UTILISING CNNS FOR HAND GESTURE RECOGNITION

Aditi Jangra
NIIT University
MehtA+

Jerry Zheng
Westminster School
MehtA+

Mike Mitchell
Innovation High School
MehtA+

Sushmit Chakma
Chattogram Cantonment Public College
MehtA+

July 25, 2024

ABSTRACT

Hand recognition plays a critical role in allowing humans to more easily interact with machines by enabling another method of intuitive communication with digital devices. This study investigates the use of Convolutional Neural Networks (CNNs) in recognizing various hand gestures. We utilized a dataset consisting of 20,000 images. We experimented with many different types of CNNs, including: VGG-16 and ResNet-18. By leveraging the spatial feature extraction capabilities of CNNs, we've built an architecture for hand gesture recognition. This research provides insights into the application of deep learning techniques for gesture recognition and suggests potential future directions for improving interaction with digital devices in various domains such as medical and industrial applications.

1 Introduction

Hand gesture recognition is a major component in the further development of human-computer interaction, allowing for intuitive and natural communication with digital devices. It has a wide variety of use cases: from medical applications, where devices can be interacted with without physical contact in order to maintain a sterile environment, to industrial applications, where machinery can be operated in cases when manual control could be challenging or dangerous.

2 Related Work

Hand gesture recognition using Convolution Neural Networks is a very active area of research, with many having done research in the area before, due to the variety of applications. Additionally, there have been many different approaches, such as the utilisation of transformers [1] in order to integrate speech with hand gestures, or with a focus on real-time detection [2].

3 Methodology

3.1 Dataset

The dataset that we used contained 20,000 images, split between 10 different hand gestures, namely: palm, l, fist, fist moved, thumb, index, ok, palm moved, c, down. Additionally, the gestures were performed by 10 different subjects, including 5 males and 5 females. Hand gesture recognition database we used composed by a set of near infrared images acquired by the Leap Motion sensor.

Preprocessing With the aforementioned dataset, we transformed the categorical labels of the hand gestures into integer labels. We then created a one-hot encoded representation of the labels, before splitting the dataset 80-20 between a training and validation dataset.

3.1.1 Data Transformations

The image preprocessing begins with a series of transformations to prepare the data for model input.Initially the input image is converted into a PIL (Python Imaging Library) format for compute. Then it is resized to match the dimensions of input image which is 256 X 256.It is then followed by a centre crop. And then it is converted to a Pytorch tensor for numerical computation on it. Finally, the tensor undergoes normalization, adjusting its values to have a mean of 0.5 and a standard deviation of 0.225.

3.2 Model

Throughout the research we tried out various CNN models for our approach. We tried VGG-16, VGG-19 and Resnet-18 for the CNN architecture and LSTM (Long Short Term Memory). We aimed to learn about the temporal and spatial information of hand gesture with only one model. To address this we introduced the architecture which included a 2-D CNN followed by a LSTM and Softmax Classifier.

3.2.1 Experiments with VGG-16

VGG-16 is a CNN renowned for it's effective, yet straightforward design. Comprised of 16 layers, VGG-16 has 13 convolutional layers, and 3 fully connected layers. VGG-16 then makes use of max-pooling layers to manage the number of parameters.

Details We import the necessary Pytorch libraries and modules for loading the VGG16 model with weights pre-trained on ImageNet.To preserve the feature extraction capabilities of the pre-trained layers, we freeze all parameters in the model, preventing them from being updated during training.This allows us to leverage the generalized features learned from a large dataset.We then modify the final layer of the model's classifier to match our specific classification task.We replaced the last fully connected layer with a new one that outputs 10 classes to match our requirement.

Optimiser and Loss We used the Cross Entropy Loss for our this and Stochastic Gradient Descent as our optimiser. And for VGG-16 we implemented a learning rate of 0.001 and momentum as 0.9.

Cross Entropy Loss Function:

$$H = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{1}$$

Accuracy metrics In a batch size of 64 we ran the model for 4 epochs. For the training dataset the results were as follows-

Epoch Number	Correctly Classified Images	Total Training Images
1	15938	16000
2	15985	16000
3	15998	16000
4	16000	16000

For the validation dataset the results were as follows-

Epoch Number	Correctly Classified Images	Total Validation Images
1	3984	4000
2	3995	4000
3	3994	4000
4	3999	4000

We implemented a function to validate the model's performance and visualize its predictive accuracy through a confusion matrix. The function processes the validation dataset, and generates predictions for each input image. These predictions, along with their corresponding true labels, are collected and used to construct a confusion matrix, as per Figure 1. This matrix is then visualized as a heatmap using Seaborn, with color intensity indicating the frequency of predictions in each category. The plot illustrates correct predictions along the diagonal and misclassifications off the diagonal.

