Exploring CNN Architectures and Transfer Learning for Image Classification

Eoin Lawless

GUI Assessment 2

Abstract

This paper presents the process and results of experimenting with various Convolutional Neural Network (CNN) architectures, both from scratch and using transfer learning with VGG-16 as a base model. Different image resolutions, color modes (RGB vs Grayscale), and techniques are examined. The findings highlight the comparative performance of each model under different conditions. Results are discussed in terms of accuracy, training time, and generalizability.

Keywords: Convolutional Neural Networks, Image Classification, Transfer Learning, VGG-16,

1 Introduction

This study investigates the performance of different Convolutional Neural Network (CNN) architectures and training strategies. In particular, the focus is on comparing CNNs built from scratch with those using transfer learning based on VGG-16 models. Additionally, variations in image resolution, color mode, and techniques for improving efficiency are examined..

1.0.1 Dataset Organization

The dataset used in this project was a curated subset of the Hagrid dataset,consisting of 18 distinct hand gestures, significantly reduced in size to manage resource constraints. Initially containing over 125,912 images, the dataset was halved to remain under 4GB, thereby making it feasible to process with the available computational resources.

1.1 Data Pre-Processing

Effective data pre-processing is crucial for the successful training of machine learning models. This project involved several key steps to prepare the dataset for training, validation, and testing of CNNs aimed at gesture recognition.

The pre-processing began by structuring the dataset into three sets: training, validation, and testing. Separate directories were created for each set, further organized into sub-directories for each of the eighteen gesture classes. The data was split with 70% for training, 20% for validation, and 10% for testing, ensuring sufficient data for learning and evaluation.

The training machine was equipped with an Intel Iris Xe Max integrated graphics card, which did not support GPU-based data processing. Consequently, all models were compiled and executed on the CPU, specifically an Intel Core i7 processor. This limitation required models to be trained asynchronous, as the machine could not be used for other tasks during training.

1.1.1 Randomization

To ensure generalization and reduce potential biases, the dataset was randomized using a seed derived from the student ID (396053), ensuring consistency and reproducibility of the splits across different runs. This process involved shuffling the images within each class before distributing them across the training, validation, and test directories according to the predefined ratios.

1.1.2 Image Data Generators - Data Augmentation

Further data preparation involved the use of an Image Data Generator, which facilitates real-time data augmentation and image preprocessing. This alters images during training through transformations such as rotations and shifts, significantly augmenting the size of the dataset and introducing variability that helps in developing a more generalized model.

The augmentation parameters were chosen to include typical transformations that are likely to occur in a real-world application, such as minor rotations and horizontal flips. This approach simulates more realistic scenarios where the gestures might not be perfectly aligned or positioned identically during each instance.

2 The Models

Two sets of models were developed:

1. CNN architectures built from scratch.
2. VGG-16 models using transfer learning.

CNN from Scratch

The Convolutional Neural Network (CNN) architecture comprised several convolutional layers, pooling layers, and fully connected layers as follows:

Model Name	Image Size	Color Mode	Deeper Layers	Reduce Overfitting	Val-Accuracy
cnn_64x64_rgb	64x64	RGB	No	No	0.767
cnn_128x128_rgb	128x128	RGB	No	No	0.740
cnn_64x64_greyscale	64x64	Grayscale	No	No	0.655
cnn_64x64_rgb_deeper	64x64	RGB	Yes	No	0.898
cnn_64x64_rgb_overfitting	64x64	RGB	No	Yes	0.674
basic_cnn_64x64_rgb	64x64	RGB	No	No	0.442

Table 1: Summary of CNN Models and Their Performance

- **Convolutional Layers:** The networks included 2-4 convolutional layers, using 64 and 128 filters, respectively, to extract features from the input images. For deeper networks, additional convolutional layers with 256 and 512 filters were added.
- **Pooling Layers:** MaxPooling layers followed each convolutional layer to reduce spatial dimensions and promote the extraction of dominant features.
- **Fully Connected Layers:** After flattening the output from the convolutional and pooling layers, two fully connected layers with 1024 and 512 neurons were used. These layers are crucial for learning non-linear combinations of the high-level features extracted by the convolutional layers.

- **Dropout:** For models aimed at reducing overfitting, Dropout layers with rates up to 0.5 were included. These layers randomly drop units during training to ensure the network does not rely on any specific set of neurons.
- **Batch Normalization:** Some models included batch normalization layers to standardize the inputs to a layer for each mini-batch. This helps stabilize the learning process and dramatically reduces the number of training epochs required to train deep networks.

