Project #3: Training Networks with Tensorflow and Keras

Amir Sadovnik
COSC 424/525: Deep Learning (Spring 2023)

---

1 Overview

In this project you will be building a face attribute classifier for images. You will be experimenting with different architectures and measuring your results on each.

2 Dataset

You will be using a modified version of the FairFace dataset (https://github.com/joojs/fairface). This is a set of 86,744 training face images and 10,954 validation face images. In order to decrease the training time I converted all images to gray scale and resized them to 32×32. Each face has 3 different attributes which can be used for a classification task: race, gender, and age. All files can be found in the zip file on Canvas. The *train* folder contains the training images and the *fairface_label_train.csv* file contains all the label. There is a similar folder and file for the validation set.

As the three different attributes have a different number of possible values, your final layers for each classifier will vary. For each of the networks below please attempt to classify 2 of the attributes (you can choose which).

3 Network Details

3.1 Task 1: Fully Connected Neural Network

1. Build a feed forward neural network with the following specifications (Test on two different tasks):

    - Hidden layer 1: 1024 neurons with hyperbolic tangent activation function in each neuron.

- Hidden layer 2: 512 neurons, with sigmoid activation function in each of the neuron.
- 100 neurons, with rectified linear activation function in each of the neuron.
- Output layer: n (depending on the task) neurons representing the n classes, using the softmax activation function.

2. Using Min-Max scaling to scale the training dataset and using the same Min and Max values from the training set scale the test dataset ($\frac{X-X_{min}}{X_{max}-X_{min}}$).

3. Using mini-batch gradient descent to optimize the loss function: "categorical cross-entropy" on the training dataset. Please record the loss value for each of the epochs and create an epoch-loss plot and an accuracy-loss plot for both the training and validation set.

4. Report the following:

   - Final classification accuracy.
   - The n-class confusion matrix.

3.2 Task 2: Small Convolutional Neural Network

1. Build a convolutional neural network with the following specifications (Test on two different tasks):

   - Convolution layer having 40 feature detectors, with kernel size 5 x 5, and ReLU as the activation function, with stride 1 and no-padding.
   - A max-pooling layer with pool size 2x2.
   - Fully connected layer with 100 neurons, and ReLU as the activation function.
   - Output layer: n (depending on the task) neurons representing the n classes, using the softmax activation function. function for each of the 10 neurons.

2. Using Min-Max scaling to scale the training dataset and using the same Min and Max values from the training set scale the test dataset ($\frac{X-X_{min}}{X_{max}-X_{min}}$).

3. Using mini-batch gradient descent to optimize the loss function: "categorical cross-entropy" on the training dataset. Please record the loss value for each of the epochs and create an epoch-loss plot and an accuracy-loss plot for both the training and validation set.

4. Report the following:

   - Final classification accuracy.
   - The n-class confusion matrix.

3.3 Task 3: Your own Convolutional Neural Network

1. Build another convolutional neural network, where you choose all the parameters to see if you can get a higher accuracy.
2. Using Min-Max scaling to scale the training dataset and using the same Min and Max values from the training set scale the test dataset ($\frac{X-X_{min}}{X_{max}-X_{min}}$).
3. Using mini-batch gradient descent to optimize the loss function: "categorical cross-entropy" on the training dataset. Please record the loss value for each of the epochs and create an epoch-loss plot and an accuracy-loss plot for both the training and validation set.
4. Report the following:
   - Final classification accuracy.
   - The n-class confusion matrix.

3.4 Task 4: Your own Convolutional Neural Network on both Tasks Simultaneously

1. Build another convolutional neural network, where you try and classify both tasks with a single network. After your flatten layer have two more fully connected layers for each "branch". Note that in order to do so you will not be able to use the Sequential model.
2. Using Min-Max scaling to scale the training dataset and using the same Min and Max values from the training set scale the test dataset ($\frac{X-X_{min}}{X_{max}-X_{min}}$).
3. Using mini-batch gradient descent to optimize the loss function: "categorical cross-entropy" on the training dataset. Please record the loss value for each of the epochs and create an epoch-loss plot and an accuracy-loss plot for both the training and validation set.
4. Report the following:
   - Final classification accuracy.
   - The n-class confusion matrix.

3.5 Task 5: Variational Auto Encoder (COSC 525 only)

1. Build a variational autoencoder with the following specifications (in this one you have a little more flexibility):
   - Should have at least two convolution layers in the encoder and 2 deconvolution layers in the decoder.
   - Latent dimension should be at least 5.
   - Loss should be either MSE or binary cross entropy.

2. Using Min-Max scaling to scale the training dataset and using the same Min and Max values from the training set scale the test dataset ($\frac{X-X_{min}}{X_{max}-X_{min}}$).

3. Using mini-batch gradient descent to optimize the loss function on the training dataset. Please record the loss value for each of the epochs and create an epoch-loss plot and an accuracy-loss plot for both the training and validation set.

4. Qualitatively evaluate your model by generating a set of faces by randomly choosing 10 latent vectors and presenting the resulting images

4 Additional Information

1. In order to create a dataset you will need to read in the csv file and images. Using the *pandas* and/or *pillow* libraries could be useful for those tasks.

2. You may also find the *sklearn* library useful for doing one-hot encoding and getting the confusion matrix.

3. Since the dataset is pretty large I recommend only loading the first n rows from the CVS while debugging. Once you have everything working you can scale up to the full dataset to get the actual accuracy.

4. It would be useful to organize each task as a function, where the function returns the correct model. Training can then happen in a similar manner (VAE may be an exception). This function can take different parameters such as learning rate, learning rate decay, momentum, to allow you to test different options.

5. In order to draw plots for loss and accuracy for both training and validation data I recommend using TensorBoard. Take a look here for a tutorial. Here is a link showing how to use it in a Jupyter Notebook

6. Exact learning parameters are not mentioned. You will need to select your own learning rate, momentum etc.

7. Please submit a Jupyter Notebook. Submit both the ipynb file in addition to an html file version. Make sure to show the output for all tasks.

5 Report

Your Jupyter notebook should include all the following (use markup for text portions) :

1. A short introduction to the problem.

2. Introduction to each network (including the ones you designed).

3. Results from each network for the two tasks. This should include the loss plots and the evaluation criteria.

4. Conclusion - what did you observe when you ran these experiments?
5. How to run your code.

6 Submission

Please submit thhe html file and the ipynb file seperately.