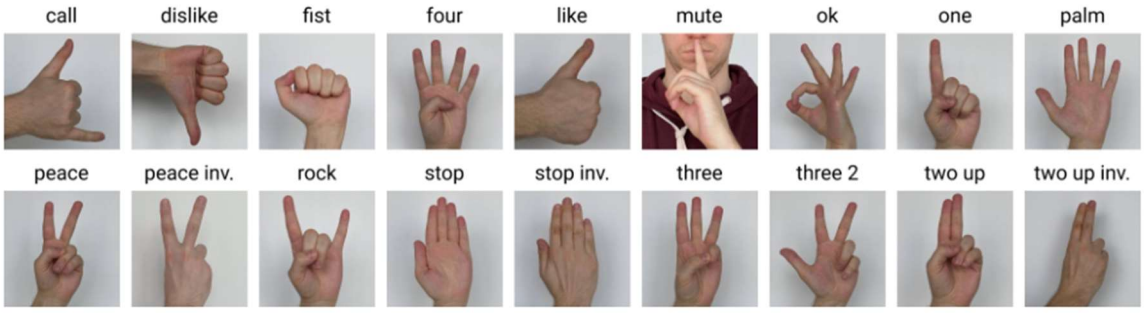
Daniel Fitzsimons / G00393110

Introduction

This Assignment investigates and analyses the recognition of hand gestures through the use of convolutional neural networks. Hand Gesture recognition is vital in technological fields such as virtual reality, robotics and human-computer interaction. This can be challenging based on several factors of variations including the environment, lighting, and orientation of the images. Using the HaGRID dataset provided from Kaggle [1], this provided a collection of hand gesture images spread across 18 different consistent classes. The objective is to build and compare multiple CNN models to determine the most effective approach for gesture recognition.

Data Set and Splitting

The HaGRID original data set is 716GB in size and contains 552.992 images within the data set. These images are divided into 18 separate classes as follows:



For this assignment, due to the data set's large size, the data set was cut down to just under 4GB in size.

Data Splitting

The dataset's total size was determined using the `cardinality()` method, which returns the total number of batches in the dataset. Based on this total, the dataset was programmatically divided into training, validation, and test datasets using the calculated indices:

- **Training Set:** This was split to 70% of the data.
- **Validation Set:** 10% of the data here is reserved for validating the model.
- **Test Set**: The remaining 20% is used to test model after training.

Data Preprocessing

Image Resizing

The initial image size was 512x512, but due to hardware limitations at the start, testing on this image size was deemed far too large as the machine kept crashing and wouldn't allow models to train. I opted to go straight for image size 128x128, which benefitted the training time for

models. Using 128x128 along with L4 specifications on Google Collab, allowed for better training of models on the dataset.

RGB VS Grayscale

Choosing between grayscale and RBG images for model training can vary due to Colour information helping the model to distinguish between nuanced variations in gestures that might not be apparent in grayscale, which should enhance accuracy and effectiveness in recognizing diverse hand gestures.

Data Augmentation

Data Augmentation was achieved by applying various transformations to the training images, such as random rotations, flips, and shifts, which introduced variability without altering the underlying gestures. These augmentations help models come variant with such changes in input images.

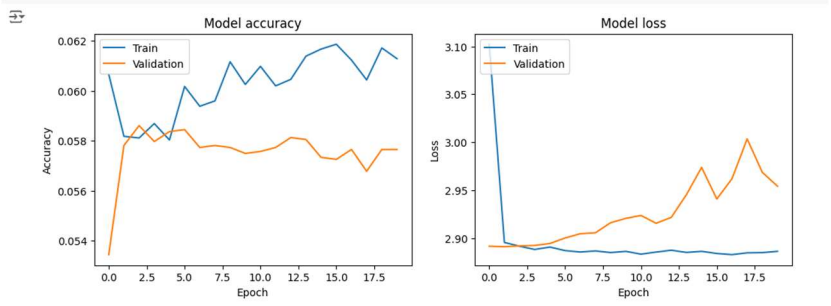
Using Google Collab GPU's

Introducing Google Collab to this project enabled me to access hardware beyond my machine's capabilities, With the use of the L4 GPU and High RAM, this allowed for the training of the models the decrease from hours to minutes, therefore speeding up the process and allowing for the training of deeper models.

