Hand Gesture Recognition using AlexNet

Anjan Krishna Kandimalla
ES DS, University at Buffalo
Person Number: 50471086
anjankri@buffalo.edu

Sarwabowma Sri Sai Karthikeya Sarraju
ES AI, University at Buffalo
Person Number: 50442901
sarwabow@buffalo.edu

Abstract

The model that we developed was Hand Gesture Recognition using AlexNet where our model was designed to predict the hand gestures made in our normal life. We used the neural network AlexNet which we built from zero and then trained the model with the dataset that we got. We evaluated our model based on Accuracy, Precision, Recall, and F1 score. With AlexNet neural network we achieved an accuracy almost close to 99 percent. Later we integrated with webcam and tested hand gesture recognition in real time. Talking about contributions, as mentioned earlier we wrote our neural network on our own we didn't use the pre-trained model available through Python libraries.

Approach

We used AlexNet for the project, we also tried using Vgg16 and Resnet10 neural network models but because of our limited computational resources we used AlexNet when using other neural networks, we got our kernel crashed, for your reference we also attaching the ipynb files of Vgg16 and Resnet10. Coming to the architecture of AlexNet, Alexnet consists of 5 convolutional layers and three fully connected layers[1].

For the first convolution layer, we have a kernel of size 11 with stride 4, for the second convolution layer, the kernel size is 5 and the convolution layers third, fourth, and fifth have a kernel of size 3. The first and second convolutional layers are followed by a max pooling layer to down sample the resulting feature maps. Once the input was passed through all convolutional layers, we flatten our feature maps and pass it through fully connected layers, there will be a dropout added after the first layer to prevent overfitting. The main advantage of AlexNet is that we can train our models faster when compared to other neural networks. The disadvantage is that it uses the normal distribution to initiate weights in neural networks, which doesn't help solve the gradient vanishing problem[2]. The optimizer we used was Adam, which is considered the best optimizer, one main advantage of Adam optimizer is it converges faster, and it is insensitive to noise[3]. To calculate the loss values, we used cross entropy function. We trained our model for 10 epochs and divided our dataset into batches with a size of 128. Our total code consists of three main functions one is AlexNet, second is Train function and last is accuracy function. For writing the AlexNet function

[1] Alex Krizhevsky, Ilya Sutskever and Geoffrey E.Hinton. ImageNet Classification with Deep convolutional Neural networks

[2] https://medium.com/analytics-vidhya/concept-of-alexnet-convolutional-neural-network-6e73b4f9ee30

[3] https://dev.to/amananandrai/10-famous-machine-learning-optimizers-1e22#:~:text=Adam%20is%20a%20popular%20optimization,use%20than%20other%20optimization%20algorithms.

we wrote it on our own but used IEEE paper as our reference, for writing the Train function and Accuracy function we used the Pytorch documentation we used small snippets of pytorch documentation[4]. AlexNet is the project's heart and the main part of the code.

