ASL Recognition

Ertuğ Ümsür, Ishan Porwal

Background:

American Sign Language is the primary language of many North Americans who are deaf and hard of hearing. In 1817, the first American school for the deaf was opened by European experts who migrated to the US and they planted the first seeds for the development of the American Sign Language (ASL). The first sign language used in the US was a mix of French Sign Language and local signs. Throughout the 19th century, ASL was standardized and today more than half a million people use it as their primary way of communication.

American Sign Language is the only way of communication for many Americans and Canadians, and its usage is essential for the incorporation of people who are deaf and hard of hearing into society. Even though this constitutes incredible importance for people using it as a way of communication, unfortunately, there are not enough non-deaf or non-hard-of-hearing speakers of ASL to allow the homogenous incorporation of its speakers into society. Thus, we need to develop tools to shatter the communication barrier between this community and the overarching society, and Machine Learning tools possess incredible importance in solving this problem. The automation of the translation process allows ASL users to communicate even with individuals who have never been exposed to any sign language. Utilization of Machine Learning solutions promises great advances and today with our project, we seek to contribute to this essential process.

Scope:

Many machine-learning solutions incorporate the whole extent of American Sign Language into their systems. Some of these include AlexNet and GenASL (Amazon's sign language avatar). However, ASL is a language that contains many complex relations of words and face mimics and it is impossible for us to simulate ASL to its full extent with our current resources. Thus, our project will focus on translating individual signs of the ASL. These will include simple numbers and letters, except j and z since these signs require movement to perform. For this purpose, we used the Modified National Institute Standards and Technology's (MNIST) sign language database. To combat the uniformity of our dataset and introduce diversity, we utilized data augmentation

Concepts Utilized:

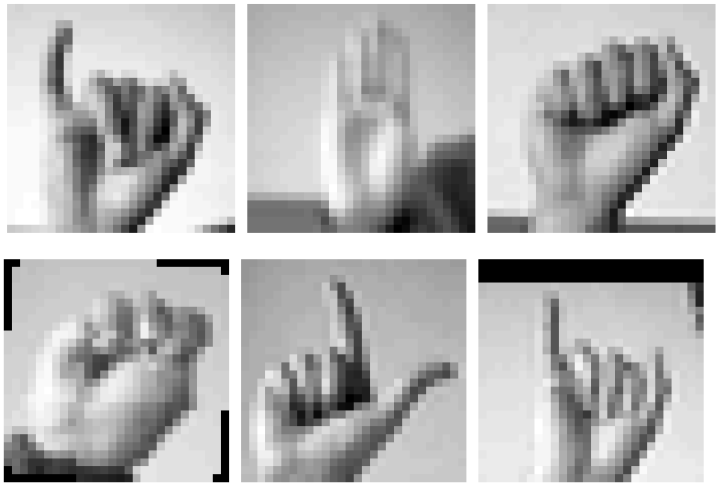
In this project, we used all the concepts we learned throughout the semester, including activation functions, deep neural networks, dropout functions, etc. However, our main focus was on using Convolutional Neural Networks (CNNs) and comparing the effectiveness of different implementation models that include CNNs in translating ASL. In total, we had three different implementation models: one with a simple Neural Network with linear relations and ReLu activation functions, one with Convolution layers, ReLu activation functions, and multilayer linear network, and finally one with Batch Normalization included on top of the second model. We hypothesized that a simple Neural Network would reach the lowest

accuracy since it can not capture the complex relationships of the images' positional encodings. We thought the second model would be effective since Convolutional layers allow positional relations to be captured but it would be comparatively less effective than the last model. We hypothesized that the last model's Batch Normalization layers would reduce overfitting and allow better results for testing data.

Also, we used some data augmentation to better train our model and attempt to prevent it from overfitting to the original almost uniform dataset.

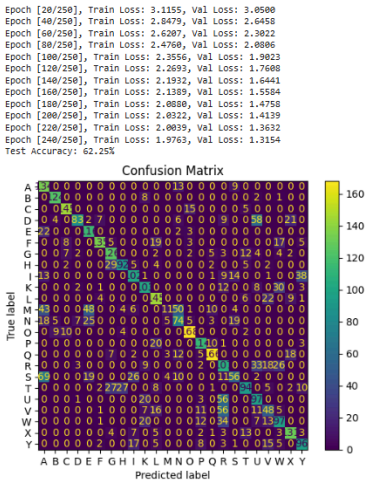
3 Different Models: SimpleNN vs CNN vs CNN with Batch Normalization

