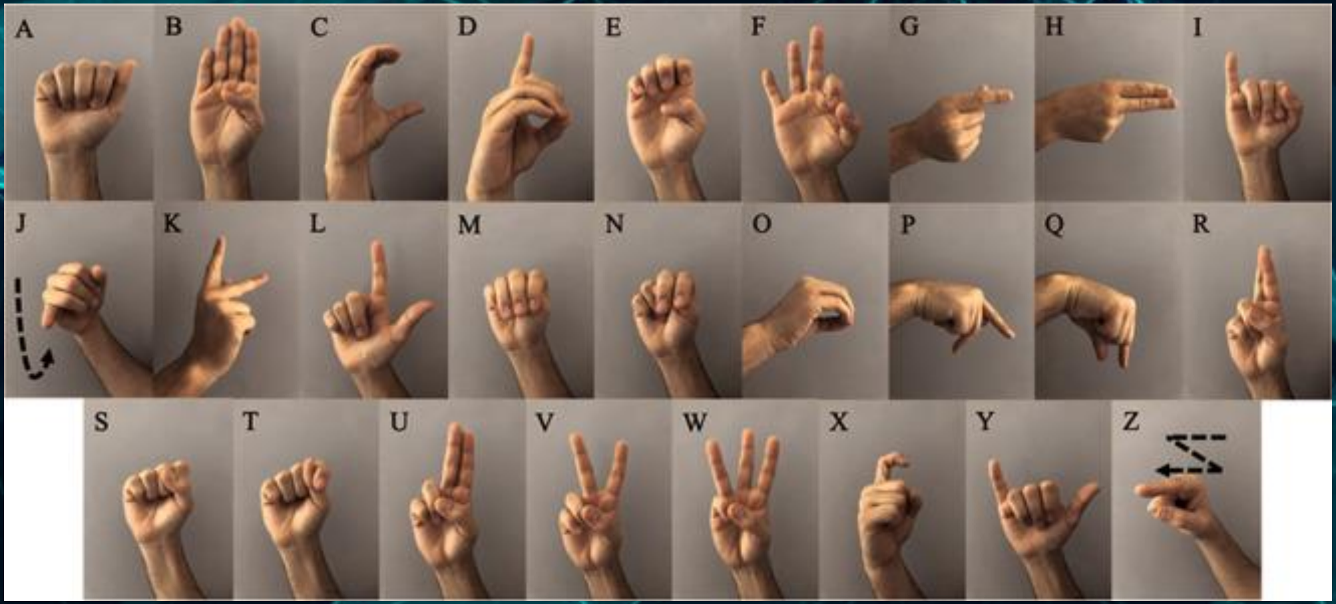
Project Proposal

Elective 4 | Deep Learning

DS4A

American Sign Language Image Classification using Convolutional Neural Network



I n t r o d u c t i o n

According to the National Institute on Deafness and Other Communication Disorders, the 200-year-old American Sign Language is a comprehensive, sophisticated language (of which letter gestures constitute only a minor part), but it remains the primary language for many deaf North Americans (NIDCD). Building a system that recognizes sign language will allow the deaf and hard of hearing to communicate more effectively with modern technologies.

Image classification has a great help to society, especially to people with disabilities. Building a system that recognizes sign language will allow the deaf and hard of hearing to communicate more effectively. Using Convolution Neural Network, a dataset of different images will be fed to the model for training and testing.

O b j e c t i v e

- To develop a model using Convolutional Neural Network that will classify what letter in the English Alphabet is associated with the respective Sign Language.
- To contribute knowledge for the development of Sign language recognition systems.
- To discuss the results obtained using (ResNet) Residual Neural Network.
- To provide an efficient and accurate way to convert sign language into text or voice has aids for the hearing impaired.

Scope and Limitation

A model that categorizes American sign language using various visuals in text or voice is being developed. We encounter many difficulties with our datasets and models when writing the code for our project. One epoch takes 1-2 hours to run, and if we continue it will take a month to exhaust all of our epochs.

Realted Works/ Related Project

•There are eight existing studies that are related to our study or project. One of the studies proposes using CNN for the recognition of gestures in Indian sign language.

•They used smartphone cameras in selfie mode and built a mobile application to detect the gestures. They obtained 92.88% accuracy.

• They used DenseNet, a type of CNN that utilizes dense connections between layers through Dense Blocks.

M e t h o d o l o g y

This chapter presents the methodology used for performing and analyzing the study. It explains the Dataset, Deep learning model CNN and ResNet. Additionally, it provides more details on a model for classifying the sign languages' alphabets.