The models were compiled using the Adam optimizer with a specified learning rate of 0.0005, and the categorical cross-entropy loss function was employed. Each model variation was trained for 15 epochs with a batch size of 32. The only model trained without the data augmentation is the basic CNN model.

VGG-16 with Transfer Learning

The VGG-16 models utilized transfer learning, leveraging the pre-trained VGG-16 architecture as a base and fine-tuning the top layers for the gesture recognition task. Several configurations were explored, including variations in image size, the inclusion of deeper layers, and techniques to reduce overfitting.

Model Name	Image Size	Color Mode	Deeper Layers	Reduce Overfitting	Val-Accuracy
vgg16_64x64_rgb	64x64	RGB	No	No	0.715
vgg16_128x128_rgb	128x128	RGB	No	No	0.904
vgg16_64x64_rgb_deeper	64x64	RGB	Yes	No	0.720
basic_vgg16_64x64_rgb	64x64	RGB	No	No	0.413

Table 2: Summary of VGG-16 Models and Their Performance

- **Base Model:** The pre-trained VGG-16 model served as the base, with the top layers removed.
- **Fine-Tuning:** The last four layers of the VGG-16 base were fine-tuned to adapt to the new dataset, while the earlier layers were frozen to retain learned features.
- **Global Average Pooling:** Replaced the fully connected layers to reduce the total number of parameters and prevent overfitting.
- **Fully Connected Layers:** One or two dense layers with 512 or 1024 neurons were added for deeper configurations.
- **Dropout and Batch Normalization:** Included in some models to reduce overfitting, with dropout rates up to 0.5.

The models were compiled using the Adam optimizer with a learning rate of 0.0001, and the categorical cross-entropy loss function was used. Each model variation was trained for 15 epochs with a batch size of 32. The only model trained without the data augmentation is the basic VGG-16 model.

3 Experiments and Results

The results are presented in terms of training accuracy, validation accuracy, training loss, validation loss, and average epoch time. These metrics provide a comprehensive evaluation of each model's performance, considering both effectiveness and efficiency.

Model	Training Accuracy	Val Accuracy	Training Loss	Val Loss	Avg Epoch (s)
CNN 64x64 RGB	75.82%	76.72%	0.7516	0.7399	450.3
CNN 128x128 RGB	71.39%	74.03%	0.8978	0.8121	1485.0
CNN 64x64 Greyscale	63.02%	65.55%	1.1596	1.0757	415.8
CNN 64x64 RGB Deeper	89.54%	89.88%	0.3246	0.3212	403.0
CNN 64x64 RGB Overfitting	58.67%	67.44%	1.2809	1.0459	442.6
VGG-16 64x64 RGB	81.32%	71.46%	0.5556	0.9405	886.0
VGG-16 128x128 RGB	96.50%	90.39%	0.1045	0.3606	2340.0
VGG-16 64x64 RGB Deeper	80.43%	72.08%	0.5807	0.9041	825.0

Table 3: Model performance comparison with 15 epochs and variations.

Analysis

The performance of different models varied significantly across the metrics evaluated. Key observations include:

- **Training Accuracy and Validation Accuracy:**
  - *CNN 64x64 RGB Deeper* achieved the highest training accuracy of Scratch CNN Models having 89.54%, reflecting its ability to fit the training data well.
  - *VGG-16 128x128 RGB* outperformed other models with a validation accuracy of 90.39%, showcasing the effectiveness of transfer learning with larger image sizes.

