Assignment 1 - Kaggle Part

Face Classification using Convolutional Neural Network

AIST 4010: Foundation of Applied Deep Learning (Spring 2022)

DUE: **11:59PM (HKT), Feb. 14, 2022**

1 Introduction

Even though face recognition may sound quite trivial to us humans, it remained a challenging computer vision problem in the past decades. Thanks to deep learning methods, computers now can leverage huge dataset of faces to learn rich and compact representations of human faces, allowing models to even outperform the face recognition capabilities of humans.

Face recognition mainly consists of two parts. The task of classifying the ID of the face is known as **face classification**, which is a closed-set problem. The task of determining whether two face images are of the same person is known as **face verification**, which is an open-set problem[1].

In this assignment, you will use Convolutional Neural Networks (CNNs) to design an end-to-end system for the face classification task (well, other techniques are required if you want to get higher scores.) In this task, your system will be given an image of a face as input and should output the ID of the face.

You will train your model on a dataset with a few thousand images of labelled IDs (i.e., a set of images, each labeled by an ID that uniquely identifies the person.) You will learn more about embeddings[2], several loss functions, and, of course, convolutional layers as effective shift-invariant feature extractors. You will also develop skills necessary for processing and training neural networks with big data, which is often the scale at which deep neural networks demonstrate excellent performance in practice.

- Goal: Given a person's face, return the ID of the face
- Kaggle: `https://www.kaggle.com/c/aist4010-spring2022-a1`

2 Face Classification

2.1 Face embedding

Before we dive into implementation, let's ask ourselves a question: how do we differentiate faces? Yes, your answers may contain skin tone, eye shapes, etc. Well, these are called facial features. Intuitively, facial features vary extensively across people (and make you different from others). Your main task in this

[1] For close-set task, all testing identities are predefined in training set. For open-set task, testing identities typically do not appear in training set.

[2] In this case, embeddings for face information.

assignment is to train a CNN model to extract and represent such important features from a person (You can also include some additional architectures or techniques to improve your performance). These extracted features will be represented in a fixed-length vector of features, known as a **face embedding**.

Once your model can encode sufficient discriminative facial features into face embeddings, you can pass the face embedding to a fully-connected(FC) layer to generate corresponding ID of the given face.

Now comes our second question: how should we train your CNN to produce high-quality face embeddings?

2.2 Multi-class Classification

It may sound fancy, but conducting face classification is just doing a multi-class classification: the input to your system is a face image and your model needs to predict the ID of the face.

Suppose the labeled dataset contains a total of M images that belong to N different people (where $M > N$). Your goal is to train your model on this dataset so that it can produce "good" face embeddings. You can do this by optimizing these embeddings for predicting the face IDs from the images. The resulting embeddings will encode a lot of discriminative facial features, just as desired. This suggests an N-class classification task.

A typical multi-class classifier conforms to the following architecture:

$$\text{Classic multi-class classifier} = \text{feature extractor(CNN)} + \text{classifier(FC)}$$

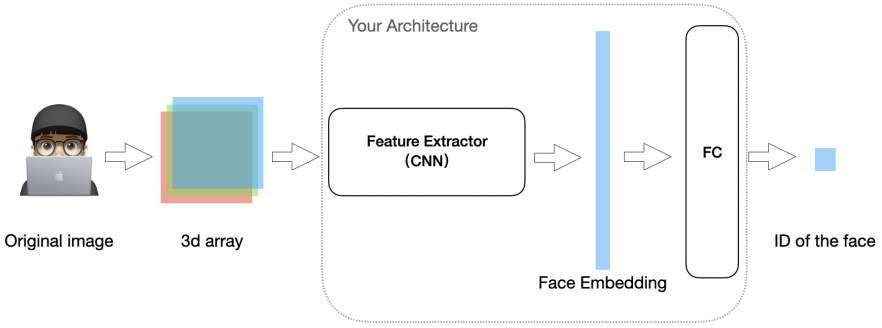

Figure 1: A typical face classification architecture

More concretely, your network consists of several (convolutional) layers for feature extraction. The input will be (possibly a part[3] of) the image of the face. The output of the last such feature extraction layer is the face embedding. You will pass this face embedding through a linear layer whose dimension is *embedding dim* $\times$ *num of faceids*, followed by Softmax, to classify the image among the N (i.e., num of faceids) people. You can then use cross-entropy loss to optimize your network to predict the correct person for every training image.

The ground truth will be provided in the training data (making it supervised learning). You are also given a validation set for fine-tuning your model. Please refer to the **Dataset section** where you can find more details about what dataset you are given and how it is organized. To understand how we (and you) evaluate your system, please refer to the **System Evaluation section**.

That's pretty much everything you need to know for your first Kaggle competition. Go for it!

[3]It depends on whether you pre-process your input images

3 Dataset

The data for the assignment can be downloaded from the Kaggle competition link[4]. The dataset contains images of size 64×64 for all xyz channels. In this competition, we are dealing with faces from 1000 people (That being said, we have 1000 classes.)

You will be given a human face image. What you need to do is to learn to classify this image into correct people IDs.

3.1 File Structure

The structure of the dataset folder is as follows:

- `classification-data`: Each sub-folder in train, val and test contains images of one person, and the name of that sub-folder represents their ID.
  - `train`: You are supposed to use the train data set to train your model for the classification task.
  - `val`: You are supposed to use val data to validate the classification accuracy.
  - `test`: You are supposed to assign IDs for images in test data and submit your result.
- `test-list.txt`: This file contains the trials for Classification Test. The first column is the images name. Your task is to assign ID to each image and generate submission file based on the order given here.
- `sample-submission.csv`: This is a sample submission file for this competition.

3.2 Loading Training Data - ImageFolder

To load the images, we recommend that you look into the ImageFolder dataset class of PyTorch at `https://pytorch.org/vision/main/generated/torchvision.datasets.ImageFolder.html`. The images in sub-folders of classification data are arranged in a way that is compatible with this dataset class.

4 System Evaluation

The evaluation metric is quite straightforward:

$$accuracy = \frac{\#\ correctly\ classified\ images}{\#\ total\ images} \tag{1}$$

5 Submission

Following are the deliverables for this assignment:

- Kaggle submission. Please submit your results on Kaggle page with your nickname. Make sure your nickname appears on the public leaderboard. If you are better than the baseline, you can be above 60%. If you are better than the deep learning baseline, you can be above 80%.

[4] `https://www.kaggle.com/c/aist4010-spring2022-a1/data`

- Blackboard submissions. Please pack all files in one '.zip' file named as 'nickname_real name_SID' like 'lctest_Licheng ZONG_1155123456.zip'
  - A one page report describing your model architecture, loss function, hyper parameters, any other interesting detail led to your best result for the above competition. Please limit the report to **one page**.
  - **All** your source codes (including data-processing, training, prediction, etc.) as the format of '.ipynb' or '.py' in **one folder**.

6 Conclusion

Nicely done! Here is the end of Assignment 1 (Kaggle Part), and the beginning of a new world. As always, feel free to ask on Piazza if you have any questions. We are always here to help.

Good luck and enjoy the challenge!