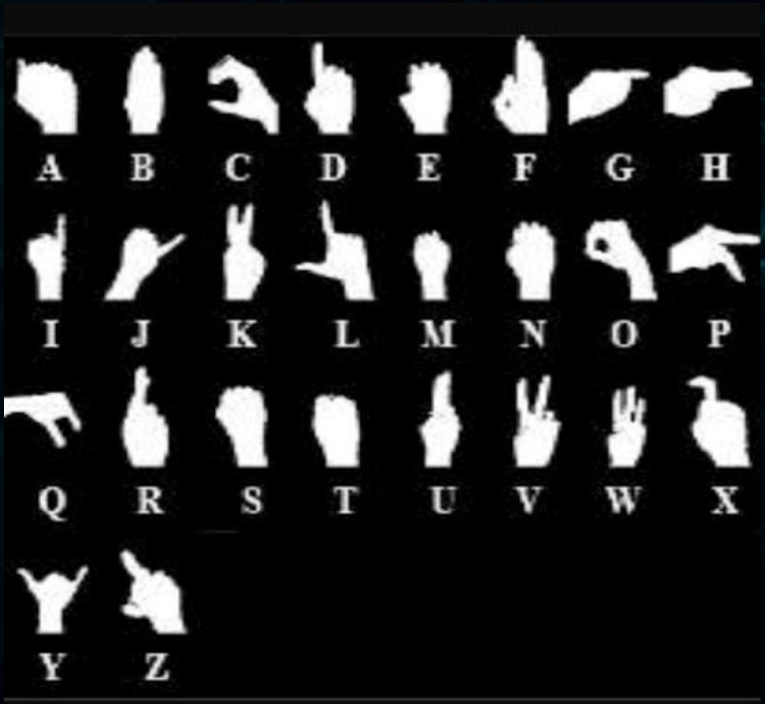
D a t a s e t

The dataset used by the researchers was obtained from Kaggle. It contains two folders namely asl_alphabet_test and asl_alphabet_train. Each folder contains images labeled by its respective english letter. It also includes images for “Space”, “del”, and “noting”.

* asl_alphabet_train folder contains 3000 images of each letter, including the three labels.
* asl_alphabet_test have 28 images.
* training data (27,455 cases)
* test data (7172 cases)

Deep Learning Model

* Computer Vision has many interesting applications ranging from industrial applications to social applications.
* It can recognize the hand symbols and predict the correct corresponding alphabet through sign language classification.



CNN

Convolutional Neural Networks, also known as CNNs, are a subset of artificial neural networks used in deep learning and are frequently employed for object and picture recognition and categorization.

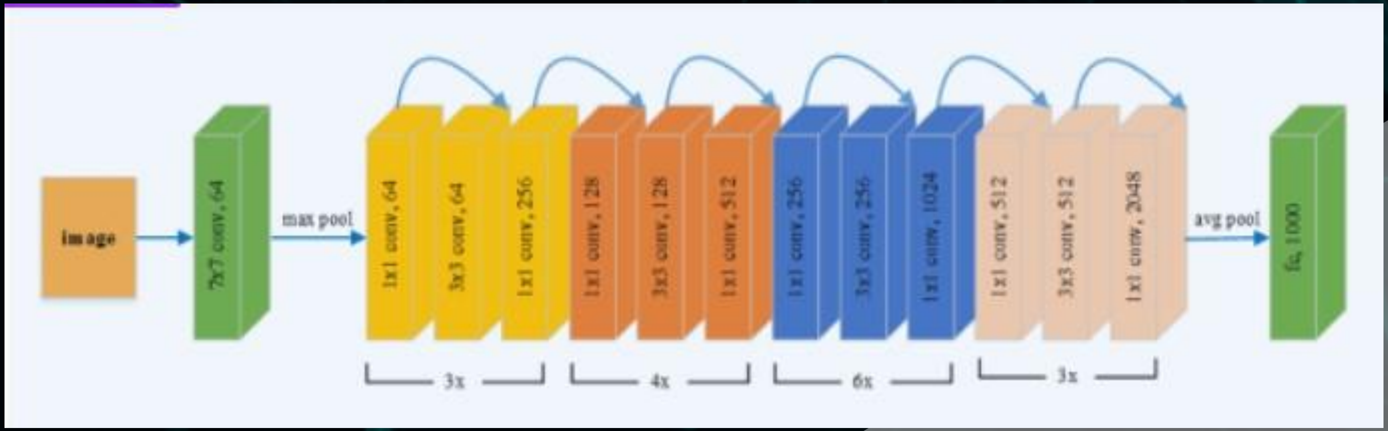
* speech recognition in natural language processing
* video analysis, and difficulties with image processing
* computer vision
* self-driving car obstacle detection

ResNet

A ResNet is a type of convolutional neural network, it was able to achieve very high accuracy in image classification tasks, such as the ImageNet dataset, by addressing the problem of vanishing gradients in very deep networks.

