

University of Calgary

ENEL 525 Fall 2024 - Final Project

Daniel Le

December 2024

Table of Contents

Introduction.....	3
Methodology.....	3
Problem Description.....	3
Data Utilization.....	3
Network Design.....	3
Simple Convolutional Model:.....	4
Paired Convolutional Model:.....	4
Transfer Learning with VGG16:.....	5
Transfer Learning with Enhanced Data Augmentation:.....	5
Transfer Learning with Additional Convolutional Layers:.....	6
Backpropagation Concept.....	7
Derivation of Loss function.....	7
Training Scheme.....	8
Testing Scheme.....	8
Results.....	9
Method Results.....	9
Single Convolutional Layer.....	9
Paired Convolutional Layer.....	13
Transfer Learning with VGG16.....	17
Transfer Learning with Aggressive Data Augmentation.....	20
Transfer Learning with extra Custom Layers.....	23
Final Model Results.....	27
Discussion.....	28
Analysis and Explanation.....	28
Initial Development.....	28
Performance Increase.....	29
Effectiveness.....	29
Conclusion.....	29
References.....	31

Introduction

The objective for this final project is to create a convolutional neural network (CNN) that is able to classify aerial images into 21 different classes depending on the land use shown in the image. The dataset is UC Merced Land Use Dataset [1], which contains 1000 images, 100 images per class, with classes being agricultural, airplane, baseball diamond, beach, and more..

In the project, I tested various methods for constructing my neural network for image classification, focusing on the feature extraction from 256x256 pixel aerial imagery. This report will go over their results and what I chose to be my final submission. It will utilize deep-learning, backpropagation, and transfer learning as its main architecture to achieve high accuracy in distinguishing between different categories.

Methodology

Problem Description

The problem this project is addressing is developing an effective convolutional neural network (CNN) for classifying aerial images into 21 land use categories. Utilizing the UC Merced Land Use Dataset [1], which contains a total of 2100 images equally distributed among the 21 classes, I aim to use machine learning and deep learning techniques learned through the course to achieve high classification accuracy.

Data Utilization

For this project, the entire UC Merced Land Use Dataset [1] was used. This dataset provides a diverse range of high-resolution aerial images, each representing a different type of land use, such as agricultural fields, runways, baseball diamonds, beaches, and more.

Network Design

To tackle this classification task, multiple CNN architectures and techniques were explored. All designs share similar parameters for the convolution layers and the same output layer. Each convolution layer uses a kernel size of 3x3 and a pooling size of 2x2, with ReLU as the activation function. These parameters were chosen due to their common practice in many CNN designs and deep learning frameworks. In the output layer, the data is flattened and then passed through three dense layers: the first with 512 units, the second with 256 units (both using ReLU activation), and the final layer with 21 units using Softmax for classification.

All approaches followed course video lessons and the “Training a neural network on MNIST with Keras” [2] Example by Tensorflow and differed in the model for each approach. Each

approach also used Data Augmentation in their training, due to problems of overfitting which will be explained later on.

Here is a description of each approaches differences:

Simple Convolutional Model:

- Architecture:
 - Conv Layers: Single convolutional layers with filters of 32, 64, and 128, each followed by a pooling layer.

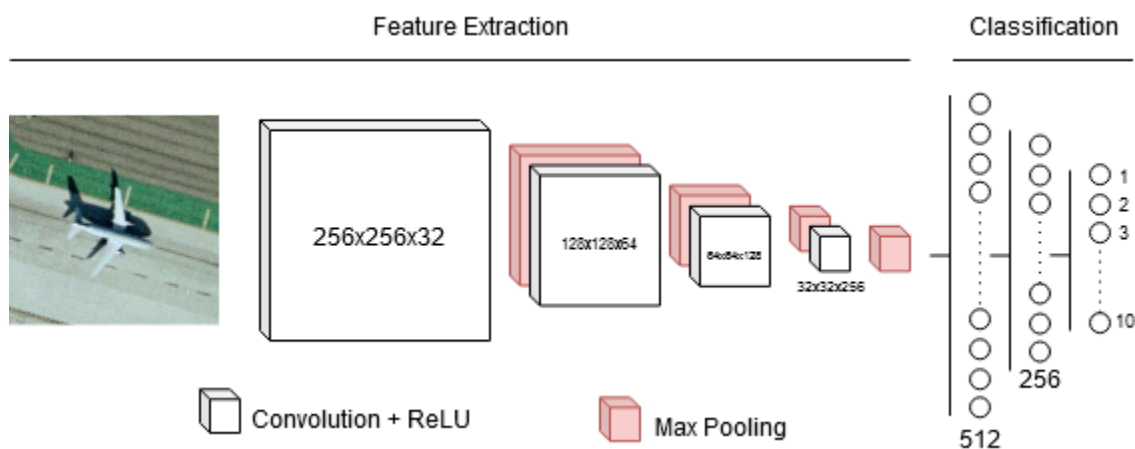


Figure 1. Simple Method Convolution Neural Network Diagram

Paired Convolutional Model:

- Architecture:
 - Conv Layers: Three pairs of convolutional layers with filters set to 32, 64, and 128, respectively. Each pair is followed by a pooling layer.