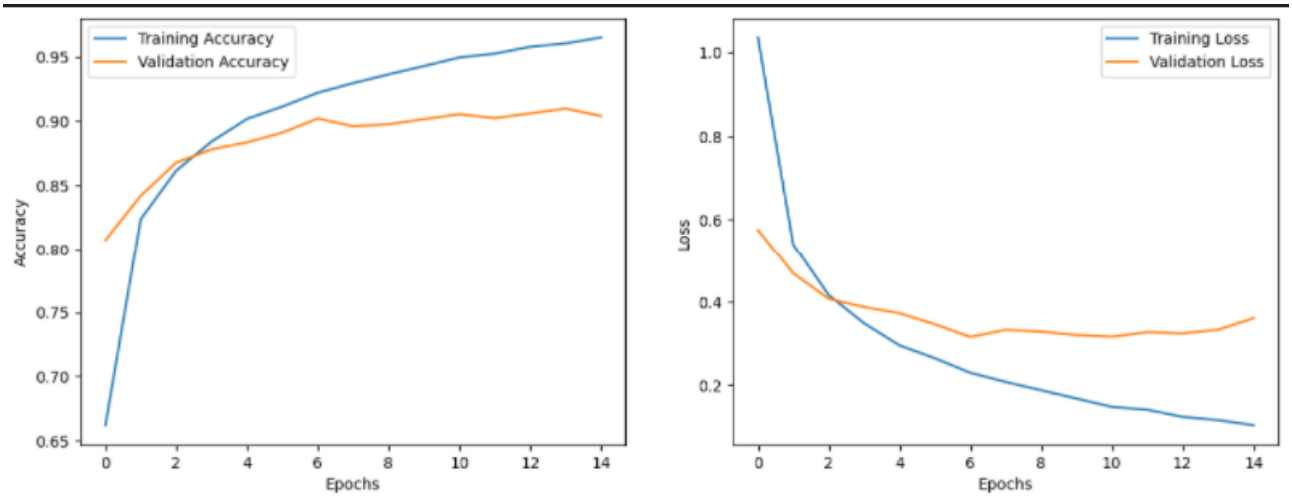


Figure 1: vgg16_128x128_rgb.h5 training graph

- **Average Epoch Time:**

Model Name	Average Epoch Time (seconds)
basic_cnn_64x64_rgb	195.2
vgg16_128x128_rgb	2361.0

Table 4: Average Epoch Time for Selected Models

The average epoch time for the selected models is summarized in Table 4. Key observations include:

- *VGG-16 128x128 RGB* required the longest training time per epoch (2361.0 seconds), which is expected due to the larger image size and the complexity of the VGG-16 model. Despite the long training time, this model achieved high accuracy, demonstrating the effectiveness of a deep and complex architecture combined with larger image sizes.
- *basic_cnn_64x64_rgb* had the shortest average epoch time (195.2 seconds) among all models. This model did not use data augmentation during training, which contributed to its faster training time. However, this came at the cost of accuracy, as the model exhibited significant overfitting with a high training accuracy but poor validation accuracy.

- **Greyscale vs RGB**

  - *cnn_64x64_rgb* required an average training time per epoch of 452 seconds. This model achieved a validation accuracy of 77.06%. The use of RGB images provides richer information, which helps in achieving higher accuracy.
  - *cnn_64x64_greyscale* had a shorter average epoch time of 422 seconds. However, the validation accuracy was lower at 65.55%. Grayscale images contain less information compared to RGB images, which can make it more challenging for the model to learn distinguishing features.