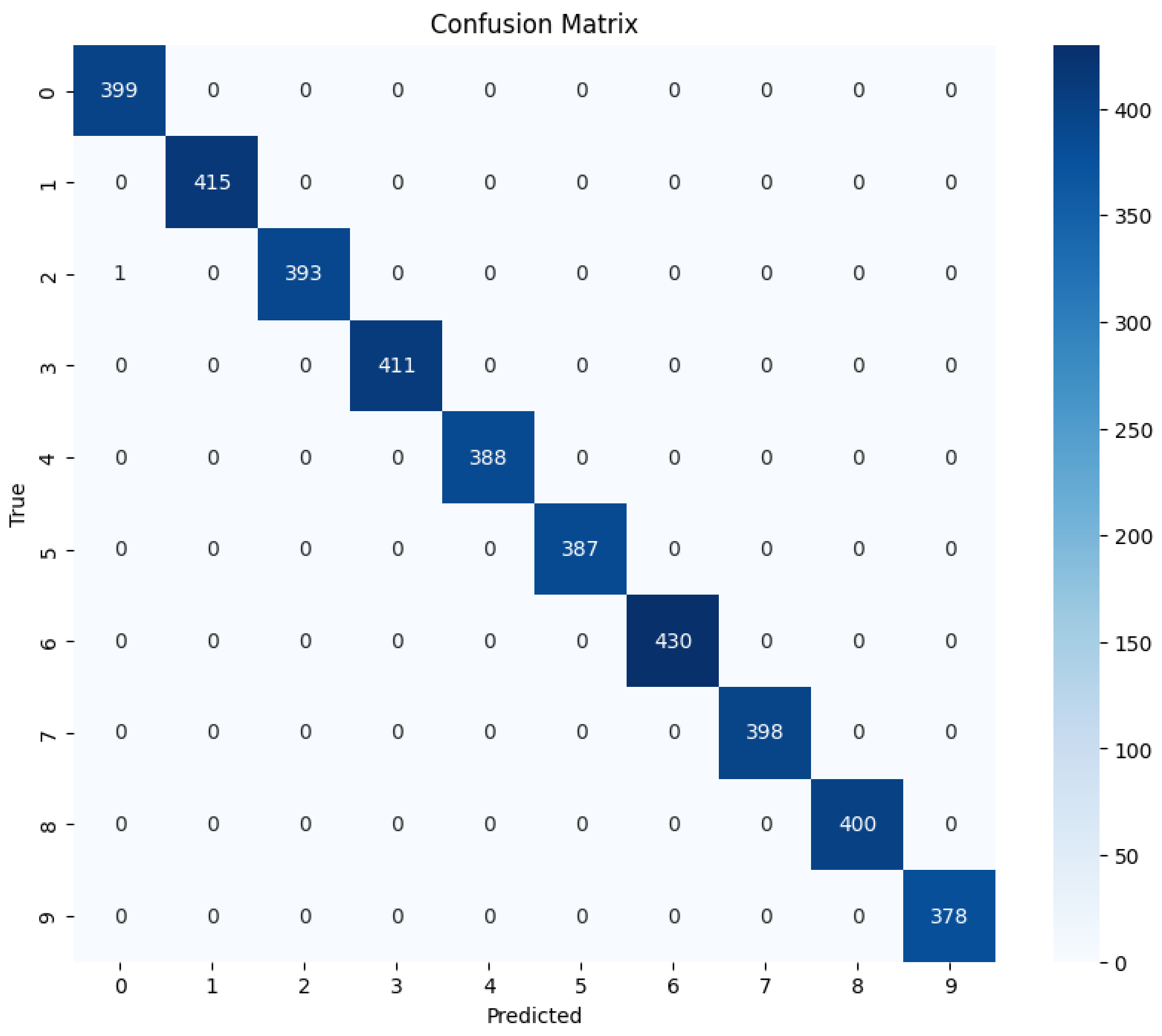

Figure 1: VGG-16 Confusion matrix

3.2.2 Experiments with ResNet-18

ResNet-18 is a CNN architecture known for introducing the concept of residual learning. ResNet-18 consists of 18 layers, including 17 convolutional layers and 1 fully connected layer. One of the key features of ResNet-18 is its use of residual blocks, which allow it to learn residual functions, as opposed to unreferenced functions. Each block contains shortcut connections which allow the model to skip between layers, allowing the training of deeper models.

