Artificial Intelligence Spring 2019
Homework 5: Deep Learning

PROGRAMMING

The objective of this homework is to build a hand gesture classifier for sign language. We will be using Google Colaboratory to train our model (set up instructions at the end). The dataset will be in the form of csv files, where each row represents one image and its true label.

We have provided the skeleton code to read in the training and testing datasets. Before you begin coding, go through the provided code and try to figure out what each function is responsible for. Most of the functionalities are implemented in the **SignLanguage** class. Below are brief descriptions of each function and what is expected of you.

- **create_model**(self): You will generate a keras sequential model here. Make sure to set the self.model variable with your compiled sequential model.
- **prepare_data**(self, images, labels): Mainly this splits the data into training and validation sets. You may choose to normalize or downsample your data here, as well.
- **train**(self, batch_size, epochs, verbose): This method invokes the training of the model. Make sure to return the generated history object. Your model will be trained for a max of 50 epochs during grading. Make sure you are using the input parameters (batch_size, epochs, verbose)
- **predict**(self, data): This method will be invoked with the test images. Make sure to downsample/resize the test images the same way as the training images, and return a list of predictions.
- **visualize_data**(self, data): Nothing to do here. This is solely to help you visualize the type of data you are dealing with.
- **visualize_accuracy**(self, history): Nothing to do here. It plots out the accuracy improvement over training time.

Here are a few guides that may help you get started

- Keras Guide: Getting started with the Keras Sequential model
- Introduction to Convolutional Neural Networks (CNNs)
- A practical guide to CNNs
- Check also the class slides on deep learning in the Lectures folder.

A few points to note:

- We will train each model for 30 epochs to ensure convergence. However, you are free to tune the batch_size hyperparameter.
- We require a train-validation split, but you are free to choose the dataset split ratio.

The following questions might help you better approach CNNs and Keras:

1. What is one-hot encoding? Why is this important and how do you implement it in keras?
2. What is dropout and how does it help overfitting?
3. How does ReLU differ from the sigmoid activation function?
4. Why is the softmax function necessary in the output layer?
5. This is a more practical calculation. Consider the following convolution network:
    a. Input image dimensions = 100x100x1
    b. Convolution layer with filters=16 and kernel_size=(5,5)
    c. MaxPooling layer with pool_size = (2,2)

    What are the dimensions of the outputs of the convolution and max pooling layers?

You will submit a README.txt or pdf file that should contain the following:

- Your name and UNI
- 1-2 sentence answers to the questions above.
- A very brief explanation of your architecture choices, workflow or any relevant information about your model.

1. Test-Run Your Code

Before you submit, make sure it works! Go to Runtime > “Restart and run all…”. This will restart your kernel, clearing any set variables and run all cells again. This is to make sure your results are reproducible and do not depend on an overwritten variable!

2. Grading Submissions

Your model will be tested on the test set using the grading script as given in the skeleton code. We will only be using your **SignLanguage** class. *Please make sure not to edit the grading script portion of the notebook to avoid grading issues*.

Any model that achieves 90% accuracy in the test set will receive a full score. The scoring algorithm is tentatively $grade = 100 * min(90, accuracy)/90$. Any changes to the grading scheme will be updated here.

To avoid unlucky optimizations, we will train your model twice and take the maximum accuracy on the test set as your model accuracy. Extra credit will be given to the top models in the class.

3. Submission

When you are ready to submit your file, run the notebook using Runtime > “Restart and run all…”. Once you have your output, go to File > Download .ipynb and File > Download .py. Zip both the .ipynb and the .py file along with your README.txt or README.pdf as **hw5_myUNI.zip** and submit on Courseworks.

Make sure to include your uni and name in the first cell of your notebook!

Setting up Google Colaboratory

1) Go to https://colab.research.google.com and sign in using your LionMail/gmail account.
2) In the pop up window, select “UPLOAD” and upload the sign_language.ipynb file we have provided. (If this is not your first time, then it should automatically appear in the “RECENT” tab)
3) Once the notebook is open, you will need to set up the runtime to use GPU for training. To set up the GPU, go to Runtime > Change Runtime Type and fill in the following values
   Runtime type: Python 3
   Hardware accelerator: GPU
   Omit code cell output: False (Uncheck)
4) Next, upload your data files to Colaboratory. On the left hand panel, go to the Files tab. Click on “Upload”, and select your train.csv and test.csv files.
5) You should be set up and ready to go! The main advantage of Google Colaboratory is that all your packages should come pre-installed (especially tensorflow!).