Implementation and Analysis of Human Activity Recognition from Images using CNN based Models

Group 54: Gautham Shaji, Rohan Jayachandran
GXS210034 RXJ220025

Abstract

Human action recognition is a critical task in computer vision with applications in various domains, such as surveillance, robotics, human-computer interaction, and healthcare. In this project, we investigate the effectiveness of three popular convolutional neural network (CNN) architectures - VGG16, ResNet50, and VGG19 - in recognizing human actions from images. For each architecture, we explore different approaches, including transfer learning, fine-tuning, and adaptive learning rates. We perform six different experiments using these approaches, aiming to compare their performance in terms of accuracy and loss on the human action recognition task. Our results provide valuable insights into the benefits and drawbacks of transfer learning, fine-tuning, and adaptive learning rates for each architecture, and contribute to the understanding of the most suitable model and approach for human action recognition.

Introduction

Human action recognition is an important task in computer vision, with applications in various fields such as surveillance, robotics, human-computer interaction, and healthcare. Deep learning models, particularly convolutional neural networks (CNNs), have shown great success in this domain. In this project, we investigate the effectiveness of three popular CNN architectures - VGG16, ResNet50, and VGG19 - in recognizing human actions from images. For each architecture, we explore different approaches, including transfer learning, fine-tuning, and adaptive learning rates. The goal is to compare the performance of these models in terms of accuracy and loss on the task of human action recognition and to analyze the benefits and drawbacks of transfer learning, fine-tuning, and adaptive learning rates for each architecture.

Data

For our experiments, we chose to use a publicly available Kaggle dataset containing 12,600 images for human action recognition. The dataset is split into three subsets: training, validation, and testing. We decided to split the data as follows: 90 percent of the images are used for training plus validation, with 80 percent allocated for training and 20 percent for validation. The remaining 10 percent of the images are used for testing purposes. This split ensures that we have a sufficient number of samples to train the models effectively, while also allowing us to validate and test the models on unseen data, providing a better assessment of their performance in real-world scenarios. The data is balanced in terms of the class distribution.

Methodology

In this study, we aim to analyze the performance of different deep learning approaches for the human action recognition task. Our methodology involves the following key components:

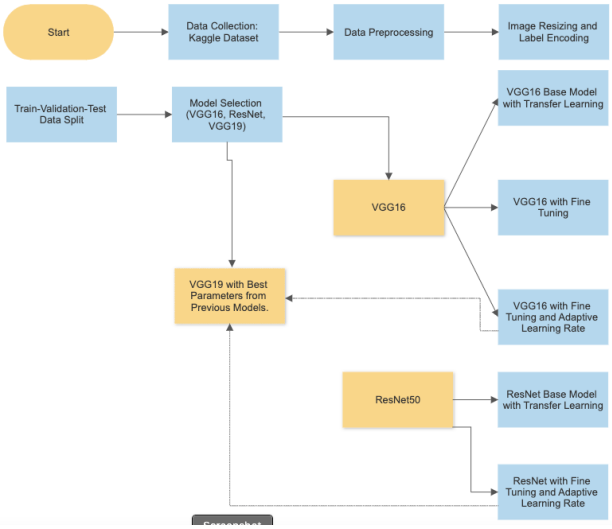

Figure 1: *Project Flowchart*

Dataset Selection: We use a Kaggle dataset consisting of 12.6k images, representing various human actions. The dataset is divided into training, validation, and testing subsets, ensuring that the models are evaluated on unseen data.

Model Architectures: We investigate three popular convolutional neural network (CNN) architectures - VGG16, VGG19, and ResNet - to build our human action recognition models.

Training Strategies: We explore different training strategies, including Transfer Learning, Fine-Tuning, and Adap-

tive Learning Rate, to improve the performance of our models on the task.

Data Augmentation and Normalization: To enhance the generalizability of our models, we apply data augmentation techniques and normalization according to the specific requirements of the pre-trained models.