Details We implemented a pre-trained ResNet-18 model from the torchvision library. We then began by importing necessary modules from We began by importing the necessary modules from PyTorch and torchvision. The model, pre-trained on ImageNet, was then loaded using the models.resnet18() function with the pretrained parameter set to True.To preserve the learned features and prevent them from being altered during our training process, we froze all the pre-trained layers of the model. Then the last fully connected layer (fc) of the ResNet-18 model was modified to match our specific classification task. We replaced this layer with a new Linear layer, where the input features remained the same as in the original model (model.fc.in-features), but the output was adjusted to match the number of classes in our dataset (num-classes).

Optimiser and Loss We used the Cross-Entropy loss, with Stochastic Gradient Descent and Adam as our optimizers. And for ResNet-18 we implemented a learning rate of 0.001 and momentum as 0.9.

Cross Entropy Loss Function:

$$H = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{2}$$

Accuracy metrics In a batch size of 64 we ran the model for 4 epochs. For the training dataset the results were as follows-

Epoch Number	Correctly Classified	Total Training Images
1	15913	16000
2	15983	16000
3	15968	16000
4	15994	16000

For the validation dataset the results were as follows-

Epoch Number	Correctly Classified	Total Training Images
1	3973	4000
2	3989	4000
3	3986	4000
4	3994	4000

In a similar way the confusion matrix for Resnet-18 is then visualized as a heatmap using Seaborn, with color intensity indicating the frequency of predictions in each category.

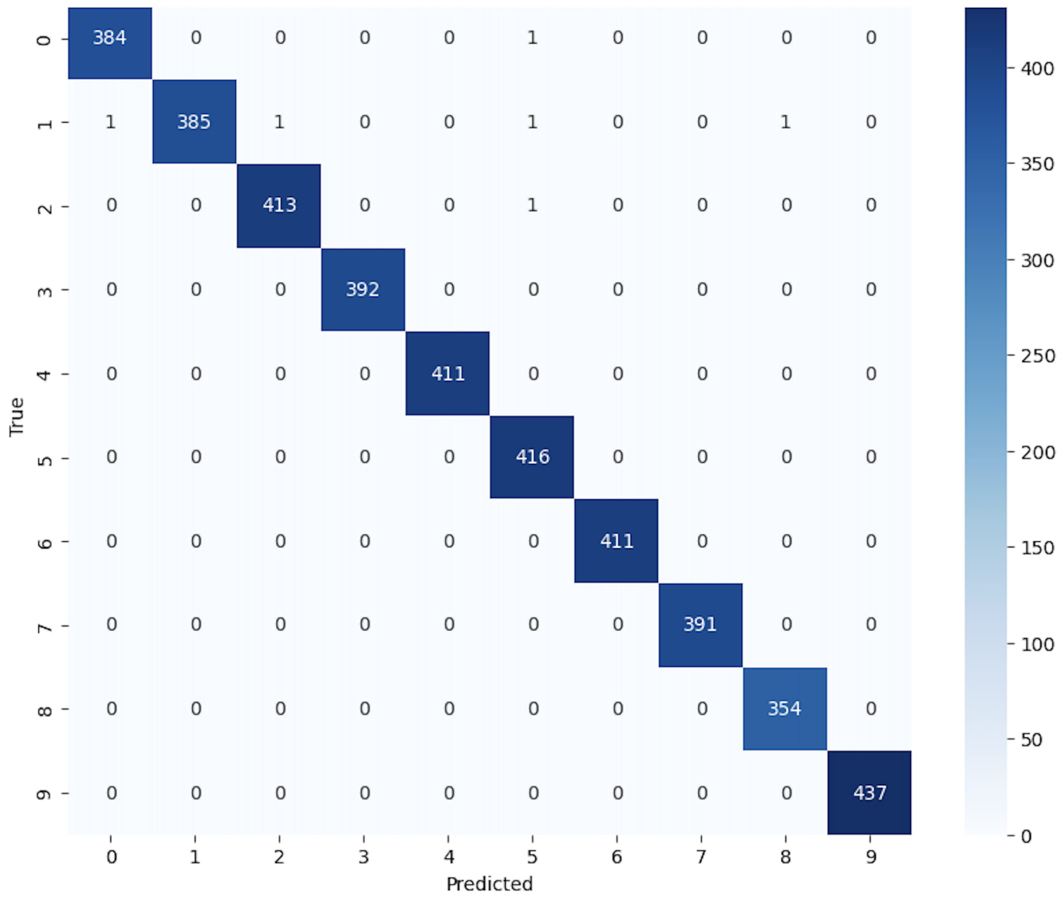

Figure 2: ResNet-18 Confusion matrix

4 Results and comparison of the Models

Below is a comparision table for both the models. In our evaluation process, we implemented a comprehensive set of metrics to assess the performance of our gesture recognition model. After the model processes the validation dataset, we collect all predictions and true labels.These metrics are computed using scikit-learn's functions, with the 'weighted' average option to account for potential class imbalance in our dataset. To quantify the model's performance more precisely, we calculate several key metrics:

Confusion Matrix Using all labels and true predictions, we construct a confusion matrix, which provides a detailed view of the model's classification performance across all classes.