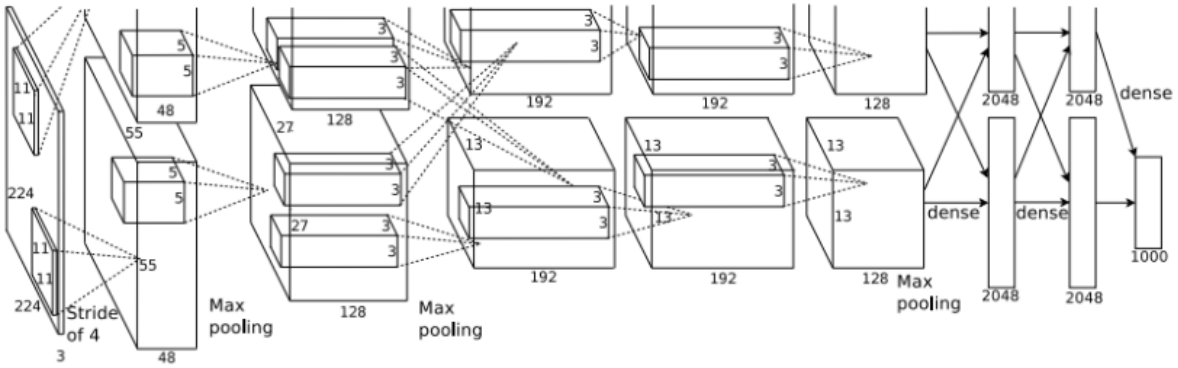


Figure 1 Architecture of AlexNet

https://medium.com/analytics-vidhya/concept-of-alexnet-convolutional-neural-network-6e73b4f9ee30

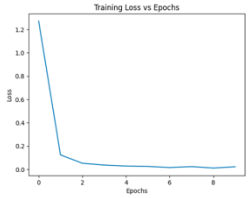
Experimental Protocol

The data set we used contains 10 labels which are[5]
01_palm
02_l shape
03_fist
04_fist_movesd
05_thumb
06_index
07_ok
08_palm_moved
09_c shape
10_down

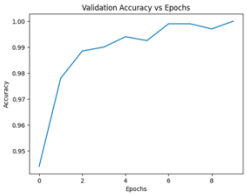
The dataset consists of different hand gestures made by people in their daily life, since we were trying to recognize the hand gestures this dataset can be helpful for building the project. We evaluate our success on accuracy of our model and on precision, recall and f1 scores. We ran our model on gpu named cuda 11.3 version, we train our model on gpu cuda. Training on gpu helped us to train our model fast. When we ran on CPU on epoch took almost 67min to run, when we switch to cuda it took only 9 mins for epoch and total 10 epochs took 92mins to run.

[4] https://pytorch.org/tutorials/
[5] https://www.kaggle.com/datasets/gti-upm/leapgestrecog

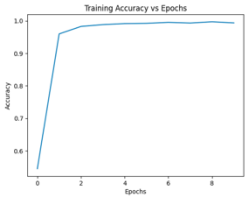
Results



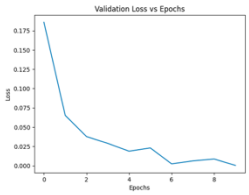
The graph represents the loss values of training data for each epoch



The graph represents the loss values of validation data for each epoch



The above graph represents the accuracy of the model on training data for each epoch

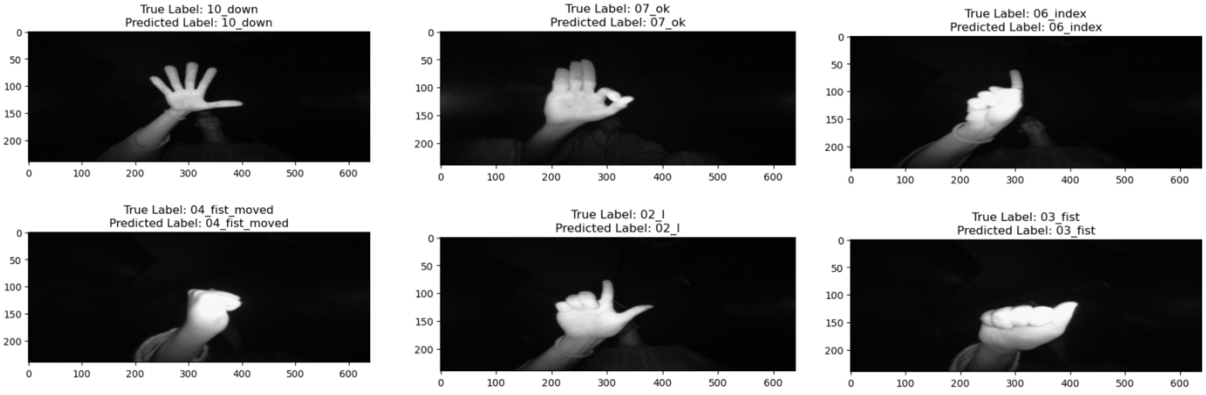


The above graph represents the accuracy of the model on validation data for each epoch