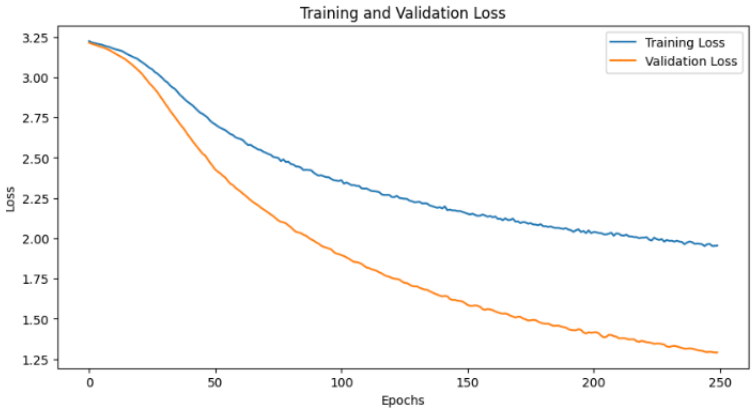
Before training our data, we preprocessed images for all of the models. Our preprocessing included randomized image crop, rotation, color jitter, and affine. All of these preprocesses had a 50% chance of being applied individually. We didn't apply any preprocesses to our testing or validation datasets. This is the reason the validation losses were much smaller than training losses in the learning curves.



All of the models used the same CrossEntropyLoss function and the optimizer for all models was identical. The Epoch size was kept constant at 250 Epochs for each model. With these settings here are the results of our model training:

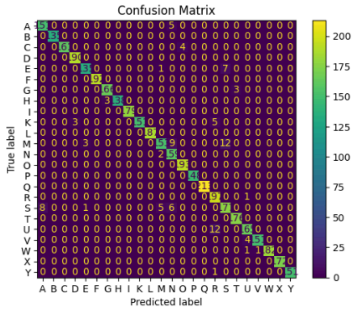
<u>For the SimpleNN model:</u>

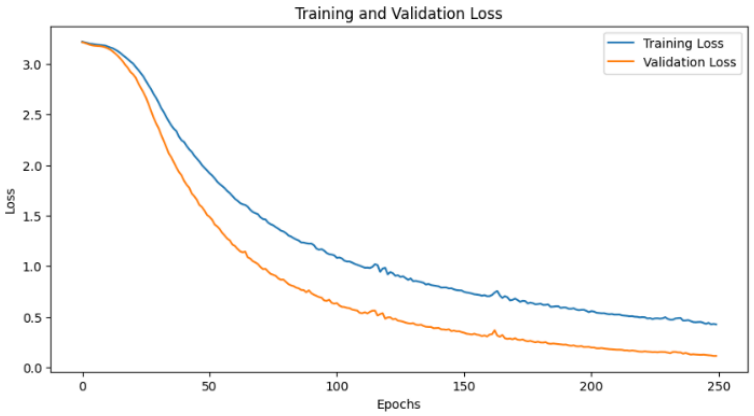




For the CNN model:

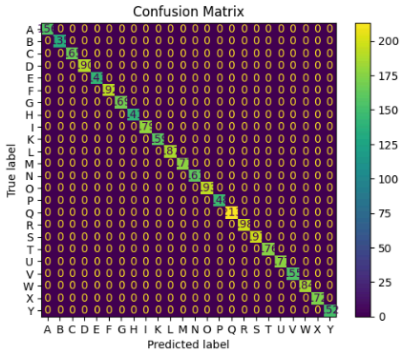
```
Epoch [20/250], Train Loss: 3.0209, Val Loss: 2.9175
Epoch [40/250], Train Loss: 2.2441, Val Loss: 1.8975
Epoch [60/250], Train Loss: 1.6925, Val Loss: 1.2128
Epoch [80/250], Train Loss: 1.3429, Val Loss: 0.8674
Epoch [100/250], Train Loss: 1.1062, Val Loss: 0.6284
Epoch [120/250], Train Loss: 0.9829, Val Loss: 0.4806
Epoch [140/250], Train Loss: 0.8046, Val Loss: 0.3871
Epoch [160/250], Train Loss: 0.7031, Val Loss: 0.3043
Epoch [180/250], Train Loss: 0.6244, Val Loss: 0.2535
Epoch [200/250], Train Loss: 0.5453, Val Loss: 0.2015
Epoch [220/250], Train Loss: 0.4935, Val Loss: 0.1551
Epoch [240/250], Train Loss: 0.4568, Val Loss: 0.1255
Test Accuracy: 97.65%
```