F1 Score The F1 score provides a balanced measure of the model's accuracy by considering both precision and recall.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{3}$$

Recall Recall quantifies the model's ability to identify all relevant instances

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

Precision Precision measures the accuracy of positive predictions. We calculate these metrics both globally (across all classes) and for each individual class. This approach allows us to assess the model's overall performance as well as its effectiveness for each specific gesture type.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

Model	F1 Score	Recall	Precision
VGG-16	0.9997	0.9998	0.9998
ResNet-18	0.9985	0.9985	0.9985

4.0.1 Resnet-18

This table presents a detailed performance analysis of our gesture recognition model across different classes. It is structured to show metrics for 10 classes, labeled from Class 0 through Class 9, which correspond to different gesture types in our dataset. For each of these classes, the table provides three key performance metrics: F1 Score, Recall, and Precision.

Metric	Class 0	Class1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
F1 Score	0.9974	0.9948	0.9976	1.0000	1.0000	0.9964	1.0000	1.0000	0.9986	1.0000
Recall	0.9974	0.9897	0.9976	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Precision	0.9974	1.0000	0.9976	1.0000	1.0000	0.9928	1.0000	1.0000	0.9972	1.0000

4.0.2 VGG-16

Provides the same mentioned information but for VGG-16 model.

Metric	Class 0	Class1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
F1 Score	0.9987	1.0000	0.9987	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Recall	1.0000	1.0000	0.9975	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Precision	0.9975	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

5 Conclusion and Future Work

In our future work, we aim to enhance our gesture recognition system through several key advancements:

Dataset Expansion: We plan to significantly expand our dataset, incorporating a wider range of gestures from diverse populations. This includes deliberately introducing imperfections such as varying lighting conditions, complex backgrounds, and different camera angles to enhance the model's robustness in real-world scenarios.

Mobile and IoT Deployment: Our focus will shift towards optimizing the system for deployment on mobile phones and IoT devices. This involves reducing the model's size and computational demands, implementing efficient inference algorithms for real-time processing, and adapting to the resource constraints typical of mobile and IoT hardware.

Video Processing Capabilities: We intend to develop robust frame extraction and analysis techniques for continuous video streams. This enhancement will allow our system to process gesture recognition in real-time video inputs, significantly broadening its practical applications.

Advanced Architecture Integration: To further improve our model's performance, we plan to explore the integration of Visual Transformers and Long Short-Term Memory (LSTM) networks. Visual Transformers could enhance our model's ability to capture long-range dependencies and contextual information within gesture sequences. LSTMs, being well-suited for processing sequential data, could aid in understanding the context and evolution of gestures over time. This hybrid approach aims to capture both spatial and temporal relationships in gesture data more effectively.

6 Division of Labor

We divided the work as follows:

- Inputting and transforming the data: Aditi Jangra, Jerry Zheng
- Training the VGG-16: Aditi Jangra
- Training the ResNet-18 : Aditi Jangra
- Research Paper : Aditi Jangra, Jerry Zheng
- Poster : Aditi Jangra, Jerry Zheng, Mike Mitchell

7 Acknowledgements

We would like to acknowledge the efforts and sincerely like to thank our instructors- Ms. Haripriya Mehta,Mr. Arrun Sivasubramanian, Mr. Bhagirath Mehta.

References

[1] Esam Ghaleb, Ilya Burenko, Marlou Rasenberg, Wim Pouw, Ivan Toni, Peter Uhrig, Anna Wilson, Judith Holler, Aslı Özyürek, and Raquel Fernández. Leveraging speech for gesture detection in multimodal communication. *arXiv preprint arXiv:2404.14952*, 2024.

[2] Okan Köpüklü, Ahmet Gunduz, Neslihan Kose, and Gerhard Rigoll. Real-time hand gesture detection and classification using convolutional neural networks. In *2019 14th IEEE international conference on automatic face & gesture recognition (FG 2019)*, pages 1–8. IEEE, 2019.