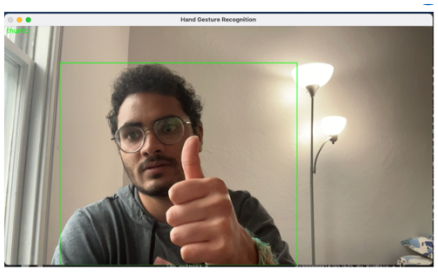
We can see that the loss values for both training and validation datasets is constantly decreasing and after a certain number of epochs the loss values were constant for the rest of the epochs. By looking at both loss values graph we can say that our model performed well since the loss values goes on decreasing on each epoch. Coming to the accuracy graphs it is clearly visible that right from the first epoch the accuracy is almost greater than 90 percent. The above are some qualitative results and as mentioned above we also evaluate our model on quantitative results like accuracy, precession, recall, f1 score.

Providing the scores below
Accuracy: 0.99933337
Precision: 0.999316
Recall: 0.999390
F1 score: 0.999351

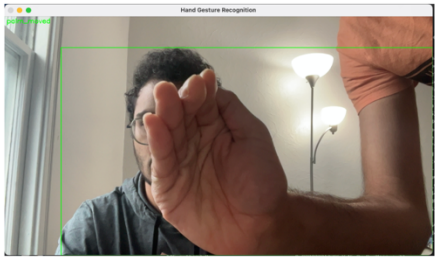
Based on the above scores we can draw a conclusion that our model really performed well.



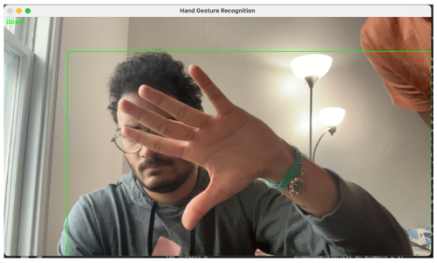
Real Time testing results



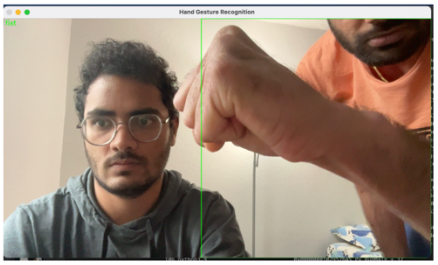
Thumb



Palm_Moved



Down



Fist

Analysis

Although we received an accuracy of 99 percent, we still have some problems while trying to predict the label of the image when we use the data that was captured from the real world. Like the model is predicting the label but it was predicting the wrong label, the model was not performing well on new data.

Discussion and Lessons Learned

We can say that gesture recognition will be a key part of many computer vision areas like in the self-driving cars, robotics, virtual reality etc. Through this project we injected us with many experiences, initially we tried to integrate the model with camera, but in real time testing we did not achieve the accuracy as we got on the dataset, that's the point where we learned that there are a lot of things to be learned we just had the basics of computer visions. Coming to the things we learned we got an overview of deep learning and how neural networks are used in classification of images, and how to include deep learning into computer vision.

References

[1] Alex Krizhevsky, Ilya Sutskever and Geoffrey E.Hinton. ImageNet Classification with Deep convolutional Neural networks

[2] https://medium.com/analytics-vidhya/concept-of-alexnet-convolutional-neural-network-6e73b4f9ee30

[3] https://dev.to/amananandrai/10-famous-machine-learning-optimizers-1e22#:~:text=Adam%20is%20a%20popular%20optimization,use%20than%20other%20optimization%20algorithms.

[4] https://pytorch.org/tutorials/

[5] https://www.kaggle.com/datasets/gti-upm/leapgestrecog