Assignment 2

Deadline: Tuesday, March 13th (noon)

February 26, 2018

Question 1: Neural Codes & Nearest Neighbor retrieval (7.5pt)

The Caltech101 dataset consists of images of 101 different objects. In this question you will develop an image retrieval system using image representations (neural codes) learned with a deep convolutional neural network and a given distance metric.

In the tasks below you will need to implement the following steps:

- Retrieval for n selected (distinct) query images from the dataset
  - For each query image, obtain the 5 most similar images (excluding the query image itself!)
- Evaluation of the quality of the retrieval
  - The Caltech101 images are annotated with their object class. Use these annotations to evaluate the accuracy of the retrieval task.
  - For each query image, count the number of images whose class corresponds to the one from the query. The score of the retrieval for that image then ranges between:
    - **5** *all* retrieved images' classes agree with the query image class
    - **0** *none* of the images' classes agree with the query image class
  - Compute the average of all n queries

Task 1.1: Neural codes image retrieval

a) Implement the retrieval task and evaluate the results for $n = 200$ images. Use the provided VGG16 network pre-trained on ImageNet to compute "neural codes" and L2-distance. Specifically use the codes produces by the following layers of the model:

- the "fc1"-layer
- the "fc2"-layer

Provide the retrieval evaluation scores for both tasks.

b) Which representation ("neural code") provided better features for the given retrieval task? Justify your answer and discuss possible reasons for the observed results. Relate your answer to the conclusions in the paper "Neural Codes for Image Retrieval".

Task 1.2: Detailed evaluation

a) The retrieval scores can vary from one query image to another. Some images are quite representative and for them retrieval works well, some are not so much. For the same retrieval task given above using "fc2"-features, find (if possible) six query images such that they range from excellent to poor retrieval performance. More specifically find example query images that result in query scores of exactly 0, 1, 2, 3, 4, and 5.

   Visualise the six (or less) resulting query images.

b) Looking at the results, what can you say about the "types" of images that obtain good retrieval scores compared to those obtaining poor retrieval scores? Give an explanation and possible solution(s).

   (HINT: How did we obtain data representations for similarity measures?)

Task 1.3: Subjective evaluation

We will now use the "fc2"-features to do image retrieval for query images from the "BACKGROUND_Google" set from the Caltech101 dataset. These images are not associated to a particular class, so we will evaluate them subjectively instead.

a) Find two query images from the "BACKGROUND_Google" class, such that for the first query image relevant/similar images are retrieved (according to your own definition of relevancy/similarity), and for the second image mainly irrelevant/dissimilar images are retrieved. For each of them, visualise its 5 nearest neighbors in the Caltech101 dataset *(so do NOT retrieve images from the "BACKGROUND_Google" class!)*, according to the "fc2-features" and L2-distance.

b) Motivate your idea of "relevance": why do you consider the results for the first image relevant/similar, and those for the second image irrelevant/dissimilar?

c) Explain why you think this retrieval method (nearest neighbor for neural codes from VGG16) performs better on the first image than on the second.

Task 1.4: Dimensionality reduction

a) So far we've been using 4096-dimensional neural codes. This space is however still quite high-dimensional. Apply a dimensionality reduction method and evaluate the effect on the retrieval performance.

   - Use PCA to obtain lower-dimensional representations of the Caltech101 data "fc2"-features (try the same compression rates as in Table 2 of the "Neural Codes for Image Retrieval" paper).
   - Evaluate the same retrieval task as explained at the start of this question for each of the compression rates/dimensionalities. Report the retrieval scores.

   HINT: See `http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html` *on how to transform a dataset with PCA.*

b) Discuss your results: how much can you further reduce the dimensionality of the data representations, without affecting the retrieval performance (much)? Compare these results to those from the paper, are your conclusions similar or not?

Question 2: Fashion-MNIST (12.5pt)

For this question we will work with the "Fashion-MNIST" dataset. This dataset is modelled to have the same specifics as MNIST; it consists of a training set of 60,000 examples, and a test set of 10,000 examples. Each example is a 28x28 greyscale image, associated with a label from one of 10 classes. The images represent various clothing items (as opposed to handwritten digits for MNIST), each class represents a different type of clothing item. The following classes exist:

- 0: T-shirt/top
- 1: Trouser
- 2: Pullover
- 3: Dress
- 4: Coat
- 5: Sandal
- 6: Shirt
- 7: Sneaker
- 8: Bag
- 9: Ankle boot

In this question we will investigate various ways to model visual similarity for this dataset, in order to perform image retrieval. For more info about the dataset, see `https://github.com/zalandoresearch/fashion-mnist`.

The dataset can directly be obtained through Keras.

Consider the following situation: We have a fully labelled dataset (the **labelled set**) of the images from the first 5 classes (t-shirts/tops, trousers, pullovers, dresses, coats). We are then supplied with an unlabelled dataset (the **retrieval set**) containing the remaining Fashion-MNIST images (sandals, shirts, sneakers, bags, ankle boots) on which we want to be able to perform image retrieval. So we cannot use labels from the retrieval set, since we do not know them (note that in our case we do have the labels, but we will only use them for evaluation).

Use the code provided to split the dataset up into two sets representing 5 classes each. Observe that the labelled and the retrieval set have exactly the same size.

