Exercise 6

Project Description

**Title**: Convolutional Neural Network for Multi-class image classification

**Team**: Sascha and Shivali Dubey.

**Problem Definition**:

Multi-class image classification has various applications, for instance, in self-driving cars to detect and classify pedestrians, motorcycles, trees, bicycles etc; classification of features on the Earth such as roads, rivers, agricultural fields etc using satellite images. With the advancements in deep learning, every year, new algorithms/ models keep on outperforming the previous ones, to achieve the best possible accuracies for image classification. One of the most popular dataset used is the ImageNet dataset. In our project we propose to implement the deep learning algorithms based on a few selected studies [1,2,3] with the aim to attain the best possible classification accuracy.

**Dataset**: Tiny ImageNet. Tiny Imagenet has 200 classes. Each class has 500 training images, 50 validation images, and 50 test images.  
Source: https://tiny-imagenet.herokuapp.com/

**Approach**: With such a large dataset, one of the main challenges of classification is diversity of the images. Our model/algorithm must be able to handle fine-grained and specific classes even when they are hard to distinguish. In other words, we need to maximize inter-class variability, while minimize intra-class variability. At the same time, attaining the best possible classification accuracy is always a challenge for any given algorithm. The predictions go wrong when you have too many false positives and false negatives.

The project has been divided into two parts. Each team member performs one task as described below.

Task 1 (Sascha): In context of deep learning, the input images and their subsequent outputs are passed from a number of such filters. The numbers in filters are learnt by neural net and patterns are derived on its own.

These kind of problems need to leverage the ideas or concepts learnt from image classification as well as from object localization.

We are using CNN because the weights and filters are learnt automatically, also the weights are same at all locations.

**Step 1: Break the image into overlapping image tiles**

Similar to our sliding window search above, let’s pass a sliding window over the entire original image and save each result as a separate, tiny picture tile:

#### Step 2: Feed each image tile into a small neural network

We’ll keep the **same neural network weights** for every single tile in the same original image. In other words, we are treating every image tile equally. If something interesting appears in any given tile, we’ll mark that tile as interesting.

#### Step 3: Save the results from each tile into a new array

We don’t want to lose track of the arrangement of the original tiles. So we save the result from processing each tile into a grid in the same arrangement as the original image. It looks like this:

In other words, we’ve started with a large image and we ended with a slightly smaller array that records which sections of our original image were the most interesting.

#### Step 4: Downsampling

The result of Step 3 was an array that maps out which parts of the original image are the most interesting. But that array is still pretty big:

To reduce the size of the array, we downsample it using an algorithm called [max pooling](https://en.wikipedia.org/wiki/Convolutional_neural_network#Pooling_layer). It sounds fancy, but it isn’t at all!  
So far, we’ve reduced a giant image down into a fairly small array.

Guess what? That array is just a bunch of numbers, so we can use that small array as input into another neural network. This final neural network will decide if the image is or isn’t a match. To differentiate it from the convolution step, we call it a “fully connected” network.

So from start to finish, our whole five-step pipeline looks like this:

For example, the first convolution step might learn to recognize sharp edges, the second convolution step might recognize beaks using it’s knowledge of sharp edges, the third step might recognize entire birds using it’s knowledge of beaks, etc.