Assignment 2

Deadline: Friday, 29th May (23:59)

15th May, 2020

Question 1: Neural Codes for Image Retrieval (10pt)

Use the representations learned by a convolutional neural network (ConvNet) for image retrieval, as proposed in

> A. Babenko, A. Slesarev, A. Chigorin, V. Lempitsky, *Neural Codes for Image Retrieval*, ECCV, 2014 (https://arxiv.org/abs/1404.1777).

You can use the Jupyter notebook `2IMM10_Assignment_2_1.ipynb`, which already downloads, loads and pre-processes the data, and provides some helper functions. Write your code between all two consecutive occurrences of "`# ...`". See the text cells in the notebook for additional information.

Data: Tiny Imagenet

Use the training set of *Tiny Imagenet*,[1] consisting of 200 classes, with 500 images per class (total $100,000$ images), each image being of dimensions 64×64 RGB. The Jupyter notebook already gets the data for you, and also performs the following steps.

- Split the 200 classes into two sets, one containing 190 classes and the other containing the remaining 10 classes.
- Shuffle the set with 190 classes and divide it into *training*, *validation*, and *test* sets, according to the proportions 80/10/10. These will be used to train, validate and test the model (Task 1.2).
- The set with the remaining 10 classes serves as *out-of-domain* (ood) data, used for image retrieval (Task 1.3).
- Normalize pixel values to $[0, 1]$.

Task 1.1: Implement ConvNet (3pt)

Implement a modified version of the ConvNet architecture from Babenko et al.:

- For `Layer 1`, use kernel size 4×4 (instead of 11×11) and stride 1 (instead of 4).
- For the hidden fully connected layers, `Layer 6` and `Layer 7`, use 2048 units (instead of 4096).
- Consult the paper for the rest of the architecture.
- For all pooling layers, use a window size of 3×3 and stride 2.
- Apply dropout with rate 0.5 before `Layer 6` and `Layer 7`.

[1] https://tiny-imagenet.herokuapp.com/

Task 1.2: Train and Evaluate ConvNet (2pt)

- Train the model by optimizing *cross-entropy* on the training set with the *Adam* optimizer, using a learning rate of 0.0001 and a *batch size* of 100. Set Adam's flag *amsgrad* to *True*.
- Train for maximal 100 epochs and perform early stopping, by monitoring the loss on the validation set, and terminating training as soon as it starts to increase.
- Evaluate and report the *train*, *validation* and *test* performance, in terms of *cross-entropy*, *classification accuracy* and *top-5 classification accuracy*.
- Name two techniques which would likely improve the test accuracy.

Task 1.3: Image Retrieval (5pt)

Use the trained ConvNet to perform image retrieval on the ood data. When using a certain image as query image, the remaining $4,999$ images should serve as a retrieval date base. Perform image retrieval using neural codes from the same 3 layers which were also used in the paper by Babenko et al. Compare the results.

- Obtain neural codes for each image in the ood data.[2]
- Normalize the codes to have unit length.
- For the first 10 images in the ood set, find the respectively 5 closest[3] images in the data base. Plot the query image next to the 5 retrieved images (sorted from most to least similar) and mark the images which have the same class as the query image (see Fig. 2 and 3 in the paper).
- What are the qualitative differences between the different layers for neural codes?
- Compute and report the *mean average precision* (mAP) over the whole ood set, for each of the 3 layers.
- Do the observed mAP values (roughly) confirm the observations by Babenko et al.?

Question 2: Triplet networks & one-shot learning (10pt)

In practice 4b.4, we train a Siamese network for one-shot learning task on the Omniglot dataset. In this assignment, we will work on the same data set with the same task but extend it to triplet networks, we will also compare our model performance under different triplet selection method. The assignment contains the following 4 tasks.

Task 2.1: Build the triplet network (3pt)

We will define a triplet Network for use with the Omniglot dataset. Each branch of the triplet is a **Convnet** model that transforms data to an embeddings space.

HINT: you may need **Concatenate** from keras.layer to merge the output layer

Task 2.2: Define triplet loss (2pt)

You can find the formula of the triplet loss function in our lecture note.

HINT: you can play with the margin value to get better model performance.

[2]The Jupyter notebook already provides functions to get these codes.

[3]Hint: You might want to exploit the relation between inner products and Euclidean distances.

Task 2.3: Select triplets for training (3pt)

We have two different options for the triplet selection method, and we will compare the model performance under these two selection methods after building our model.

a) Random triplets selection, including the following steps:

   - Pick one random class for anchor.
   - Pick two different random picture for this class, as the anchor and positive images.
   - Pick another class for Negative, different from anchor class.
   - Pick one random picture from the negative class.

b) Hard triplets selection. For easy implement, for a picked anchor, positive pair, we will choose the hardest negative to form a hard triplet, that means, after picking an anchor, positive image, we will choose the negative image which is nearest from anchor image from a negative class, i.e : distance between anchor image and negative image can get the minimum value. The whole process including the following steps:

   - Pick one random class for anchor.
   - Pick two different random picture for this class, as the anchor and positive images.
   - Pick another class for Negative, different from anchor class.
   - Pick one hardest picture from the negative class.

HINT: when picking the hardest negative, you may need the model.predict to get the embedding of images, the calculate the distances

Task 2.4: One-shot learning with different selection method (2pt)

With different triplets selecting method (random and hard), we will train our model and evaluate the model by one-shot learning accuracy.

- You need to explicitly state the accuracy under different triplets selecting method
- When evaluating model with **test oneshot** function, you should evaluate on 20 way one-shot task, and set the number (k) of evaluation one-shot tasks to be 250, then calculate the average accuracy.

HINT: After training our model with random selection method, before train model under hard triplets selection, you should re-build our model (re-run the cell in Task 2.1) to initialize our model and prevent re-use the trained model of random selection.

HINT: you can re-use some code from practice 4b.4.

Question 3: Peer review (0pt)

Finally, each group member must write a single paragraph outlining their opinion on the work distribution within the group. Did every group member contribute equally? Did you split up tasks in a fair manner, or jointly worked through the exercises? Do you think that some members of your group deserve a different grade from others?