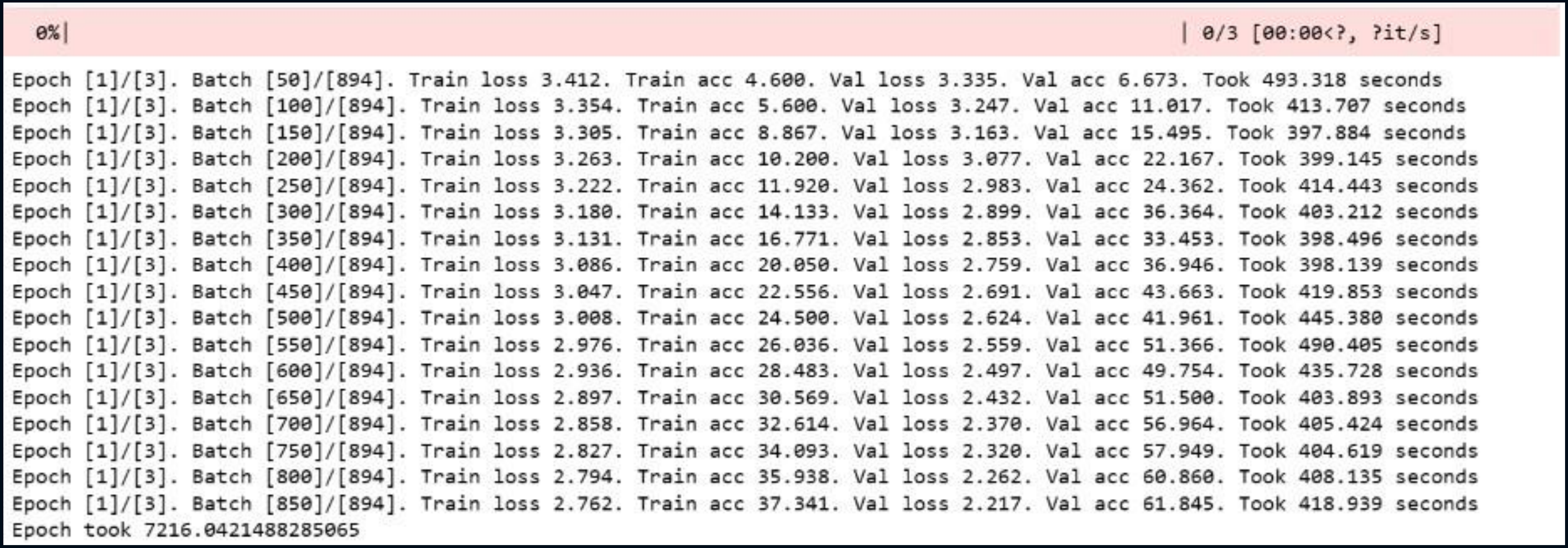
ResNet50

A ResNet50 is a deeper version of the original ResNet architecture, containing 50 layers, and was able to achieve even higher accuracy in image classification tasks, such as the ImageNet dataset. It is trained on a large dataset and the learned features can be used to train smaller datasets.