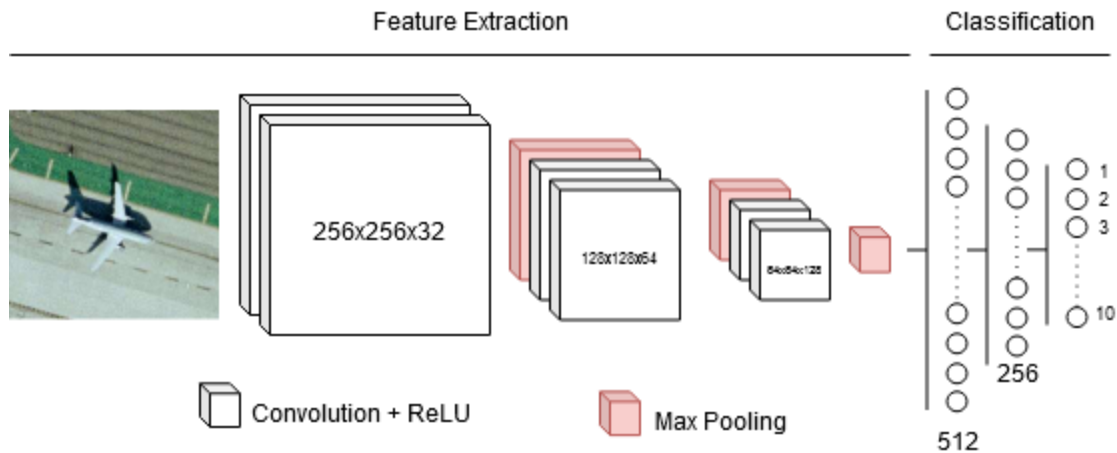


Figure 2. Paired Method Convolution Neural Network Diagram

Transfer Learning with VGG16:

- Architecture:
 - Base Model: Utilized the pre-trained VGG16 model, excluding the fully connected layers
 - Custom Top Layers: Own output layer mentioned above

Transfer Learning with Enhanced Data Augmentation:

- Architecture:
 - Base Model: Similar to the Transfer Learning Method
 - Aggressive Data Augmentation in training

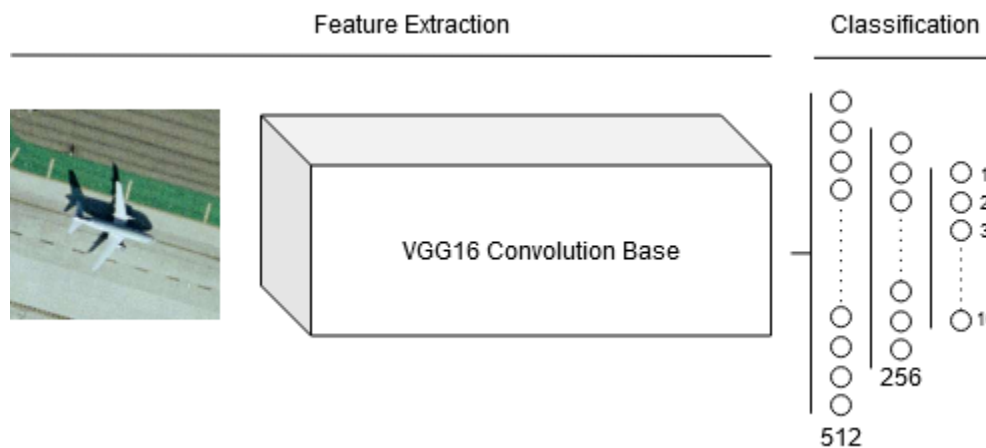


Figure 3. Transfer Learning Convolutional Neural Network Diagram

Transfer Learning with Additional Convolutional Layers:

- Architecture:
 - Base Model: Pre-trained VGG16 model as the feature extractor.
 - Extra Conv Layers: Added convolutional layers with filters 128 and 256, followed by pooling layers, before the dense layers.