Classifier Models

Basic CNN:

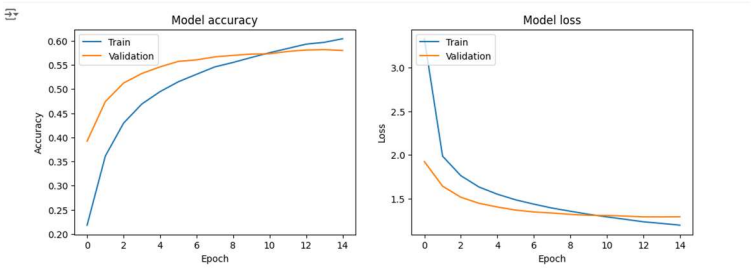
This model, constructed from scratch with three convolutional layers followed by max pooling layers and ReLU activations, integrates dropout strategies to combat overfitting. Using 20 epochs, the model's performance incrementally improved but remained notably low, peaking at a training accuracy of 6.13% and a validation accuracy of 6.76%, with final epoch losses reported at 3.0131 for training and 2.9541 for validation. Each epoch averaged 252 seconds, reflecting substantial computational demands. These results suggest challenges in model efficacy, possibly due to underfitting or inadequate training data representation.



VGG-16 Model:

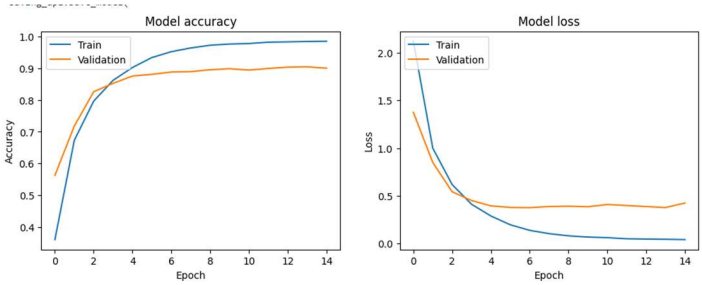
This model incorporates the VGG16 architecture, pre-trained on ImageNet, as a foundational feature extractor, with its layers frozen to preserve learned features. The network is extended with a global average pooling layer followed by a dense layer of 512 neurons with ReLU activation and a dropout rate of 0.5 to prevent overfitting. The final layer is a softmax layer

configured for multiple classes. Trained over 15 epochs with a low learning rate of 0.0001, the model demonstrated progressive learning, achieving a training accuracy of 60.43% and a validation accuracy of 57.99%, reflecting steady convergence. The model required approximately 215 to 255 seconds per epoch, indicative of the computational complexity introduced by the VGG16 layers. The final evaluation on the test set yielded an accuracy of 58.08%, underscoring the model's capability to generalize to new data while highlighting areas for potential improvement in feature adaptation and training optimization.



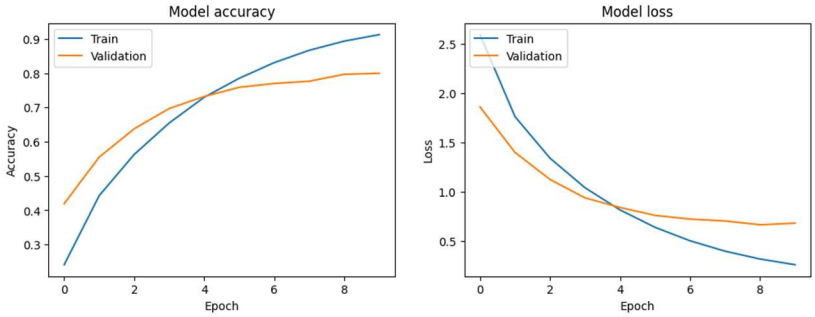
Deep CNN:

This model utilizes a deeper CNN architecture with five convolutional blocks, each followed by max pooling and batch normalization layers, culminating in dense layers for classification. The model was trained over 15 epochs using the Adam optimizer with a learning rate of 0.0001 and employed dropout to mitigate overfitting. The training process showed significant improvement, achieving a final training accuracy of 98.57% and a validation accuracy of 90.07%, with corresponding losses of 0.0420 and 0.4250, respectively. Each epoch averaged around 202 seconds, highlighting the increased computational load due to the deeper network. The robust performance metrics suggest effective feature extraction and learning, making this model a strong candidate for gesture recognition tasks.