Model Evaluation: We evaluate the performance of our models using accuracy and classification report metrics, allowing us to identify the best-performing approaches for human action recognition. The six approaches we have used in our experiments are as follows:

- **VGG16-1:** VGG16 with Transfer Learning
- **VGG16-2:** VGG16 with Fine-Tuning
- **VGG16-3:** VGG16 with Fine-Tuning and adaptive learning rate
- **ResNet-1:** ResNet with Transfer Learning
- **ResNet-2:** ResNet with Fine-Tuning and adaptive learning rate
- **VGG19:** VGG19 with fine-tuning based on the best parameters learned from the previous models

By employing these various approaches, we aim to identify the most effective method for human action recognition and provide insights into the performance of different CNN architectures and training strategies.

Implementation

Following the methodology outlined above, we implement our human action recognition models as follows:

Data Preparation: We load the dataset, preprocess the images by resizing them to 160x160 pixels, and split the dataset into training, validation, and testing subsets. Data augmentation techniques -: 0-20 degree random rotation of images, height and width shift, as well as horizontal and vertical flips were also done on the data. Finally, the data was normalized according to the specific requirements of the pre-trained models.

Model Configuration: We import the pre-trained VGG16, VGG19, and ResNet models from the Keras libraries, and modify them by adding our own classification layers for the human action recognition task.

Transfer Learning Implementation: We freeze the layers in the pre-trained models and train the newly added classification layers on our dataset, allowing the models to learn the specific features of human actions.

Fine-Tuning Implementation: We unfreeze some of the higher-level layers in the models and continue training with a lower learning rate, enabling the models to adapt to the human action recognition dataset and improve their performance.

Adaptive Learning Rate Implementation: We use the ReduceLROnPlateau callback function to adjust the learning rate during training, helping the models to converge faster and achieve better results.

Data Augmentation and Normalization: We create an ImageDataGenerator instance to apply data augmentation techniques and normalize the input data according to the requirements of the pre-trained models.

Model Training: We train the models using the fitGenerator method, specifying the number of epochs, batch size, and various callback functions, such as ModelCheckpoint, EarlyStopping, and ReduceLROnPlateau.

Model Evaluation: We assess the performance of the models on the test dataset using accuracy and classification report metrics, enabling us to compare the effectiveness of the different approaches and identify the best-performing models for human action recognition.

Experiments and Analysis

To thoroughly investigate the performance of the various approaches in our human action recognition task, we conducted several experiments using different model architectures, training strategies, and configurations. The experiments and their corresponding analysis are outlined below:

VGG16 Experiments:

a. Transfer Learning: We first trained the VGG16 model using transfer learning and observed the accuracy of the model on the validation set. The results served as a baseline for further experiments.

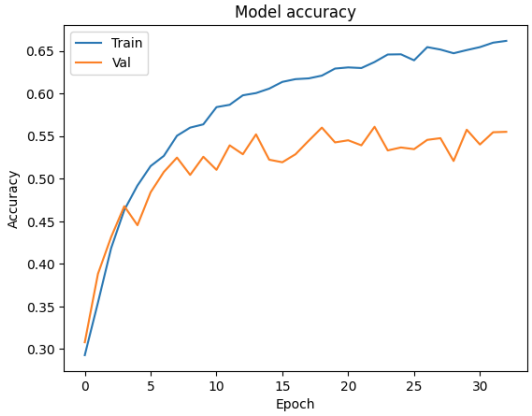

Figure 2: *VGG16-1 Accuracy*

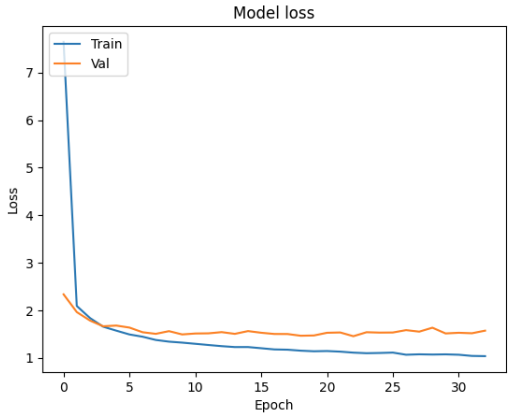

Figure 3: *VGG16-1 Loss*

VGG16 Model Accuracy without Fine-Tuning: 54.48%
Classification Report:

	precision	recall	f1-score	support
0.0	0.42	0.43	0.43	180
1.0	0.51	0.62	0.56	159
2.0	0.47	0.50	0.48	159
3.0	0.69	0.66	0.68	163
4.0	0.49	0.52	0.50	162
5.0	0.52	0.41	0.46	170
6.0	0.57	0.48	0.52	153
7.0	0.84	0.87	0.86	185
8.0	0.36	0.44	0.39	167
9.0	0.52	0.43	0.47	170
10.0	0.77	0.77	0.77	193
11.0	0.58	0.62	0.60	167
12.0	0.37	0.23	0.29	166
13.0	0.54	0.76	0.63	144
14.0	0.46	0.40	0.42	182
accuracy			0.54	2520
macro avg	0.54	0.54	0.54	2520
weighted avg	0.54	0.54	0.54	2520

Figure 4: *VGG16-1 Classification Report*

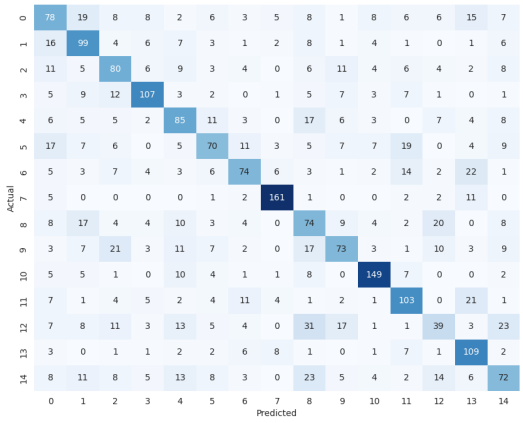

Figure 5: *VGG16-1 Confusion Matrix*

b. Fine-Tuning: We then fine-tuned the VGG16 model, unfreezing some of the higher-level layers to adapt to the human action recognition dataset. We compared the accuracy of the fine-tuned model with that of the transfer learning approach.

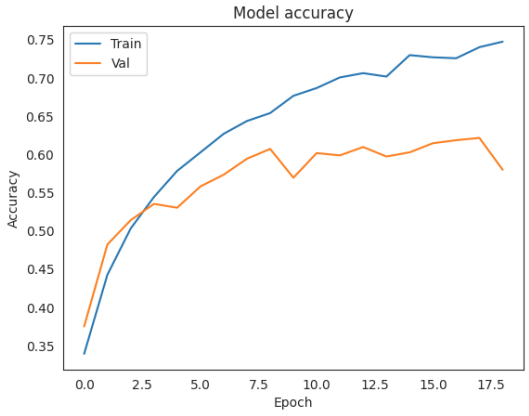

Figure 6: *VGG16-2 Accuracy*

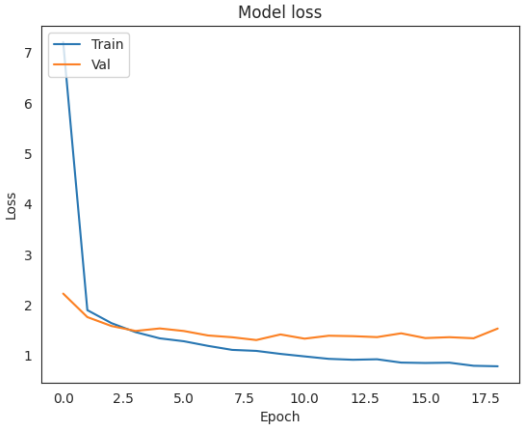

Figure 7: *VGG16-2 Loss*

79/79 [==============================] - 14s 170ms/step
VGG16 Model Accuracy with Fine-Tuning: 60.24%
Classification Report:

	precision	recall	f1-score	support
0.0	0.37	0.62	0.46	180
1.0	0.49	0.75	0.59	159
2.0	0.67	0.51	0.58	159
3.0	0.72	0.66	0.69	163
4.0	0.59	0.56	0.57	162
5.0	0.56	0.54	0.55	170
6.0	0.62	0.58	0.59	153
7.0	0.90	0.89	0.89	185
8.0	0.45	0.40	0.43	167
9.0	0.53	0.60	0.57	170
10.0	0.87	0.84	0.85	193
11.0	0.64	0.59	0.62	167
12.0	0.46	0.41	0.43	166
13.0	0.77	0.70	0.73	144
14.0	0.64	0.37	0.47	182
accuracy			0.60	2520
macro avg	0.62	0.60	0.60	2520
weighted avg	0.62	0.60	0.60	2520