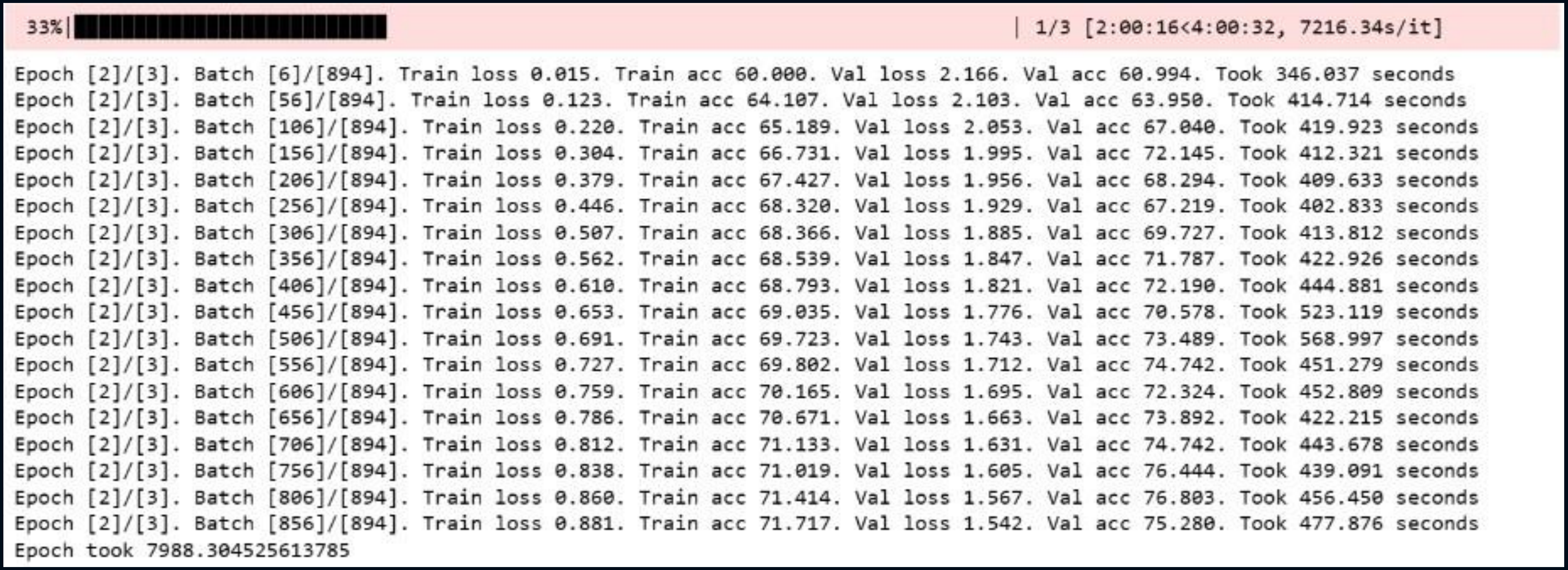
Summary

First Epoch



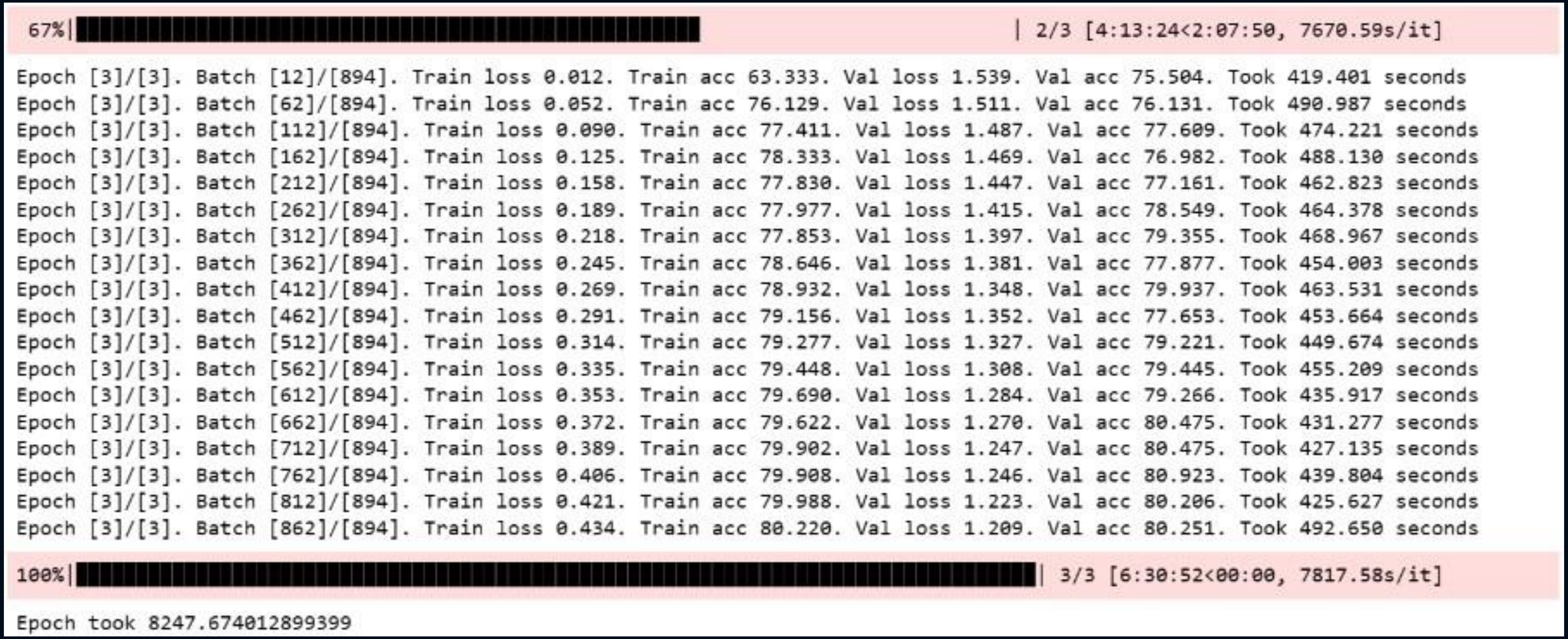
```
 0%|                                                                                       | 0/3 [00:00<?, ?it/s]

Epoch [1]/[3]. Batch [50]/[894]. Train loss 3.412. Train acc 4.600. Val loss 3.335. Val acc 6.673. Took 493.318 seconds
Epoch [1]/[3]. Batch [100]/[894]. Train loss 3.354. Train acc 5.600. Val loss 3.247. Val acc 11.017. Took 413.707 seconds
Epoch [1]/[3]. Batch [150]/[894]. Train loss 3.305. Train acc 8.867. Val loss 3.163. Val acc 15.495. Took 397.884 seconds
Epoch [1]/[3]. Batch [200]/[894]. Train loss 3.263. Train acc 10.200. Val loss 3.077. Val acc 22.167. Took 399.145 seconds
Epoch [1]/[3]. Batch [250]/[894]. Train loss 3.222. Train acc 11.920. Val loss 2.983. Val acc 24.362. Took 414.443 seconds
Epoch [1]/[3]. Batch [300]/[894]. Train loss 3.180. Train acc 14.133. Val loss 2.899. Val acc 36.364. Took 403.212 seconds
Epoch [1]/[3]. Batch [350]/[894]. Train loss 3.131. Train acc 16.771. Val loss 2.853. Val acc 33.453. Took 398.496 seconds
Epoch [1]/[3]. Batch [400]/[894]. Train loss 3.086. Train acc 20.050. Val loss 2.759. Val acc 36.946. Took 398.139 seconds
Epoch [1]/[3]. Batch [450]/[894]. Train loss 3.047. Train acc 22.556. Val loss 2.691. Val acc 43.663. Took 419.853 seconds
Epoch [1]/[3]. Batch [500]/[894]. Train loss 3.008. Train acc 24.500. Val loss 2.624. Val acc 41.961. Took 445.380 seconds
Epoch [1]/[3]. Batch [550]/[894]. Train loss 2.976. Train acc 26.036. Val loss 2.559. Val acc 51.366. Took 490.405 seconds
Epoch [1]/[3]. Batch [600]/[894]. Train loss 2.936. Train acc 28.483. Val loss 2.497. Val acc 49.754. Took 435.728 seconds
Epoch [1]/[3]. Batch [650]/[894]. Train loss 2.897. Train acc 30.569. Val loss 2.432. Val acc 51.500. Took 403.893 seconds
Epoch [1]/[3]. Batch [700]/[894]. Train loss 2.858. Train acc 32.614. Val loss 2.370. Val acc 56.964. Took 405.424 seconds
Epoch [1]/[3]. Batch [750]/[894]. Train loss 2.827. Train acc 34.093. Val loss 2.320. Val acc 57.949. Took 404.619 seconds
Epoch [1]/[3]. Batch [800]/[894]. Train loss 2.794. Train acc 35.938. Val loss 2.262. Val acc 60.860. Took 408.135 seconds
Epoch [1]/[3]. Batch [850]/[894]. Train loss 2.762. Train acc 37.341. Val loss 2.217. Val acc 61.845. Took 418.939 seconds
Epoch took 7216.0421488285065
```