Task 2.1: Fashion neural retrieval

a) Design an MLP (multilayer perceptron) for classification on the first 5 classes of the Fashion-MNIST dataset (i.e. only use `x_train_l` for training). You may include Dropout and BatchNormalization if needed. Let the last hidden dense layer (before the 5-dimensional output layer) have 128 dimensions. *(HINT: you can use* `name="neural_codes"` *for this layer to make it easier to obtain features from it later.)*

   Train it to classify images into their corresponding classes. Make sure that it achieves decent accuracy (at least 90%) on the labelled test set `x_test_l` (show this!). Save the trained model to a ".h5" file. (make sure you're using Keras version 2.1.3!)

b) Briefly motivate how and why you chose this architecture.

Task 2.2: Fashion neural retrieval #2

a) Design a CNN (convolutional neural network) for classification on the first 5 classes of the Fashion-MNIST dataset (i.e. only use `x_train_l` for training), consisting of a number of Convolutions with Max-Pooling, followed by one or more Dense layers. You may use Dropout and BatchNormalization to improve generalization and training speed. Let the last hidden dense layer (before the 5-dimensional output layer) have 128 dimensions. *(HINT: you can use* `name="neural_codes"` *for this layer to make it easier to obtain features from it later.)*

Train the CNN to classify images into their corresponding classes. Make sure that it achieves decent accuracy (at least 94%) on the test set `x_test_l` (show this!). Save the trained model to a ".h5" file. (make sure you're using Keras version 2.1.3!)

b) Briefly motivate how and why you chose this architecture.

Task 2.3: Fashion neural retrieval #3

a) Design a (convolutional) Denoising Autoencoder (DAE) for the full Fashion-MNIST dataset (i.e. use `x_train`, not `x_train_l`). For the encoder, use only Convolutional layers and Max-Pooling, followed by a Dense layer with 128 units. The output of this layer will be the "code" of the autoencoder (*HINT: you can use* `name="neural_codes"` *for this layer to make it easier to obtain features from it later*). For the decoder, start with a Dense layer to upscale to a suitable dimension, and then use only Convolutional layers and UpSampling. You may use BatchNormalization to speed up training.

Train the DAE to reconstruct noisy images to the original input images. Make sure that it achieves a binary cross-entropy loss of at most 0.29 on the test set (show this!). Save the trained model to a ".h5" file. (make sure you're using Keras version 2.1.3!)

b) Briefly motivate how and why you chose this architecture.

c) Visualise a few test examples, their noisy versions, and their reconstructions. Do you consider the results acceptable? Do you think they can be useful for image retrieval? Explain why in one or two sentences.

d) Why can we train on the full dataset `x_train` here, whereas in Tasks 2.1 and 2.2 we had to use `x_train_l` (the first 5 classes only) for training?

Task 2.4: Fashion neural retrieval #4

Autoencoders come in different shapes and sizes. One key defining property of autoencoders is the means the model uses to prevent the learning of the identity function. Typically, this is done with different regularization methods. In the previous task you used a model that uses noise as a regularizer. In this task you will develop a Sparse Autoencoder (SAE). A sparse autoencoder uses a sparsity regularization to obtain sparse representations of the input data. Sparsity can be achieved by using L1-regularization on the activations of the hidden "code" layer.

a) Design a (convolutional) Sparse Autoencoder (SAE) for the full Fashion-MNIST dataset (i.e. use `x_train`, not `x_train_l`). For the encoder, use only Convolutional layers and Max-Pooling, followed by a Dense layer with 128 units. The output of this layer will be the "code" of the autoencoder (*HINT: you can use* `name="neural_codes"` *for this layer to make it easier to obtain features from it later*). Add an activity regularizer to this layer, using `regularizers.l1(10e-5)` from Keras. For the decoder, start with a Dense layer to upscale to a suitable dimension, and then use only Convolutional layers and UpSampling. You may use BatchNormalization to speed up training.

Train the SAE to reconstruct input images. Make sure that it achieves a loss value of at most 0.31 on the test set (show this!). Save the trained model to a ".h5" file. (make sure you're using Keras version 2.1.3!)

b) Briefly motivate how and why you chose this architecture.

c) Visualise a few test examples and their reconstructions. Compare the visual results to those of the DAE in Task 2.3. Also compare the loss values of the test set for the DAE and SAE. How can you explain the difference?

Task 2.5: Comparison

Obtain 128-dimensional neural code representations of the last five classes of the Fashion-MNIST dataset (the retrieval set: `x_train_r`) from the following models/layers:

1. The last dense hidden layer (before the output layer) of the MLP you trained in Task 2.1
2. The last dense hidden layer (before the output layer) of the CNN you trained in Task 2.2
3. The center layer/code of the DAE you trained in Task 2.3
4. The center layer/code of the SAE you trained in Task 2.4
5. A PCA-transformation

a) Evaluate the retrieval task as described in Question 1 on the last 5 classes (the retrieval set) of the Fashion-MNIST dataset, for the five data representations given above. Use query images from the test set and retrieve images from the training set only. Print the five resulting retrieval scores (between 0 and 5).

b) Compare the "baseline" PCA-transformed data with the other methods. Is PCA a suitable method to obtain representations for image retrieval in this situation? Why do you think so? Would you expect a similar conclusion for the Caltech101 dataset from Question 1?

c) Observe the difference between encodings from the DAE and SAE. Discuss the difference in encodings between the two autoencoders (denoising and sparse). Also discuss the difference in retrieval performance for these encodings. How would you explain this difference?

d) What is the best performing method you found in part a)? Describe what advantage you believe this method has over the others.