- **Models Trained Without Data Augmentation:**

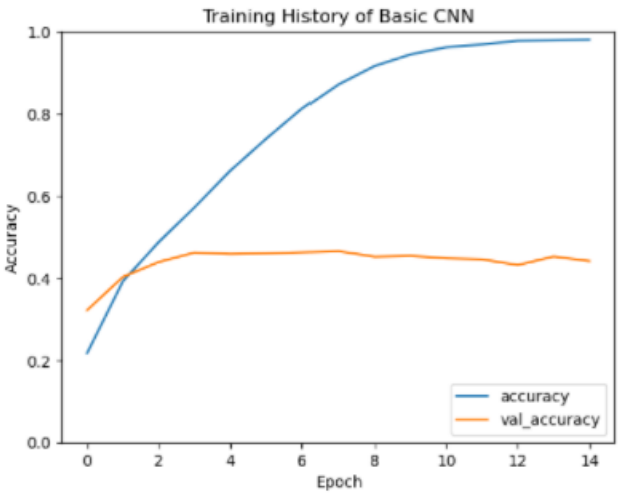


Figure 2: Training and Val Accuracy for Basic_CNN

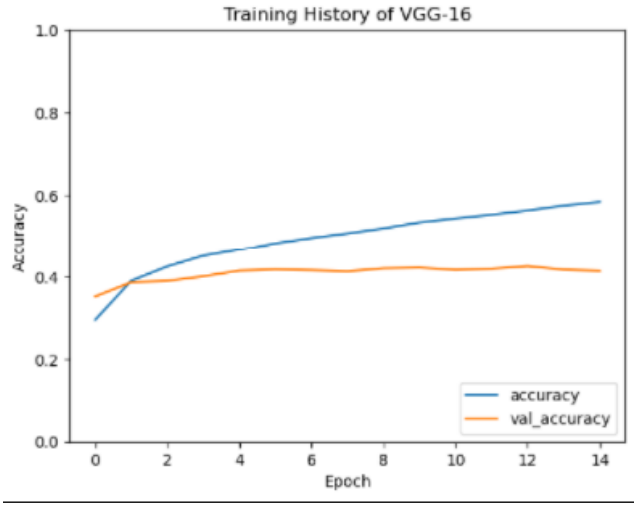
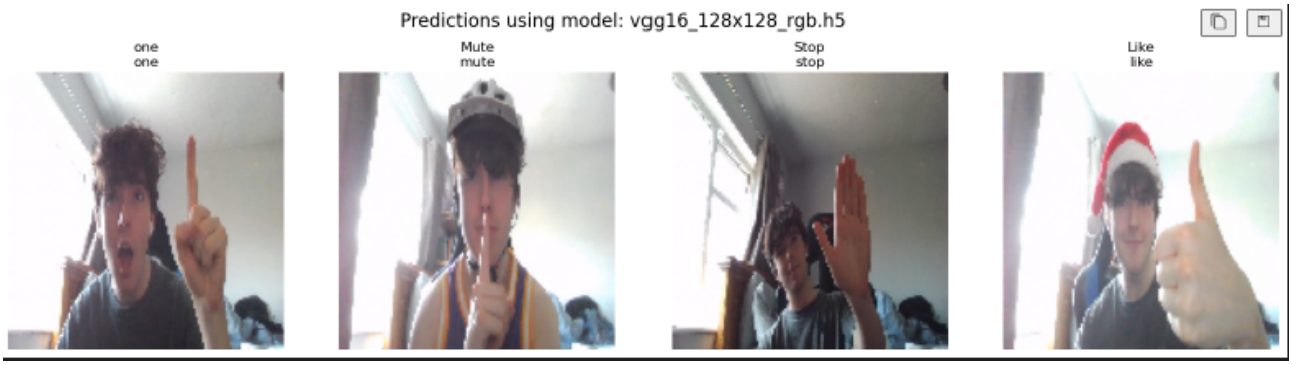


Figure 3: Training and Val Accuracy for Basic_vgg16

  - *basic_cnn_64x64_rgb* achieved a training accuracy of 97.99% and a validation accuracy of 44.16%, indicating significant overfitting.
  - *vgg16_64x64_rgb* achieved a training accuracy of 58.34% and a validation accuracy of 41.34%, showing moderate performance but still a notable gap between training and validation accuracy.

4 Final Evaluation

This paper has presented an exploration of various CNN architectures and their performance under different training conditions. The findings indicate that the VGG-16 128x128 RGB is best model :



Best VGG-16 model accuracy on self-captured images.

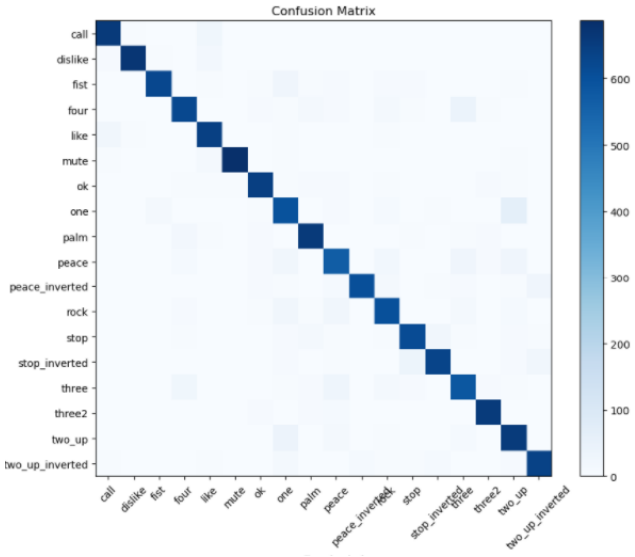


Figure 4: Confusion matrix of 90.39%.

- *VGG-16 128x128 RGB* had the lowest training loss (0.1045) and a relatively low validation loss (0.3606), suggesting a strong fit and good generalization capability.
- The confusion matrix visualizes the performance of the classification model. The strong diagonal line indicates that most predictions are correct, as the dark cells along the diagonal represent high values of true positive rates.
- The Test accuracy when used on this model for the dataset is 87.63

5 Conclusion

In conclusion, this project investigated and evaluated CNN models developed from scratch and through transfer learning. The results demonstrated the efficiency and limitations of these models in recognizing gestures, highlighting the VGG-16 128x128 RGB model as the most effective in terms of accuracy and generalization.

References

[1] Professor, Hurley, *Lecture 10 - Convolutional Neural Networks*. [Online]. Available: https://atu-main-mdl-euwest1.s3.eu-west-1.amazonaws.com/de/d8/ded830ec21602a10a503438e1c5e80c3749f8e73?response-content-disposition=inline

[2] A. Kapitanov, K. Kvanchiani, A. Nagaev, R. Kraynov, and A. Makhliarchuk, "HaGRID – HAnd Gesture Recognition Image Dataset," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2024, pp. 4572-4581.