Second Epoch



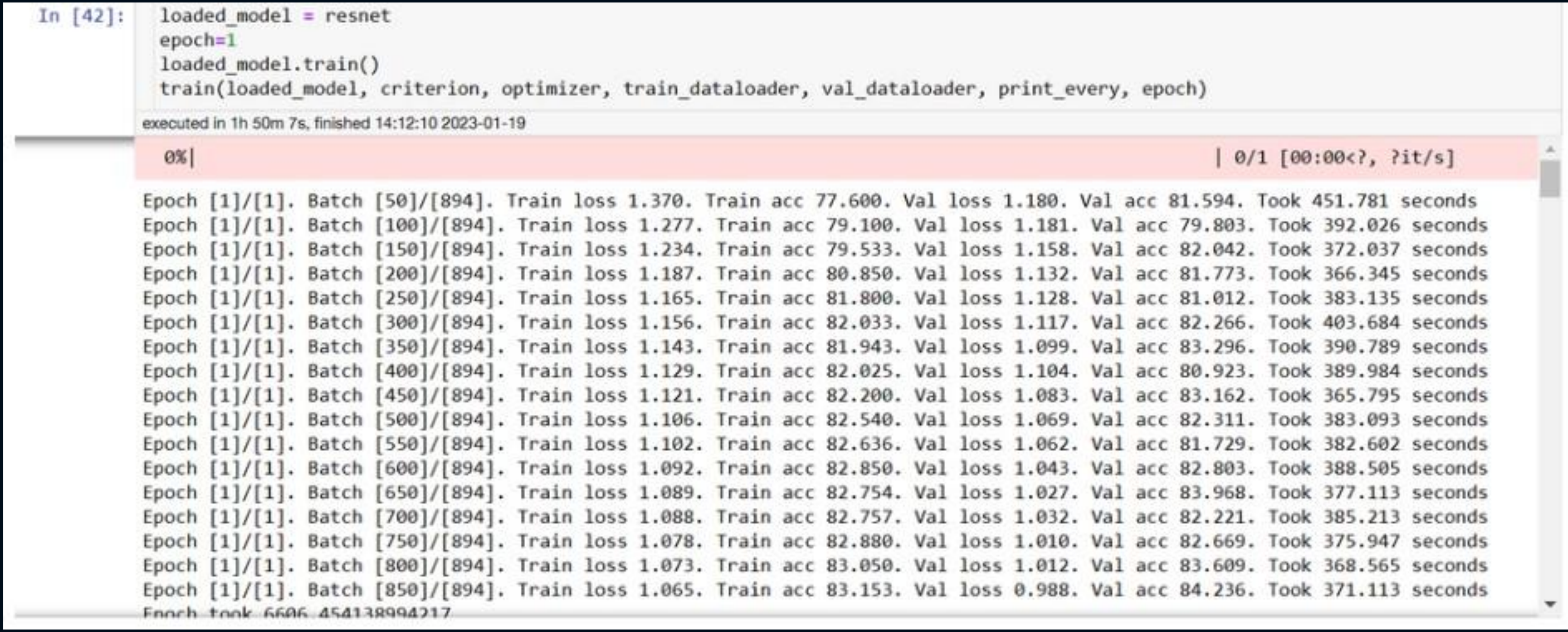
```
 33%|██████████████████████████                                          | 1/3 [2:00:16<4:00:32, 7216.34s/it]
Epoch [2]/[3]. Batch [6]/[894]. Train loss 0.015. Train acc 60.000. Val loss 2.166. Val acc 60.994. Took 346.037 seconds
Epoch [2]/[3]. Batch [56]/[894]. Train loss 0.123. Train acc 64.107. Val loss 2.103. Val acc 63.950. Took 414.714 seconds
Epoch [2]/[3]. Batch [106]/[894]. Train loss 0.220. Train acc 65.189. Val loss 2.053. Val acc 67.040. Took 419.923 seconds
Epoch [2]/[3]. Batch [156]/[894]. Train loss 0.304. Train acc 66.731. Val loss 1.995. Val acc 72.145. Took 412.321 seconds
Epoch [2]/[3]. Batch [206]/[894]. Train loss 0.379. Train acc 67.427. Val loss 1.956. Val acc 68.294. Took 409.633 seconds
Epoch [2]/[3]. Batch [256]/[894]. Train loss 0.446. Train acc 68.320. Val loss 1.929. Val acc 67.219. Took 402.833 seconds
Epoch [2]/[3]. Batch [306]/[894]. Train loss 0.507. Train acc 68.366. Val loss 1.885. Val acc 69.727. Took 413.812 seconds
Epoch [2]/[3]. Batch [356]/[894]. Train loss 0.562. Train acc 68.539. Val loss 1.847. Val acc 71.787. Took 422.926 seconds
Epoch [2]/[3]. Batch [406]/[894]. Train loss 0.610. Train acc 68.793. Val loss 1.821. Val acc 72.190. Took 444.881 seconds
Epoch [2]/[3]. Batch [456]/[894]. Train loss 0.653. Train acc 69.035. Val loss 1.776. Val acc 70.578. Took 523.119 seconds
Epoch [2]/[3]. Batch [506]/[894]. Train loss 0.691. Train acc 69.723. Val loss 1.743. Val acc 73.489. Took 568.997 seconds
Epoch [2]/[3]. Batch [556]/[894]. Train loss 0.727. Train acc 69.802. Val loss 1.712. Val acc 74.742. Took 451.279 seconds
Epoch [2]/[3]. Batch [606]/[894]. Train loss 0.759. Train acc 70.165. Val loss 1.695. Val acc 72.324. Took 452.809 seconds
Epoch [2]/[3]. Batch [656]/[894]. Train loss 0.786. Train acc 70.671. Val loss 1.663. Val acc 73.892. Took 422.215 seconds
Epoch [2]/[3]. Batch [706]/[894]. Train loss 0.812. Train acc 71.133. Val loss 1.631. Val acc 74.742. Took 443.678 seconds
Epoch [2]/[3]. Batch [756]/[894]. Train loss 0.838. Train acc 71.019. Val loss 1.605. Val acc 76.444. Took 439.091 seconds
Epoch [2]/[3]. Batch [806]/[894]. Train loss 0.860. Train acc 71.414. Val loss 1.567. Val acc 76.803. Took 456.450 seconds
Epoch [2]/[3]. Batch [856]/[894]. Train loss 0.881. Train acc 71.717. Val loss 1.542. Val acc 75.280. Took 477.876 seconds
Epoch took 7988.304525613785
```