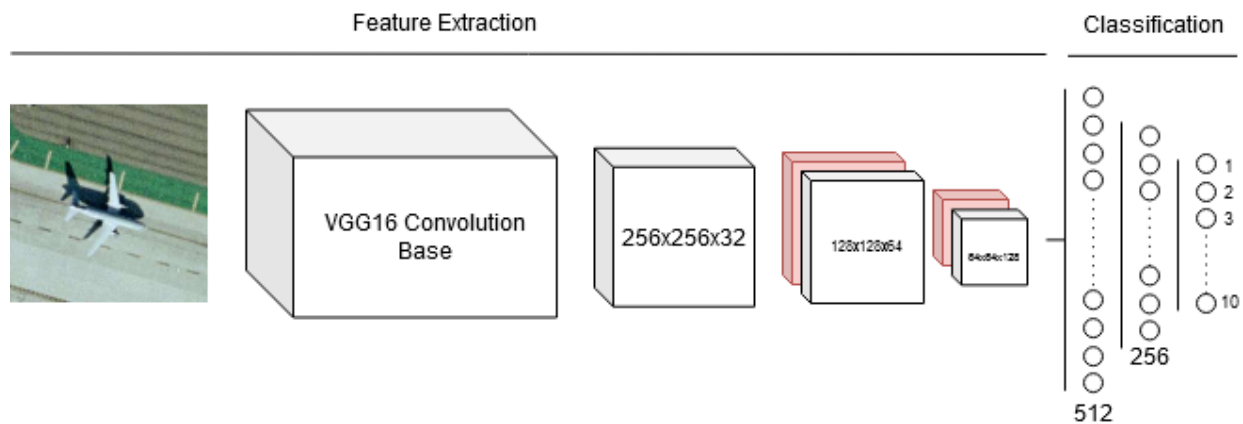


Figure 4. Transfer Learning with addition Convolutional Layers



Figure 5. VGG16 Convolution Base based on [6]

Backpropagation Concept

Backpropagation is a mechanism that allows CNNs to learn from data. The process begins with a forward pass, where input data is processed through the network to generate predictions. The loss is then calculated with a loss function that takes the error between predictions and actual targets. During the backward pass, gradients are computed and propagated backwards through the CNN, allowing the optimizer to update the weights to minimize the loss.

Derivation of Loss function

The Categorical Cross-Entropy loss function is used for my multi-class classification. This function measures the difference between the labels and the predicted probabilities output by the network. This loss function involves representing the labels as one-hot encoded vectors, where each vector indicates the correct class with a 1 and 0s for incorrect [3]. The neural network's final layer uses the softmax function to produce a probability distribution over the 21 classes. The cross-entropy loss formula calculates the loss by summing the negative log probabilities of the predicted values for the true class labels [4]. During training, the loss function quantifies how well the predictions match the true labels, and the gradients of the loss with respect to the weights are computed and used to update the network's weights [5].

The formula used in the Categorical Cross-Entropy Calculation is:

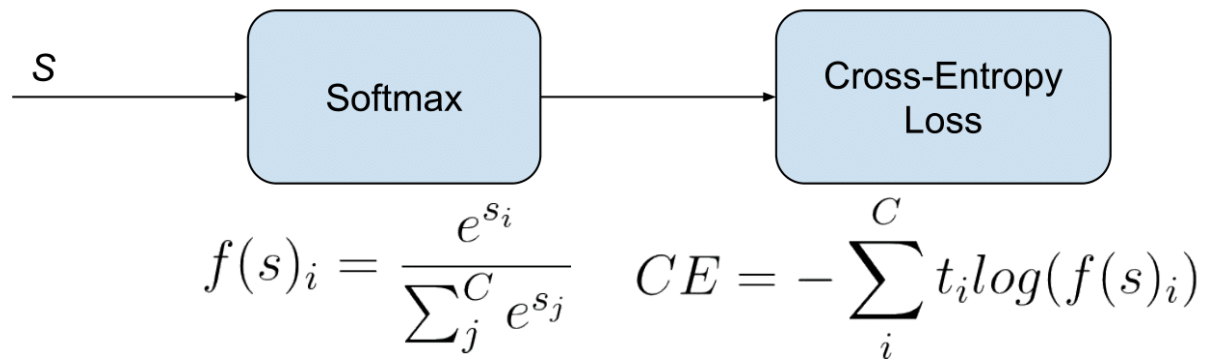


Figure 6. Cross-Entropy Loss https://gombru.github.io/2018/05/23/cross_entropy_loss/ [4]

Training Scheme

For training, 70% of the dataset was used, another 15% used for validation. Each run of the code had the training and validation data randomized, with Data Augmentation occurring during every epoch in the training process. This was done so each time the code ran would be giving results based on generalized varying data, not a set dataset. Augmentation was done so the model would not overfit during training. The Adam optimizer was used, with a learning rate of 0.0001. The main concern for determining the number of epochs was time to train as well as the diminishing return of how long the model took to train for the performance benefits. Each method had a different number of epochs with the goal of approximately 30 minutes training time and/or until the accuracy/loss no longer drastically improved.