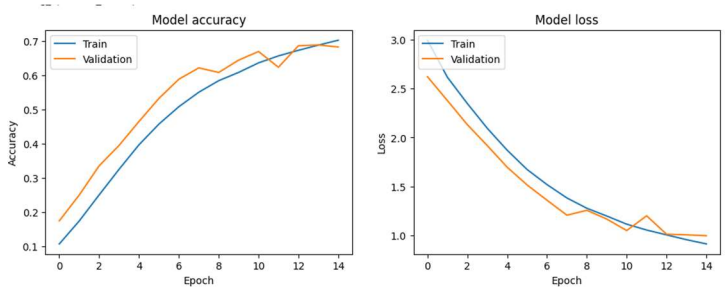
Deep CNN Grayscale:

This CNN model, designed to process grayscale images, comprises four convolutional blocks each followed by max pooling and batch normalization layers, and is capped with a dense layer for classification. The model was trained over 10 epochs with the Adam optimizer at a learning rate of 0.0001. The training process yielded a final accuracy of 91.27% and a validation accuracy of 80.00%, with respective losses of 0.2598 and 0.6816. The average training time per epoch was approximately 203 seconds. Despite the computational efficiency afforded by using grayscale images, the performance indicates that the model effectively learned from the dataset, although there remains room for improvement in validation accuracy.



CNN with Data Augmentation (Color):

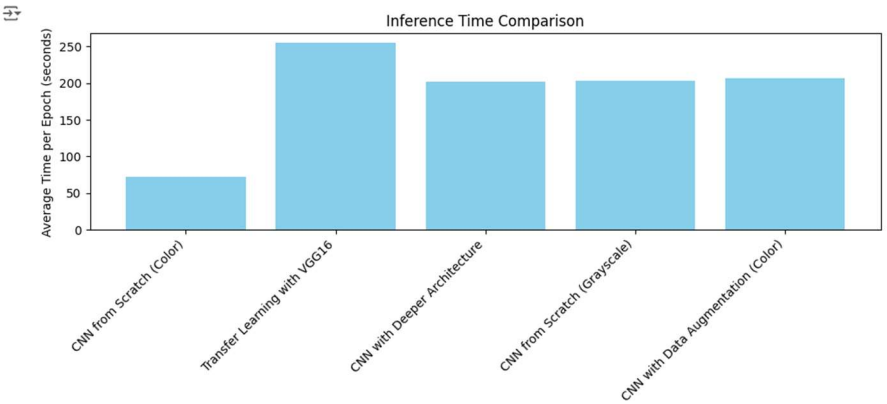
This CNN model, enhanced with data augmentation, comprises four convolutional blocks with ReLU activations, max pooling, and batch normalization layers. The architecture includes dense layers for final classification. Trained over 15 epochs using the Adam optimizer with a learning rate of 0.0001, the model demonstrated consistent improvement, achieving a training accuracy of 70.31% and a validation accuracy of 68.32%, with corresponding losses of 0.9110 and 0.9952. Each epoch averaged around 207 seconds, reflecting the computational load of the augmented dataset. The performance suggests that data augmentation effectively aided the model in learning robust features, although further tuning may be needed to enhance validation accuracy.



