**1. Final training results**

I trained the SVM model from deliverable 2 on half of the available data using a VM provided by John. As expected, increasing the training dataset decreased the model accuracy slightly (it dropped to 94%). However, upon testing the model with photos of my own hand, I concluded that it wasn’t performing well enough and was most likely overfitted. In particular, the model consistently misclassified close to all ASL letters I gave it, even after my photos were edited using OpenCV to resemble the training data.

The SVM approach assumes that HOG is the best classification feature for this problem. Because the SVM model was underperforming, I decided to switch to a convolutional neural network, as proposed in my first deliverable.

In summary, I preprocessed the images by normalizing their RGB values, formatting them as a 3D tensor, and feeding them to a Pytorch Dataloader. I then constructed a CNN with one convolutional layer, a maxpooling layer, and two fully-connected layers.

To convert the labels into values an SVM could interpret, I used scikit-learn’s LabelEncoder to transform them into categorical numerical values. Finally, I used scikit-learn to perform a 80-20 split on the pre-processed data and obtain my training and validation datasets.

Moving forward, the number of layers, input and output channel sizes of each layer, and kernel sizes all can be varied. However, given the limited time for this project and last-minute model change, I did not experiment with different sizes extensively.

**2. Final demonstration proposal**

I plan on integrating my model into a web app. I want to use a platform that allows me to build this app with only Python to keep things simple. Based on online posts, I will use Flask’s framework and Heroku’s platform to build and host my app. Even though I don’t have experience with either technologies, there are a lot of online resources I can refer to.

I anticipate that the biggest challenge in building my app will be creating the architecture to accept user input. To begin, I plan on letting the user upload a photo of an ASL letter to classify. For simplicity, this photo would need to be cropped, but I can later implement a sliding window approach to locate the ASL letter. If this works out, I can extend the web app by allowing user input directly from the laptop’s webcam via the click of a button. However, I’m unsure of how to dynamically display the webcam’s input until the user decides to take this photo, so I think implementing this feature will be the most challenging part. A rough sketch of the last stage of my web app can be found below. The red square outlining the letter can be removed if my sliding window approach works.

