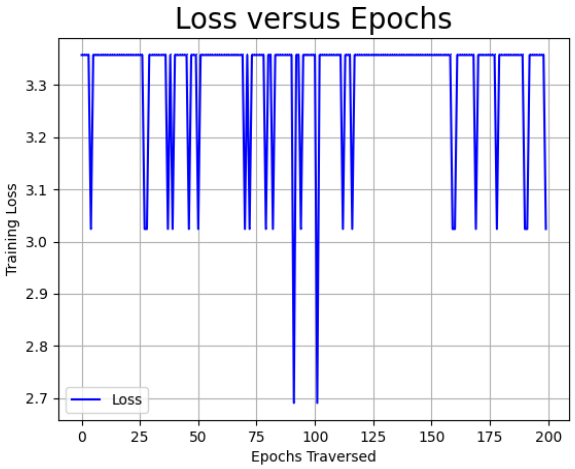
Final Training Results Compare your final results to your preliminary ones (from the previous deliverable). Have you changed anything to your model since the previous deliverable? If so, how have your changes improved the results?

Our current situation is similar to the past deliverable: we are still working to fix a bug that is preventing our model from learning. We've tried the following changes to solve this problem:

- Implemented hyperparameter optimisation: once training bug is fixed and model is able to learn, we should have no delay in finding the optimal hyperparameter combination
- Tried running our code with different dataset and our dataset with different code
- Modified regularization transforms and normalization
- Changed size of test and validation sets to allow for faster training
- Investigated structure of layers (nn.Sequential vs. nn.ModuleList)
- Changed loss function, optimizer, learning rate, architecture
    - Moved position of optimizer.zerograd()
    - Used torchsummary to confirm reasonable architecture

Now, focus on your final results. Once again, present a detailed analysis of your results, provide graphs as appropriate. Analysis requirements differ in every field, but must report at least one concrete metric relevant to the field in which you are working.
Here are some examples: Confusion matrix and accuracy/precision-recall/logistic loss

We decided to use logistical loss and validation accuracy as our measure of loss: our weights were optimized based on the logistical loss, and our hyperparameters were tuned based on test versus validation accuracies. Below is a plot of accuracy of our model using the train sets of many epochs. As seen the loss does not seem to improve over time, it seems to reach a local minimum that is not optimal. It does not seem able to get out of that minimum, even when performing gradient descent over many iterations. The lowest it is able to go is down to about 2.7 training loss. Thus, for some reason, the model is not even able to overfit on a very small set.



Final demonstration proposal:

Our final product comprises a web application utilizing HTML and CSS for the frontend, connected to the backend through JavaScript and the Flask framework. This integration approach was informed by the online tutorial from deliverable 3, leveraging our prior experience with HTML, CSS, and JavaScript.

Our selection of HTML, CSS, JavaScript, and the Flask framework stems from our team's experience with these technologies. It would ensure a smooth and efficient development process but also contributed to the overall reliability and proficiency of the final product.

The user must first activate their webcams, capture sign language images within the web app, experience real-time translation, and receive immediate results.