Third Epoch



```
 67%|██████████████████████████████████████████                    | 2/3 [4:13:24<2:07:50, 7670.59s/it]
Epoch [3]/[3]. Batch [12]/[894]. Train loss 0.012. Train acc 63.333. Val loss 1.539. Val acc 75.504. Took 419.401 seconds
Epoch [3]/[3]. Batch [62]/[894]. Train loss 0.052. Train acc 76.129. Val loss 1.511. Val acc 76.131. Took 490.987 seconds
Epoch [3]/[3]. Batch [112]/[894]. Train loss 0.090. Train acc 77.411. Val loss 1.487. Val acc 77.609. Took 474.221 seconds
Epoch [3]/[3]. Batch [162]/[894]. Train loss 0.125. Train acc 78.333. Val loss 1.469. Val acc 76.982. Took 488.130 seconds
Epoch [3]/[3]. Batch [212]/[894]. Train loss 0.158. Train acc 77.830. Val loss 1.447. Val acc 77.161. Took 462.823 seconds
Epoch [3]/[3]. Batch [262]/[894]. Train loss 0.189. Train acc 77.977. Val loss 1.415. Val acc 78.549. Took 464.378 seconds
Epoch [3]/[3]. Batch [312]/[894]. Train loss 0.218. Train acc 77.853. Val loss 1.397. Val acc 79.355. Took 468.967 seconds
Epoch [3]/[3]. Batch [362]/[894]. Train loss 0.245. Train acc 78.646. Val loss 1.381. Val acc 77.877. Took 454.003 seconds
Epoch [3]/[3]. Batch [412]/[894]. Train loss 0.269. Train acc 78.932. Val loss 1.348. Val acc 79.937. Took 463.531 seconds
Epoch [3]/[3]. Batch [462]/[894]. Train loss 0.291. Train acc 79.156. Val loss 1.352. Val acc 77.653. Took 453.664 seconds
Epoch [3]/[3]. Batch [512]/[894]. Train loss 0.314. Train acc 79.277. Val loss 1.327. Val acc 79.221. Took 449.674 seconds
Epoch [3]/[3]. Batch [562]/[894]. Train loss 0.335. Train acc 79.448. Val loss 1.308. Val acc 79.445. Took 455.209 seconds
Epoch [3]/[3]. Batch [612]/[894]. Train loss 0.353. Train acc 79.690. Val loss 1.284. Val acc 79.266. Took 435.917 seconds
Epoch [3]/[3]. Batch [662]/[894]. Train loss 0.372. Train acc 79.622. Val loss 1.270. Val acc 80.475. Took 431.277 seconds
Epoch [3]/[3]. Batch [712]/[894]. Train loss 0.389. Train acc 79.902. Val loss 1.247. Val acc 80.475. Took 427.135 seconds
Epoch [3]/[3]. Batch [762]/[894]. Train loss 0.406. Train acc 79.908. Val loss 1.246. Val acc 80.923. Took 439.804 seconds
Epoch [3]/[3]. Batch [812]/[894]. Train loss 0.421. Train acc 79.988. Val loss 1.223. Val acc 80.206. Took 425.627 seconds
Epoch [3]/[3]. Batch [862]/[894]. Train loss 0.434. Train acc 80.220. Val loss 1.209. Val acc 80.251. Took 492.650 seconds
100%|████████████████████████████████████████████████████████████████| 3/3 [6:30:52<00:00, 7817.58s/it]
Epoch took 8247.674012899399
```

