Assignment 3

Part 1. Data Collection [5 pt]

Due Monday January 28th 11:59pm. No late submissions will be accepted.

So far, we have worked with data sets that have been collected, cleaned, and curated by Machine Learning researchers and practitioners. Datasets like MNIST and CIFAR are often used as toy examples, both by students and by researchers testing a new machine learning model.

In the real world, getting a clean data set is never that easy. Over half the battle of applying machine learning is finding, gathering, cleaning, and formatting your data set.

The purpose of this assignment is to help you gain experience gathering your own data set, and understand the challenges involved in the data cleaning process.

American Sign Language

American Sign Language (ASL) is a complete, complex language that employs signs made by moving the hands combined with facial expressions and postures of the body. It is the primary language of many North Americans who are deaf and is one of several communication options used by people who are deaf or hard-of-hearing.

The hand gestures representing English alphabets are shown below. This assignment focuses on classifying a subset of these hand gesture images using Convolutional Neural Networks. Specifically, given an image of a hand showing one of the letters A-I, we want to detect which letter is being represented.

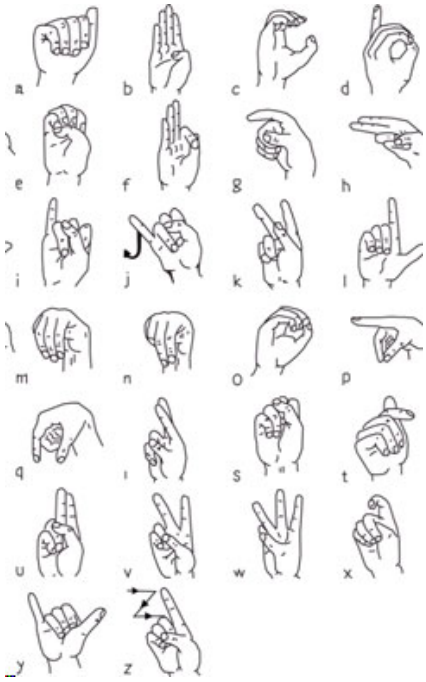

Figure 1: from https://www.nidcd.nih.gov/health/american-sign-language

Generating Data

We will produce the images required for this assignment by ourselves. Each student will collect, clean and submit three images each of Americal Sign Language gestures for letters A - I (total of 27 images)

Steps involved in Data Collection

1. Familiarize with American Sign Language Gestures for letters from A - I (9 letters)
2. Ask your friend to take **three** pictures at slightly different orientation for each letter gesture using your mobile phone
   - Ensure adequate lighting while you are capturing the images.
   - Use Right hand to create gestures (for consistency)
   - Keep your right hand fairly apart from your body and any other obstructions
3. Transfer the images to your laptop for cleaning

Cleaning Data

To make the task simpler, we will standardize the images. We will make sure that all our images are of the same size (224 x 224 pixels RGB), and have the hand in the center of the cropped regions.

You may use the following applications to crop and resize your images:

Mac

- Use `Preview`:
  - Hold down CMD + Shift will keep square aspect ratio while selecting the hand area
  - Resize to 224x224 pixels

Windows 10

- Use `Photos` app to edit and crop the image and keep the aspect ratio as Square
- Use `Paint` to resize the image to the final image size of 224x224 pixels

Linux

- You can use GIMP, or imagemagick, or other tools of your choosing

You may also use online tools such as http://picresize.com

All the above steps are indicative only. You need not follow these steps but following these will ensure that you churn out good quality dataset. You will be judged based on the quality of the images alone.

Accepted Images

Images will be accepted and graded based on below criteria

1. Final Image size 224x224 pixels (RGB)
2. File format is .jpg
3. Hand approximately centered on the frame
4. The hand is not obscured
5. The hand should not be cut off
6. Follow the ASL gestures posted earlier

Submission

Due Monday midnight. No late submissions will be accepted.