Figure 8: *VGG16-2 Classification Report*

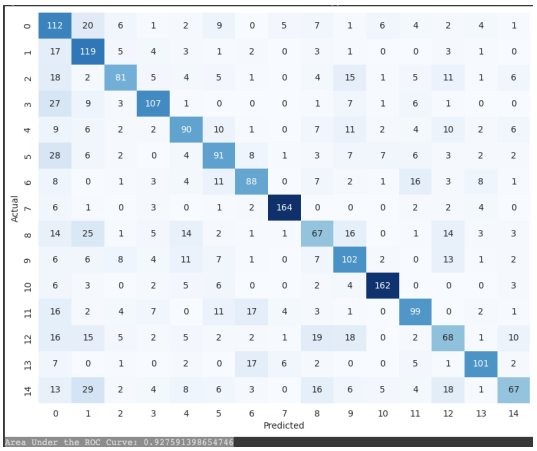

Figure 9: *VGG16-2 Confusion Matrix*

c. Fine-Tuning with Adaptive Learning Rate: We

introduced an adaptive learning rate in the fine-tuned VGG16 model using the ReduceLROnPlateau callback. We analyzed the impact of this adaptive learning rate on model convergence and overall accuracy.

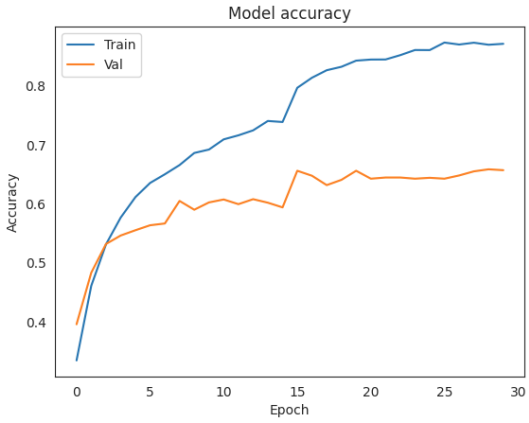

Figure 10: *VGG16-3 Accuracy*

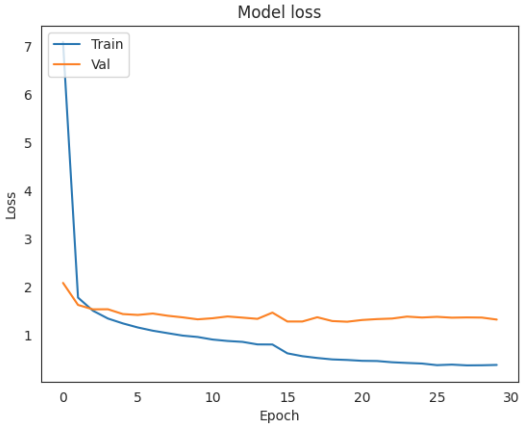

Figure 11: *VGG16-3 Loss*

```
79/79 [==============================] - 14s 172ms/step
VGG16 Model Accuracy with Fine-Tuning: 64.64%
Classification Report:

              precision    recall  f1-score   support

         0.0       0.50      0.52      0.51       180
         1.0       0.65      0.54      0.59       159
         2.0       0.62      0.76      0.68       159
         3.0       0.76      0.66      0.71       163
         4.0       0.59      0.60      0.60       162
         5.0       0.65      0.65      0.65       170
         6.0       0.67      0.64      0.66       153
         7.0       0.92      0.88      0.90       185
         8.0       0.46      0.54      0.50       167
         9.0       0.67      0.62      0.64       170
        10.0       0.85      0.85      0.85       193
        11.0       0.70      0.68      0.69       167
        12.0       0.43      0.43      0.43       166
        13.0       0.69      0.78      0.73       144
        14.0       0.58      0.51      0.54       182

    accuracy                           0.65      2520
   macro avg       0.65      0.65      0.64      2520
weighted avg       0.65      0.65      0.65      2520
```