Results

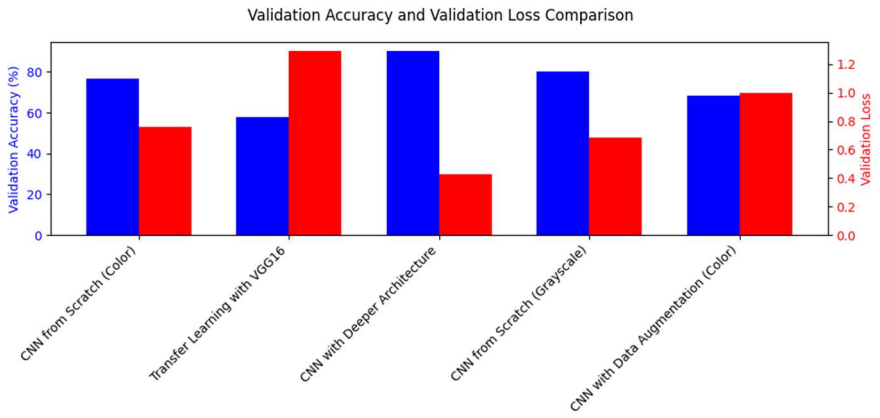
Comparisons

Inference Table



The Inference Time Comparison graph highlights the computational efficiency of each model. The CNN from Scratch (Color) model is the fastest, taking an average of 72.2 seconds per epoch. In contrast, the Transfer Learning with VGG16 model requires significantly more time, averaging between 252 to 258 seconds per epoch, reflecting the increased complexity and depth of the pre-trained architecture. The other models, including the CNN with Deeper Architecture, CNN from Scratch (Grayscale), and CNN with Data Augmentation (Color), have similar inference times, ranging from 202 to 207 seconds, indicating moderate computational demands.

Validation Accuracy and Validation Loss



The combined Validation Accuracy and Validation Loss Comparison graph provides a dual perspective on model performance. Validation accuracy, represented by blue bars, highlights the model's ability to generalize to unseen data, with the CNN with Deeper Architecture achieving the highest accuracy of 90.07%. Validation loss, represented by red bars, indicates the error in the model's predictions, with the same model also exhibiting the lowest validation loss of 0.4250. The Transfer Learning with VGG16 model, despite its sophisticated architecture, shows higher validation loss and lower accuracy, suggesting room for further optimization.

Choosing Best Model

During the initial evaluation, the CNN with Deeper Architecture emerged as the top performer based on validation metrics. It achieved the highest validation accuracy of 90.07% and the lowest validation loss of 0.4250. These metrics indicated strong generalization capabilities during the training phase. Conversely, the CNN from Scratch (Grayscale) also showed commendable performance with a validation accuracy of 80.00% and a relatively low validation loss of 0.6816.

Real-World Testing with Gesture Predictions

To further assess the practical applicability of these models, they were tested on a separate set of gesture images. Despite its high validation accuracy, the CNN with Deeper Architecture failed to correctly classify any of the gestures. This discrepancy suggests that the model, while effective during training, may not generalize well to new, unseen data, possibly due to overfitting or differences between the training data and real-world images.

On the other hand, the CNN from Scratch (Grayscale), despite having lower training metrics, managed to correctly predict one gesture. This indicates a better, albeit still limited, generalization to real-world data compared to the deeper CNN model. The grayscale model's ability to correctly classify a gesture may be attributed to its simplicity and potential robustness against overfitting.

Conclusion

This study explored the effectiveness of various convolutional neural network (CNN) architectures for hand gesture recognition using the HaGRID dataset. Multiple models were constructed and evaluated, including a basic CNN, a VGG-16 based model, a deep CNN, and a grayscale CNN. The deep CNN achieved the highest validation accuracy of 90.07% and the lowest validation loss, indicating strong learning capabilities during training. However, it struggled with real-world data, highlighting overfitting issues. Conversely, the grayscale CNN, despite lower training metrics, demonstrated better real-world applicability by correctly predicting a gesture, suggesting robustness against overfitting and potential for practical use.

Overall, while the deep CNN showed promise in training, further optimization is needed for real-world application. The simpler grayscale CNN, with its better generalization to new data, offers a valuable direction for developing effective hand gesture recognition systems.

References

[1] Kaggle HaGrid Dataset (https://www.kaggle.com/datasets/kapitanov/hagrid)