Fourth Epoch

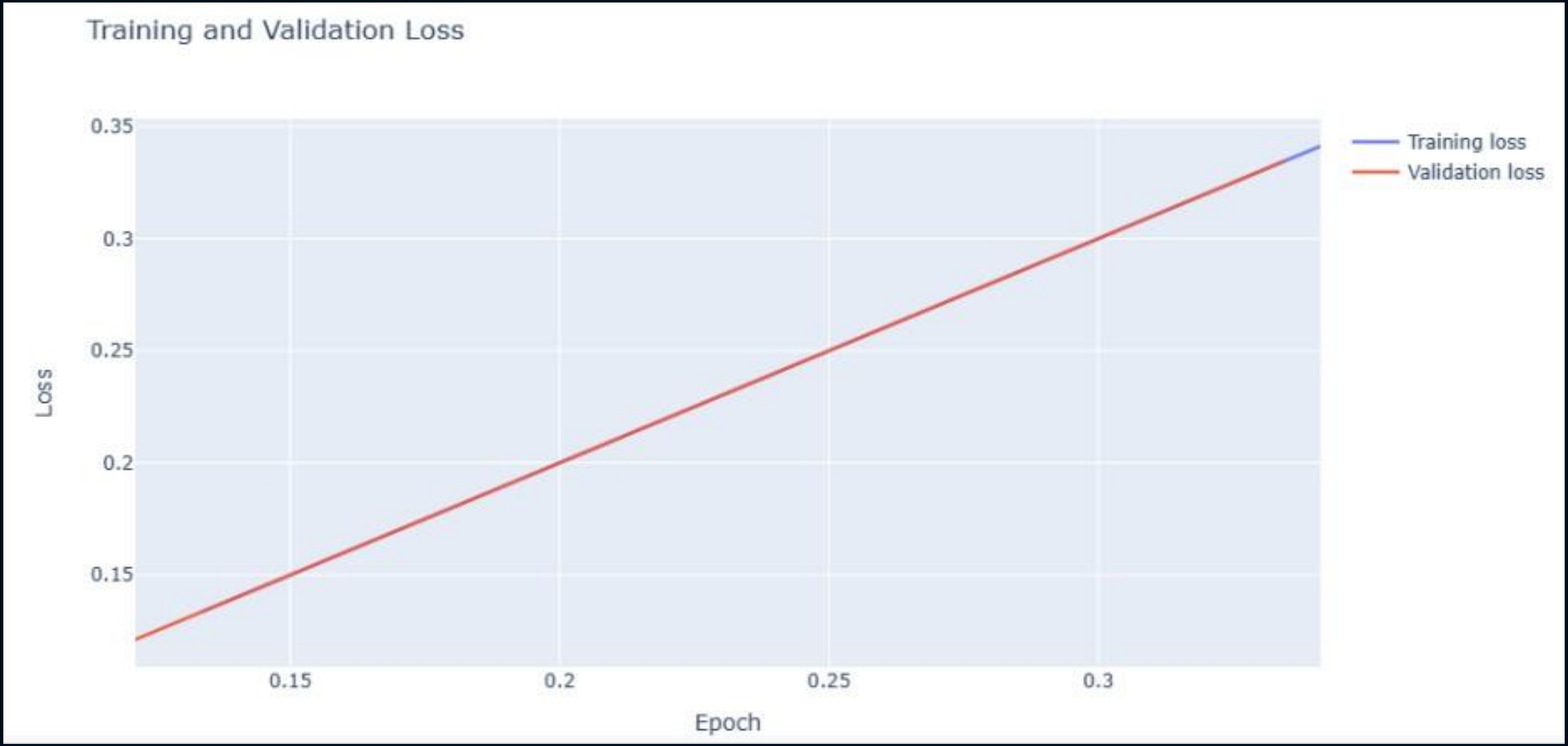


```
In [42]: loaded_model = resnet
         epoch=1
         loaded_model.train()
         train(loaded_model, criterion, optimizer, train_dataloader, val_dataloader, print_every, epoch)
```