Figure 12: *VGG16-3 Classification Report*

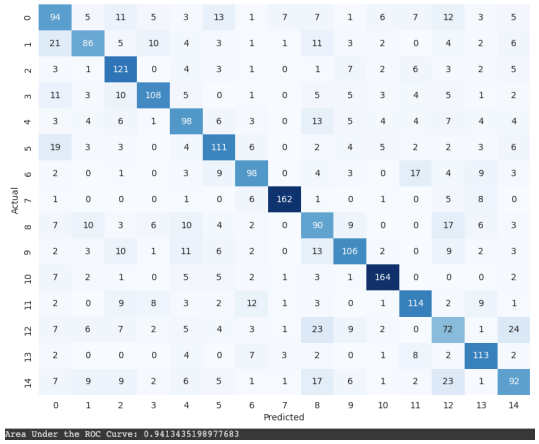

Figure 13: *VGG16-3 Confusion Matrix*

ResNet Experiments:

a. Transfer Learning: We conducted a similar experiment with the ResNet model, training it using transfer learning and assessing its accuracy on the validation set.

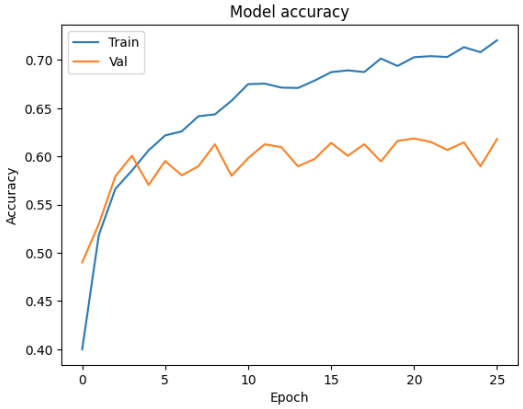

Figure 14: *ResNet-1 Accuracy*

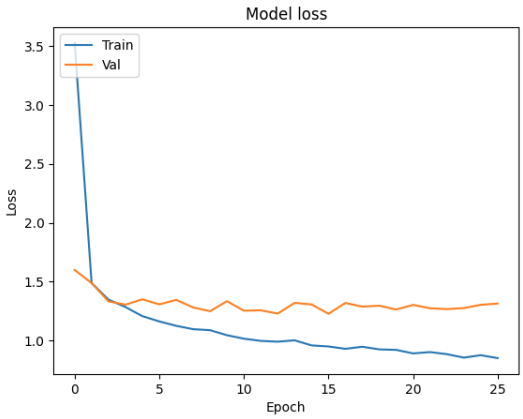

Figure 15: *ResNet-1 Loss*

```
79/79 [==============================] - 15s 178ms/step
VGG16 Model Accuracy without Fine-Tuning: 61.03%
Classification Report:

              precision    recall  f1-score   support

         0.0       0.51      0.58      0.54       171
         1.0       0.74      0.51      0.60       167
         2.0       0.60      0.60      0.60       150
         3.0       0.76      0.73      0.74       178
         4.0       0.58      0.66      0.62       167
         5.0       0.70      0.61      0.65       180
         6.0       0.89      0.46      0.61       166
         7.0       0.96      0.84      0.89       183
         8.0       0.35      0.48      0.41       173
         9.0       0.63      0.68      0.65       182
        10.0       0.80      0.70      0.75       176
        11.0       0.56      0.73      0.63       128
        12.0       0.39      0.34      0.36       170
        13.0       0.61      0.84      0.71       160
        14.0       0.39      0.40      0.40       169

    accuracy                           0.61      2520
   macro avg       0.63      0.61      0.61      2520
weighted avg       0.63      0.61      0.61      2520
```