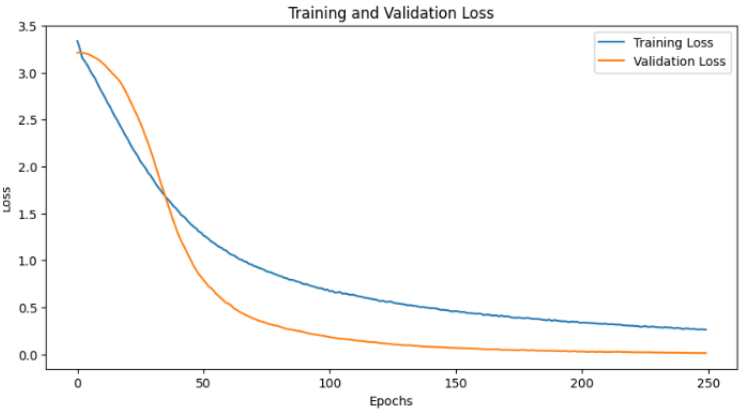




For the CNN with Batch Normalization Model:

```
Epoch [20/250], Train Loss: 2.3324, Val Loss: 2.8070
Epoch [40/250], Train Loss: 1.5566, Val Loss: 1.3498
Epoch [60/250], Train Loss: 1.1038, Val Loss: 0.5464
Epoch [80/250], Train Loss: 0.8478, Val Loss: 0.3047
Epoch [100/250], Train Loss: 0.6897, Val Loss: 0.1882
Epoch [120/250], Train Loss: 0.5764, Val Loss: 0.1230
Epoch [140/250], Train Loss: 0.4942, Val Loss: 0.0795
Epoch [160/250], Train Loss: 0.4334, Val Loss: 0.0591
Epoch [180/250], Train Loss: 0.3858, Val Loss: 0.0398
Epoch [200/250], Train Loss: 0.3360, Val Loss: 0.0284
Epoch [220/250], Train Loss: 0.3031, Val Loss: 0.0223
Epoch [240/250], Train Loss: 0.2755, Val Loss: 0.0150
Test Accuracy: 100.00%
```





The results show a clear difference between all three models and this supports our hypothesis. SimpleNN has a low accuracy of 62.25% and this is caused by the model's inability to capture positional relations in an image. Compared to the second most effective model, SimpleNN has a 35.4% accuracy difference which shows that CNN models are the most effective way of capturing image relations compared to simple neural networks. The learning curves are showing a similar picture as well. While both CNN models reach close to zero validation loss after 250 epochs, the SimpleNN validation loss only reaches 1.25.

The learning curves also show a relation between preprocessing and the models used. Even though all models produce higher losses for training data than validation data, the SimpleNN model has a wider gap between validation losses and training losses which means that SimpleNN can not capture the relations in the image as well as CNN models if the data are slightly adjusted using preprocessing.

Finally, confusion matrices show a similar relation between all three models and all of the results show that CNN with Batch Normalization model outperforms all three. Not only that but with 100% accuracy as observed by the confusion matrix and accuracy values, CNN

with Batch Normalization model has an incredible success rate in identifying ASL signs, at least trained and tested against our dataset.

Potential Improvements and Results:

While our results show the usefulness of CNN models for ASL identification and translation, especially when enhanced with Batch Normalization, there is room for further improvements and to expand our project. Here are some potential directions:

1. Including Dynamic Signs
    - Our current models exclude signs with movement like "J" and "Z" (and words) due to their motion requirements. Future iterations can integrate techniques like Recurrent Neural Networks (RNNs), allowing the model to understand and process changing frames. This enables the recognition of signs that involve movement.
2. Expanding the Dataset
    - The MNIST dataset is useful for recognition within the dataset itself, but its coverage is limited. Using diverse real-world data with different lighting conditions, and sign users of different skin tones would improve the model's robustness and generalizability.
3. Improved Preprocessing Techniques
    - While our preprocessing steps allowed us to reduce overfitting and have reliable results with real-world data, advanced techniques like Generative Adversarial Networks (GANs) could produce more realistic variations in training data.
4. Real-time implementation
    - One of our stretch goals was to create a real-time implementation. With a more diverse dataset, either self-assembled or accumulated through online scraping, or using transfer learning, we would like to reduce the overfitting in our training and implement a real-time recognition system with computer vision.

MNIST Dataset: https://www.kaggle.com/datasets/datamunge/sign-language-mnist