Deep Learning: Cat, Dog, Car or Bike?
Teacher: Mauro Sozio

Dataset: You are provided with a dataset which contains more than 3000 pictures with either a cat, a dog, a motorbike or a car. The dataset has already been split in training, test and validation sets. Your task is to build and train a CNN which is able to recognize which object is depicted in the picture. To this end, you must use and change the code we presented during our tutorial on classifying cat and dog images.
Python and Keras version. You should use Python 3.6 (there might be some incompatibility issues between keras and the most recent versions of Python). We also recommend to use Keras 2.3. You can find the documentation for keras at the following address `https://keras.io/layers/convolutional/`.
What to submit: Please fill in the codes in the scripts q1-q4.py. You need to submit the Python scripts as well as: a) in the case of Question 2, submit `q2_result` which contains your plots; b) in the case of Question 4, submit `q4_result` which contains your plots; b) for Question 5, `q5_model` which contains your trained model (use `model.save('q5_model')` to save your model) and a report of max 1 page summarizing the most important design choices you made (e.g. CNN architecture, learning rate, etc.). The format of the report can be either pdf or doc. Please compress all the files in your-student-id.zip (You may use command zip -r your student id.zip *.py ./*result ./*model).
GPU: You will have access to the HKU GPU farm similarly to assignment 1.
Running time: When using the GPUs you should rescale the pictures to 150×150, you should be able to train a few epochs within a few minutes. This is the resolution you should consider for the assignment. In case you also would like to test the code on your machine we recommend to rescale the pictures to 32×32, in which case you should be able to train the CNNs within half an hour on the slowest machine.
Plagiarism: Discussing with other students, the professor and the teaching assistant is allowed and encouraged, however, everybody should submit his/her own solutions. Students submitting the same solutions (or very similar) for one or more questions might fail the whole assignment.

Question 1(CNN Architecture) Define a CNN architecture with the following layers stacked on top of each other in the following order:

1. A convolutional layer with 32 3×3 filters.
2. A max Pooling Layer with size 2×2.
3. A convolutional layer with 64 3×3 filters.
4. A max Pooling Layer with size 2×2.
5. A convolutional layer with 128 3×3 filters.
6. A max Pooling Layer with size 2×2.
7. A convolutional layer with 128 3×3 filters.
8. A max Pooling Layer with size 2×2.
9. A dense layer with 512 units.
10. A dense layer with k units and softmax (aka cross entropy) loss function.

Use the sigmoid activation function for all layers but the last one which uses the softmax function. Use default values for the parameters which are not specified above.

a) [7pts] Determine the right value for k. To this end, you should implement `determine_k_value` in `q1.py`.

b) [7pts] There is one issue in the architecture of the previous CNN. Fix it. You should do that without changing the number of layers, the number of filters, the size of the filters, or the number of units. To this end, implement `build_cnn_architecture` in `q1.py`.

Question 2(Training a small CNN from scratch) After answering the previous questions, train the (fixed) CNN using the following values for the parameters:

- loss function=crossentropy;
- optimizer RMSprop with $\boxed{\text{learning rate} = 0.1;}$
- metrics= accuracy;
- Batch size for the training/validation generators (if you use them) =20;
- epochs=10.

$\boxed{\text{sigmoid -> relu}}$

Train the CNN and plot both the training/validation accuracy and training/validation loss as a function of the epochs.

a)[9pts] Submit `q2_result` which contains your plots and implement `multiple_choice` in `q2.py`.

b)[9pts] There are two parameters you can change so as to improve the results significantly. Change the values of those two parameters so as to improve the results. To this end, you should implement `modified_cnn` in `q2.py`.

Question 3(Optimize the learning rate) a)[12pts] Determine a range of possible values for the learning rate, which is "wide enough". In particular, you should try to guarantee that your interval contains an optimal value for the learning

rate. At the same time the interval that you provided should not be too wide, due to efficiency reasons. Intervals that are deemed to be too wide will be penalized. To this end, implement `learning_rate_range` in `q3.py`.
b)[12pts] Provide three values for the learning rate for each of the following cases "bad", "good" and "very good". To this end, implement `learnign_rate_examples` in `q3.py`.

Question 4(Transfer Learning)[14pts] Use the VGG16 as feature extractor with data augmentation (i.e. remove the top layer and freeze the VGGnet). Plot both the training/validation accuracy and training/validation loss as a function of the epochs (epochs=20) and submit the plots. To this end, implement `transfer_learning_with_vggnet` in `q4.py`. Submit `q4_result` which contains your plots.

Question 5(Open Question)[30pts] Use any of the techniques we saw during our course so as to improve the accuracy of your CNN. Your architecture will be evaluated on a test set that will not be revealed to you, while the codes will be ranked according to their accuracy on our test set. The number of points you will receive for this question depends on the position of your code in the ranking. The architectures with highest accuracy will be presented at the end of our course. Submit `q5_model` which contains your trained model (use `model.save('q5_model')` to save your model) and a report of max 1 page summarizing the most important design choices you made (e.g. CNN architecture, learning rate, etc.). The format of the report can be either pdf or doc.