Figure 16: *ResNet-1 Classification Report*

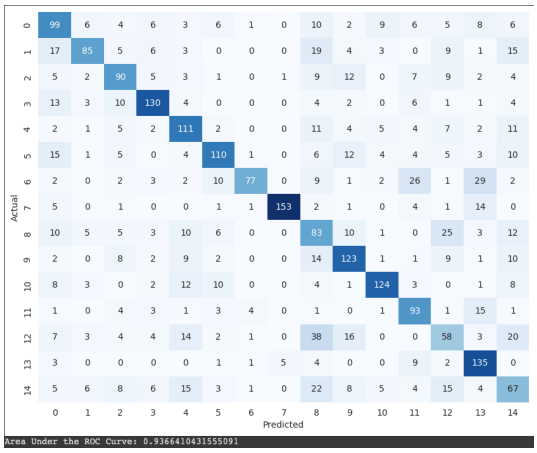

Figure 17: *ResNet-1 Confusion Matrix*

b. Fine-Tuning: We fine-tuned the ResNet model and compared its performance with the transfer learning approach. This comparison allowed us to determine which training strategy was more effective with the ResNet architecture.

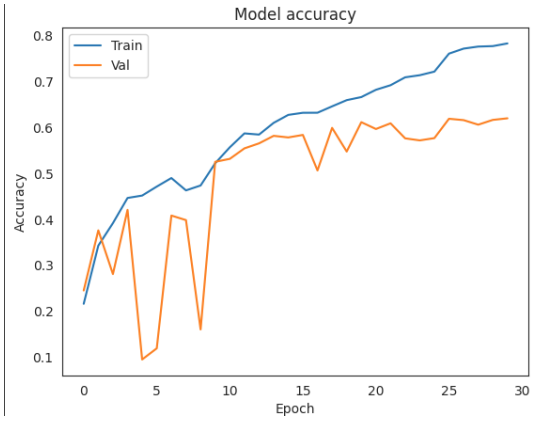

Figure 18: *ResNet-2 Accuracy*

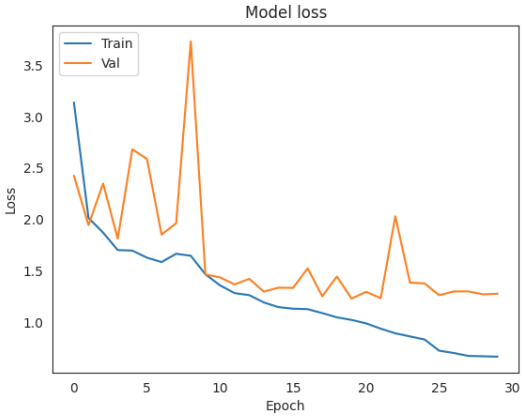

Figure 19: *ResNet-2 Loss*

```
VGG16 Model Accuracy with Fine-Tuning: 61.43%
Classification Report:

              precision    recall  f1-score   support

         0.0       0.52      0.42      0.47       171
         1.0       0.48      0.51      0.50       167
         2.0       0.62      0.73      0.67       150
         3.0       0.69      0.73      0.71       178
         4.0       0.52      0.51      0.52       167
         5.0       0.69      0.54      0.61       180
         6.0       0.57      0.61      0.59       166
         7.0       0.88      0.92      0.90       183
         8.0       0.52      0.49      0.50       173
         9.0       0.81      0.66      0.73       182
        10.0       0.74      0.81      0.77       176
        11.0       0.57      0.57      0.57       128
        12.0       0.57      0.42      0.48       170
        13.0       0.64      0.80      0.71       160
        14.0       0.39      0.47      0.42       169

    accuracy                           0.61      2520
   macro avg       0.61      0.61      0.61      2520
weighted avg       0.62      0.61      0.61      2520
```

