CSC420 Intro to Image Understanding

Assignment 5 Fall 2018

Posted: Nov 14, Due Nov 21

*Submit your solutions in a pdf on MarkUs, along with **all of your** code. Submit your source code files instead of pasting it in the pdf. Include images/printouts of your output in your pdf when asked. For each piece of code, specify the question to which it corresponds. Make sure to comment your code, if your implementation doesn't work, you can get partial credit for your approach if explained in the comments. For written questions, explain your reasoning to get full credit.*

Question 1: Facial Recognition and Fast Image Retrieval (7 points)

In this question you will implement a simplified recognition and retrieval system for images of faces based on the one discussed in class. Suppose you have a dataset of facial images for your employees. You want to build a system that can recognize an employee and let them into the office, or detect a stranger and deny access. The provided pretrained model and data is obtained from https://www.coursera.org/learn/convolutional-neural-networks/home/week/4. Refer to facenet.py for loading the model and data.

a. **[0.5 points]** Suppose you have a few (around 5 images) per employee. Is it a good idea to train a neural network to classify an input image into either one of the employees and an "unknown" category? Why or why not?

b. **[0.5 point]** When using a bag of visual words description of images, it is helpful to use a TF-IDF (term frequency inverse document frequency) weighting of each word. Explain the benefit of using TF-IDF over a present/absent representation.

c. **[0.5 point]** For our recognition and retrieval system we want our descriptors to be invariant to changes in pose or lighting. We can use a pretrained model to recover an embedding for faces that maps images of faces to a 128-dimensional vector. For more details check out FaceNet https://arxiv.org/pdf/1503.03832.pdf. The provided code includes the Keras model and pretrained weights. The model takes in a 96x96 RBG image, and outputs a 128 dimensional embedding. For each image in the saved_faces directory compute and save its embedding. Note that in this case each image has only one descriptor which is the embedding.

d. **[0.5 points]** For image retrieval, the embeddings have to be clustered. Explain the purpose of this step.

e. **[1 point]** Using K-means clustering, cluster your saved embeddings into 6 clusters (number of distinct persons in the data). You may use the scikit-learn function for k-means clustering. What should you choose as your visual word representation?

f. **[1 point]** Write an inverted index for your saved_faces dataset. For this step, you can assign an integer from 0 to 5 for each distinct visual word. You can save your visual words in a 2D

numpy array, using the index to select a specific visual word. You can save it for later use using numpy.save(). Then for each image in saved_faces, find its corresponding visual word (note that you can use the output of K-means for this). Your inverted index can be a dictionary where the key is the index of the word and the values are image filenames.

g. **[1 point]** For each image in input_faces, you want to find it's matching images from saved_faces. Describe a method to do this.

h. **[1 point]** Implement your method from part g. Note that one of the images is a new person and should produce no matches. Your output file should include the input image name, along with the names of all images that matched next to it. Don't worry if there are a few false matches. You will be marked based on your approach rather than having a perfect output.

i. **[1 points]** So far the task was to find corresponding images where there is only one person per image. Suppose you hold an office party and take lots of pictures. Your employees want to find pictures of themselves, even when taken with other people. Describe a more general retrieval system based on the idea of face embeddings. What are the challenges in this case? You may use drawings/diagrams to support your explanation.

Question 2: Deep Learning (5 points)

In this question you will experiment with training and evaluating an image classifier on the fashion-MNIST dataset (because MNIST is too easy).
Refer to https://www.tensorflow.org/tutorials/keras/basic_classification to learn how to interact with the dataset using tensorflow or keras,
Refer to https://pytorch.org/docs/master/torchvision/datasets.html#fashion-mnist if you prefer to use pytorch.
For free GPU access you can use https://colab.research.google.com/ to speed up training.

a. **[0.5 points]** Describe the vanishing gradient problem. What are some ways to mitigate the vanishing gradient problem?

b. **[0.5 points]** Why do CNN's include max pooling layers?

c. **[0.5 point]** After loading and preprocessing the data:
   i. Show a bar graph of the number of instances in each class.
   ii. Split your data into a training set and validation set. Something like 70-30 is fine.

d. **[0.5 point]** Build a model using keras, tensorflow or pytorch. Think about your choice of activation functions, number of layers...etc What should the activation at the output be? Don't worry if you don't initially have an idea about the hyperparameters of your model, the point is to try different things and experiment.

e. **[0.5 point]** Define your loss function in terms of you model's prediction and ground truth label. Write it down in the pdf.

f. **[1.5 points]** Write a training loop to train your model (you may use Adam as your optimizer, but feel free to experiment). Credit is given for plotting the training loss and validation loss, as well as the training and validation accuracy. What is your choice of batch size? When should you stop training? You may change your initial architecture and hyperparameters if you don't like your results. Try to get over 88% test set accuracy. In your report, make a diagram of your final architecture, learning rate, batch size, number of epochs trained and final training/validation loss and accuracy. Include your plots in your report.

g. **[1 point]** Plot the training and validation curves (loss and accuracy) for different batch sizes. Try 8, 16, 32, 64. Include the plots in your report and comment on your results.