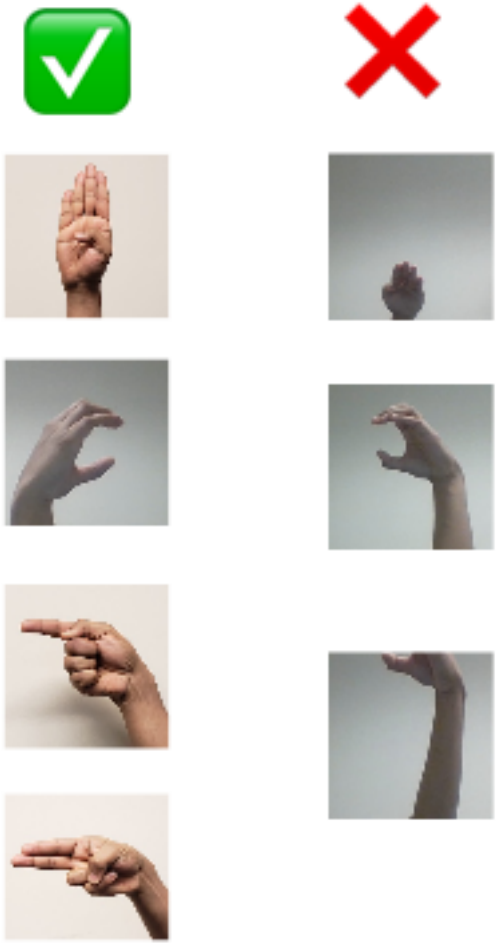

Figure 2: Acceptable Images

1. Individual Image file names should follow the convention of `student-number_Alphabet_file-no.jpg` (e.g. `100343434_A_1.jpg`)
2. Zip all the images together and name it with the following convention: `last-name_student-number.zip` (e.g. `last-name_100343434.zip`)
3. Upload the zipped folder in Quercus.

We will be anonymizing and combining everyone's images. We will announce when the combined data set will be available for download.

Part 2. Building a CNN [35 pt]

Due Sunday February 3rd 9pm

For this assignment, we are not going to give you any starter code, with the exception of part 5. You are welcome to use any code from previous assignments, lectures and tutorials. You should also write your own code.

Please do not use any other code that are not from the course and not written by you.

Ensure proper comments are included in the code. It is your responsibility to showcase that you understand what you write.

Submit your code and writeup as a PDF file. An export of a Jupyter Notebook is preferred.

1. Data Splitting [1 pt]

Split the data into training, validation, and test sets. Justify your choice.

2. Convolutional Network [5 pt]

Build a convolutional neural network model that takes the (224x224 RGB) image as input, and predicts the letter. Your model should be a subclass of `nn.Module`. Explain your choice of neural network architecture: how many layers did you choose? What types of layers did you use? Were they fully-connected or convolutional? What about other decisions like pooling layers, activation functions, number of channels / hidden units.

3. Training [5 pt]

Train your network. Plot the training curve.

Make sure that you are checkpointing frequently!

4. Hyper-parameter search [8 pt]

Part (a) [1pt]

List 3 hyperparameters that you think are most worth tuning.

Part (b) [4 pt]

Tune the hyperparameters you listed in Part (a), trying as many values as you need to until you feel satisfied that you are getting a good model. Plot the training curve of at least 4 different hyperparameter settings.

Part (c) [1 pt]

Choose the best model out of all the ones that you have trained. Justify your choice.

Part (d) [2 pt]

Report the test accuracy of your best model. You should only do this step once.

5. Transfer Learning [16 pt]

For many image classification tasks, it is generally not a good idea to train a very large Deep Neural Network model from scratch due to the enormous compute requirements and lack of sufficient amounts of training data. One of the better option is to try using an existing model that performs a similar task to the one you need to solve. This method of utilizing a pre-trained network for other similar tasks is broadly termed "Transfer Learning". In this assignment, we will use Transfer Learning to extract features from the hand gesture images. Then, train a smaller network to use these features as input and classify the hand gestures.

As you have learned from the CNN lecture, Convolution layers extract various features from the images which get utilized by the fully connected layers for correct classification. AlexNet architecture played a pivotal role in establishing Deep Neural Nets as a go-to tool for Image classification problems and we will use an imagenet pre-trained AlexNet model to extract features in this assignment.

AlexNet Paper

Part (a) [4 pt]

The code to load the AlexNet network weights is provided to you in `a3code.py`.

In the file `a3code.py`, a class `AlexNetFeatures` is provided to you. When you instantiate `AlexNetFeatures` class, the model will load pre-trained weights.

```
from a3code import AlexNetFeatures
myfeature_model = AlexNetFeatures() #loads pre-trained weights
```

It will download the weights automatically from Torch model zoo for the first time. The class expects image tensors having dimension Lx3x224x224 as inputs and it will output Lx256x6x6 tensors. (L = batch size).

Compute the AlexNet features for each of your training, validation, and test data. Save those values. You will need to load them later.

Part (b) [5 pt]

Build a convolutional neural network model that takes **as input** these AlexNet features, and makes a prediction. Your model should be a subclass of `nn.Module`.

Explain your choice of neural network architecture: how many layers did you choose? What types of layers did you use: fully-connected or convolutional? What about other decisions like pooling layers, activation functions, number of channels / hidden units in each layer?

Here's an example of how your model may be called:

```
# img = ... a PyTorch tensor with shape [1,224,224,3] containing a hand image ...
features = myfeature_model(img)
output = model(features)
prob = F.softmax(output)
```

Part (c) [5 pt]

Train your new network, including any hyperparameter tuning. Plot the training curve of your best model.

Part (d) [2 pt]

Report the test accuracy of your best model. How does the test accuracy compare to part 4(d)?