
[X_train, Y_train] = generate_dataset_classification(300, 20)
```

† Build and train a linear classifier in Keras to classify a image into one of the three possible categories (i.e. rectangle, disk, triangle). Try using the stochastic gradient descent optimizer, then the Adam optimizer.

Hints: You will have to use the following functions: `Sequential`, `add`, `Dense` (do not forget the activation), `compile`, `fit`, `np_utils.to_categorical`. For the Adam optimizer, I used a batch size of 32. You should use a small number of epochs when debugging to see if the optimization seems to converge correctly.

You can check your classifier using the following code (for example):

```
X_test = generate_a_disk()
X_test = X_test.reshape(1, X_test.shape[0])
model.predict(X_test)
```

3 Visualization of the Solution

We would like to visualize the weights of the linear classifier. Check the output of the function `model.get_weights()`: The first part corresponds to the matrix of the classifier. Its columns have the same size as the input images, because Keras uses vector-matrix multiplications instead of matrix-vector multiplications.

† Visualize the 3 columns as images.

Hint: Only two (short) lines of code are required to visualize one column.

4 A More Difficult Classification Problem

Now, the shapes are allowed to move within the images and change dimensions. You can generate the new training set with:

```
[X_train, Y_train] = generate_dataset_classification(300, 20, True)
```

Retrain your linear classifier on this new training set. Add the `metrics=['accuracy']` parameter when calling the `compile` function to get the classification error in addition to the loss value.

You can generate a test set by calling:

```
[X_test, Y_test] = generate_test_set_classification()
```

and evaluate your classifier on this test set by calling:

```
model.evaluate(X_test, Y_test)
```

† Train a convolutional (not-to-)deep network on this new dataset. What is the value of the loss function on this test set when using your deep network?

Hints: You can limit yourself to 1 convolutional layer with 16 $5{\times}5$ filters, 1 pooling layer, and one fully connected layer, but you are free to use any other architecture. You are allowed to increase the number of training samples if you want to.

5 A Regression Problem

The task now is to predict the image locations of the vertices of a triangle, given an image of this triangle. You can generate a training set by calling:

```
[X_train, Y_train] = generate_dataset_regression(300, 20)
```

You can visualize a training sample (or a prediction) by calling the `visualize_prediction` function:

```
  visualize_prediction(X_train[0], Y_train[0])
```

† Build and train a regressor on this data. Evaluate your solution on the test set generated by

```
[X_test, Y_test] = generate_test_set_regression()
```

Hint: You may have to normalize somehow the vertices in `Y_train` and `Y_test` before training and testing...