Figure 20: *ResNet-2 Classification Report*

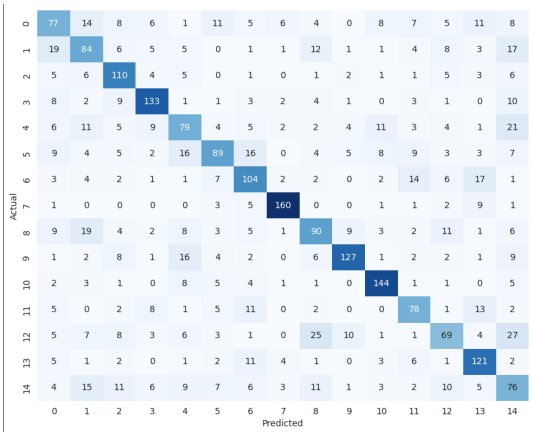

Figure 21: *ResNet-2 Confusion Matrix*

VGG19 Experiment:

a. Fine-Tuning with Best Parameters: Based on the insights gained from the previous experiments, we fine-tuned the VGG19 model using the best parameters from the VGG16 and ResNet experiments. This approach aimed to maximize the performance of the VGG19 model on the human action recognition task.

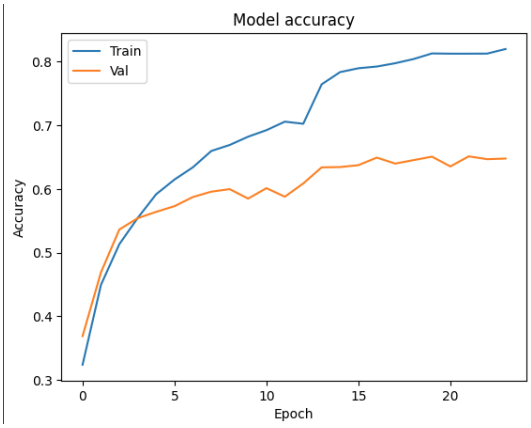

Figure 22: *VGG19 Accuracy*

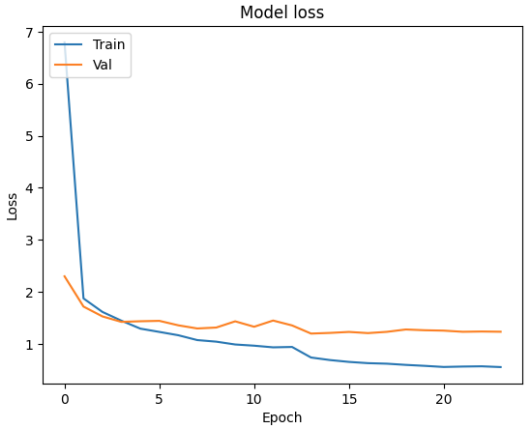

Figure 23: *VGG19 Loss*

VGG16 Model Accuracy with Fine-Tuning: 62.74%
Classification Report:

	precision	recall	f1-score	support
0.0	0.52	0.45	0.48	176
1.0	0.53	0.59	0.56	158
2.0	0.63	0.74	0.68	151
3.0	0.69	0.74	0.71	172
4.0	0.66	0.61	0.63	171
5.0	0.61	0.56	0.59	176
6.0	0.61	0.61	0.61	163
7.0	0.93	0.87	0.90	172
8.0	0.43	0.47	0.45	157
9.0	0.62	0.69	0.65	154
10.0	0.83	0.82	0.82	169
11.0	0.66	0.63	0.65	174
12.0	0.45	0.46	0.46	165
13.0	0.74	0.74	0.74	196
14.0	0.46	0.39	0.42	166
accuracy			0.63	2520
macro avg	0.63	0.63	0.62	2520
weighted avg	0.63	0.63	0.63	2520

