CIFAR-10 – Neural Networks (60 points)

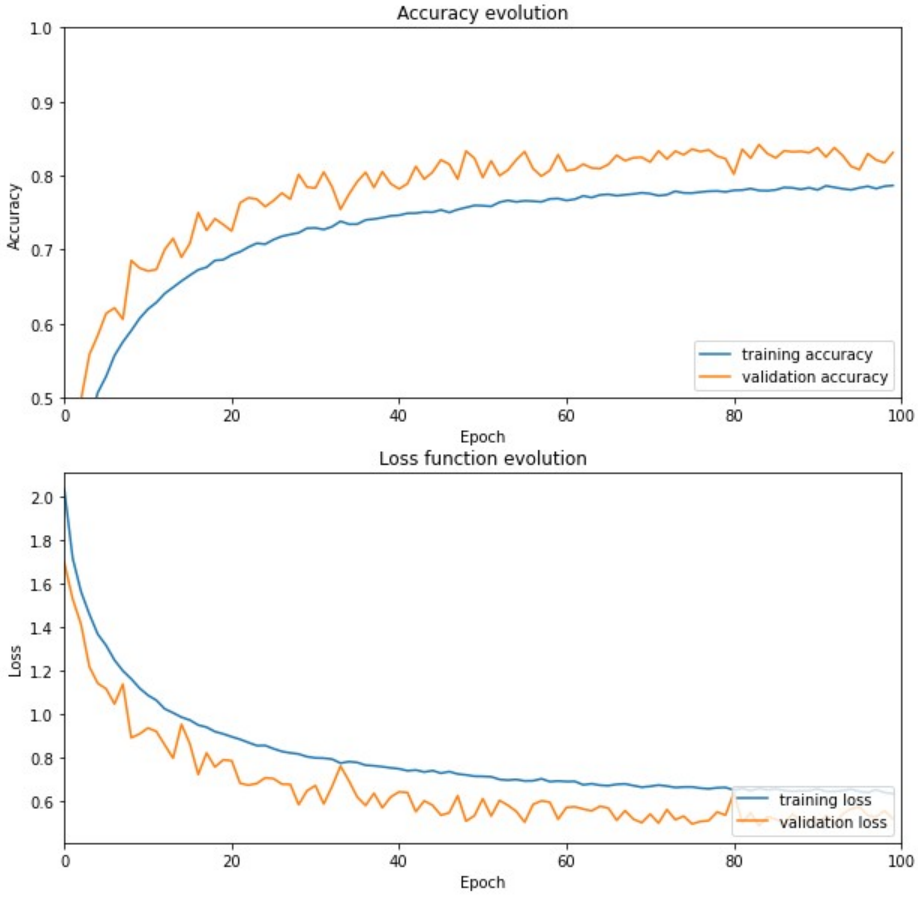
Summary of techniques I used in my implementation:

- VGG-style network architecture. (https://arxiv.org/pdf/1409.1556.pdf)
  - Three VGG blocks.
  - Convolutional layers with increasing filter sizes (23 to 128) and 3x3 sized convolution windows.
  - 2x2 pooling layer (downsampling) after each convolution layer.
  - Dropout regularization after each pooling layer with increasing dropping rate (0.1 to 0.5).

Model were trained on 100 epochs and with batch size of 64. Categorical cross entropy was used as a loss function.

I also used data augmentation techniques to modify the training data (eg. pixel shift, flipping, small rotations).

Here is reported performance history during training process:



```
Epoch 98/100
703/703 [==============================] - 15s 21ms/step - loss: 0.6499 - accuracy: 0.7824 -
val_loss: 0.5251 - val_accuracy: 0.8210
Epoch 99/100
703/703 [==============================] - 15s 21ms/step - loss: 0.6365 - accuracy: 0.7856 -
val_loss: 0.5526 - val_accuracy: 0.8174
Epoch 100/100
703/703 [==============================] - 15s 21ms/step - loss: 0.6313 - accuracy: 0.7866 -
val_loss: 0.5163 - val_accuracy: 0.8310

In [45]: runcell(5, '/home/tuomas/Python/DATA.ML.100/Ex4/cifar10nn3.py')

In [46]: runcell(6, '/home/tuomas/Python/DATA.ML.100/Ex4/cifar10nn3.py')
313/313 - 1s - loss: 0.5700 - accuracy: 0.8171
```

Classification accuracy of my implementation: 82%.

Due to the output I received during training, it seemed that the learning speed become slower and slower at the end of the epoch cycle but I didn’t observe any notes about overfitting.

Overall, the neural network performed remarkably well compared to the earlier classification models implemented during this course (1NN & Bayes classifier). This merit is achieved via the enormous number of free parameters that neural network contains (especially deep convolutional networks) so the model have abilities to fit to the data very well.