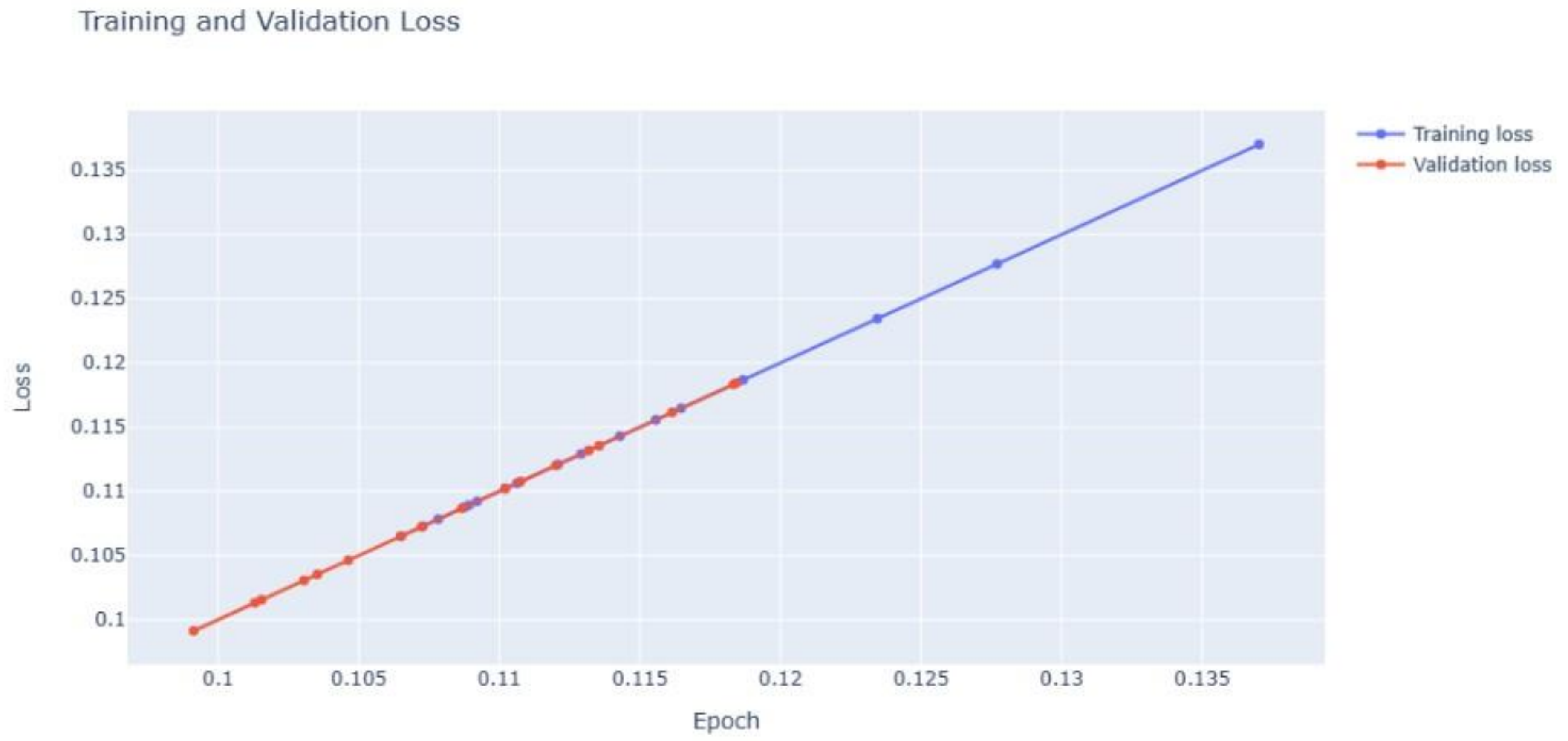
executed in 1h 50m 7s, finished 14:12:10 2023-01-19

```
0%|                                                                          | 0/1 [00:00<?, ?it/s]

Epoch [1]/[1]. Batch [50]/[894]. Train loss 1.370. Train acc 77.600. Val loss 1.180. Val acc 81.594. Took 451.781 seconds
Epoch [1]/[1]. Batch [100]/[894]. Train loss 1.277. Train acc 79.100. Val loss 1.181. Val acc 79.803. Took 392.026 seconds
Epoch [1]/[1]. Batch [150]/[894]. Train loss 1.234. Train acc 79.533. Val loss 1.158. Val acc 82.042. Took 372.037 seconds
Epoch [1]/[1]. Batch [200]/[894]. Train loss 1.187. Train acc 80.850. Val loss 1.132. Val acc 81.773. Took 366.345 seconds
Epoch [1]/[1]. Batch [250]/[894]. Train loss 1.165. Train acc 81.800. Val loss 1.128. Val acc 81.012. Took 383.135 seconds
Epoch [1]/[1]. Batch [300]/[894]. Train loss 1.156. Train acc 82.033. Val loss 1.117. Val acc 82.266. Took 403.684 seconds
Epoch [1]/[1]. Batch [350]/[894]. Train loss 1.143. Train acc 81.943. Val loss 1.099. Val acc 83.296. Took 390.789 seconds
Epoch [1]/[1]. Batch [400]/[894]. Train loss 1.129. Train acc 82.025. Val loss 1.104. Val acc 80.923. Took 389.984 seconds
Epoch [1]/[1]. Batch [450]/[894]. Train loss 1.121. Train acc 82.200. Val loss 1.083. Val acc 83.162. Took 365.795 seconds
Epoch [1]/[1]. Batch [500]/[894]. Train loss 1.106. Train acc 82.540. Val loss 1.069. Val acc 82.311. Took 383.093 seconds
Epoch [1]/[1]. Batch [550]/[894]. Train loss 1.102. Train acc 82.636. Val loss 1.062. Val acc 81.729. Took 382.602 seconds
Epoch [1]/[1]. Batch [600]/[894]. Train loss 1.092. Train acc 82.850. Val loss 1.043. Val acc 82.803. Took 388.505 seconds
Epoch [1]/[1]. Batch [650]/[894]. Train loss 1.089. Train acc 82.754. Val loss 1.027. Val acc 83.968. Took 377.113 seconds
Epoch [1]/[1]. Batch [700]/[894]. Train loss 1.088. Train acc 82.757. Val loss 1.032. Val acc 82.221. Took 385.213 seconds
Epoch [1]/[1]. Batch [750]/[894]. Train loss 1.078. Train acc 82.880. Val loss 1.010. Val acc 82.669. Took 375.947 seconds
Epoch [1]/[1]. Batch [800]/[894]. Train loss 1.073. Train acc 83.050. Val loss 1.012. Val acc 83.609. Took 368.565 seconds
Epoch [1]/[1]. Batch [850]/[894]. Train loss 1.065. Train acc 83.153. Val loss 0.988. Val acc 84.236. Took 371.113 seconds
Epoch took 6606.454138994217
```

Train cand Validation Losses with 3 epochs



Train and Validation Losses for the Fourth epoch



C o n c l u s i o n

Sign language recognition has been a major area of research in the field of digital imaging. Our project could be regarded as a promising solution in medical applications that employ deep learning with high accuracy, and potentially improve such systems.

This study can be improved by including more images in the dataset for more letters and words. The proposed system could be improved to predict a complete word by adding new words and terms. These predicted words can also be converted into speech using a text-to-speech engine.

THANK YOU