Figure 24: *VGG19 Classification Report*

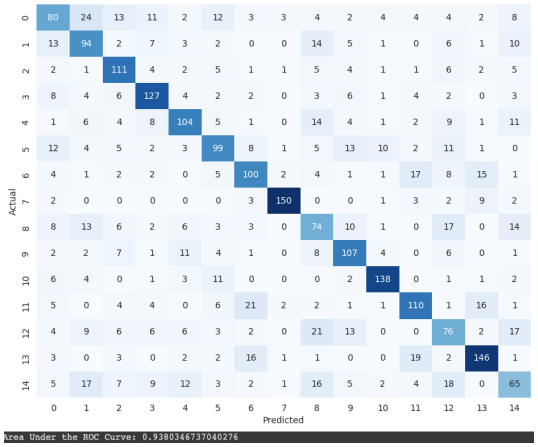

Figure 25: *VGG19 Confusion Matrix*

Comparative Analysis: The VGG16 model with fine-tuning and adaptive learning rate (VGG16-3) performed best, with an accuracy of 64.64 percent and the highest AUC of 0.941, suggesting superior class distinction. The VGG19 model with similar enhancements came in second at 62.74 percent accuracy. The base ResNet model achieved 61.03 percent accuracy, slightly improved to 61.43 percent with fine-tuning and adaptive learning rate. VGG16 models without these enhancements had lower accuracy scores. Fine-tuning and adaptive learning rate seem to enhance model performance. However, all models varied in precision, recall, and f1-score across classes, suggesting room for improvement in class prediction. While VGG16-3 performed best overall, consider other factors like computational cost and project requirements when choosing a model.

Model Name	Accuracy (%)	AUC-ROC
VGG16-1 (Base)	54.48	0.9095
VGG16-2 (Fine-Tuned)	60.24	0.9276
VGG16-3 (Fine-Tuned + Adaptive LR)	64.64	0.9413
ResNet-1 (Base)	61.03	0.9366
ResNet-2 (Fine-Tuned + Adaptive LR)	61.43	0.9318
VGG19 (Fine-Tuned + Adaptive LR)	62.74	0.938

Figure 26: *Comparative Analysis: Accuracy and AUC-ROC*

Model Name	Avg Precision	Avg Recall	Avg F1-Score
VGG16-1 (Base)	0.54	0.54	0.54
VGG16-2 (Fine-Tuned)	0.62	0.6	0.6
VGG16-3 (Fine-Tuned + Adaptive LR)	0.65	0.65	0.64
ResNet-1 (Base)	0.63	0.61	0.61
ResNet-2 (Fine-Tuned + Adaptive LR)	0.61	0.61	0.61
VGG19 (Fine-Tuned + Adaptive LR)	0.63	0.63	0.62

Figure 27: *Comparative Analysis: Precision, Recall F1-Scores*

Insights and Speculation on Results: The improve-

ment of the different models' performance due to adding fine-tuning and adaptive learning rates were in line with expectations and the following general insights over the course of the project was gained:

Effectiveness of Fine-tuning: The models with fine-tuning generally outperformed their counterparts without fine-tuning. This indicates that fine-tuning can effectively leverage pre-trained models to enhance performance on specific tasks.

Adaptive Learning Rates: Models with adaptive learning rates (VGG16-3, VGG19-3, and ResNet-2) performed better. This suggests that adaptive learning rates can help optimize the learning process and lead to better performance.

Importance of AUC: Despite variations in accuracy, all models showed substantial AUC, highlighting that models can still perform well in distinguishing classes even if their overall accuracy is not as high.

However, the better performance results of the VGG16 model against the VGG19 model were surprising. Our speculations on this result are as follows:

Overfitting: VGG19's complexity may have led it to learn noise from training data, reducing its generalization ability.

Computational limits: Insufficient resources could prevent VGG19 from reaching its full potential.

Task requirements: The problem might not need a model as deep as VGG19; VGG16's depth could be sufficient.

Optimization challenges: Deeper networks like VGG19 can be harder to optimize, affecting performance.

Conclusion

In conclusion, this project explored the performance of three widely-used convolutional neural network architectures - VGG16, VGG19, and ResNet50 - in the context of image classification. Six experimental setups were evaluated, considering the influence of fine-tuning and adaptive learning rates on the performance of these models. Despite its greater complexity, VGG19 did not outperform VGG16, underlining that a more complex model does not necessarily guarantee better performance and may sometimes lead to issues such as overfitting. It is also essential to consider the computational cost and training time associated with complex models like VGG19 and ResNet50. Future work could involve investigating other architectures, optimization techniques, and approaches to improve class-specific performance.

References

- https://towardsdatascience.com/understand-and-implement-resnet-50-with-tensorflow-2-0-1190b9b52691
- https://www.kaggle.com/code/meetnagadia/har-vgg/input
- Simonyan, K., Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
- He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
- https://keras.io/api/applications/