Table 1

Method	Number of Epochs
Single Convolutional Layer	70
Paired Convolutional Layer	35
Transfer Learning	10
Transfer Learning with Aggressive Data Augmentation	20
Transfer Learning with extra Custom Layers	10

Testing Scheme

To test each of the models, 15% of the dataset was used. To test the final model the entire dataset was used to see the accuracy of classifying all 2100 images. No outside images were used in testing.

Results

Method Results

Single Convolutional Layer

Data Augmentation:

rotation_range=45,
width_shift_range=0.2,
height_shift_range=0.2,
zoom_range=0.2,
horizontal_flip=True,
vertical_flip=True,
fill_mode='nearest'

Table 2

Final Epoch Training Loss	0.4173
Final Epoch Validation Loss	0.7388
Number of Epochs	70
Learning Rate	0.0001
Optimizer	Adam
Loss Function	Categorical Cross Entropy
Number of Conv Layers	4
Training Accuracy	86.22%
Validation Accuracy	77.78%
Testing Accuracy	72.70%

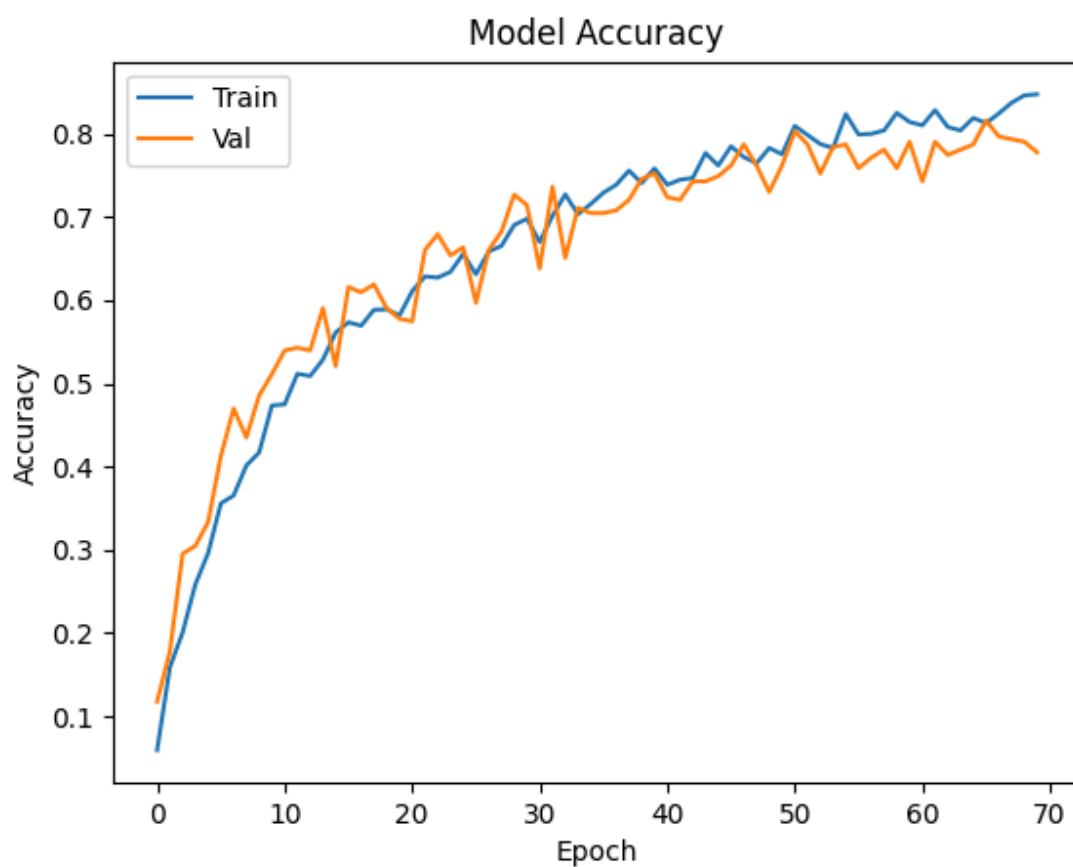


Figure 7. Training Accuracy Curve

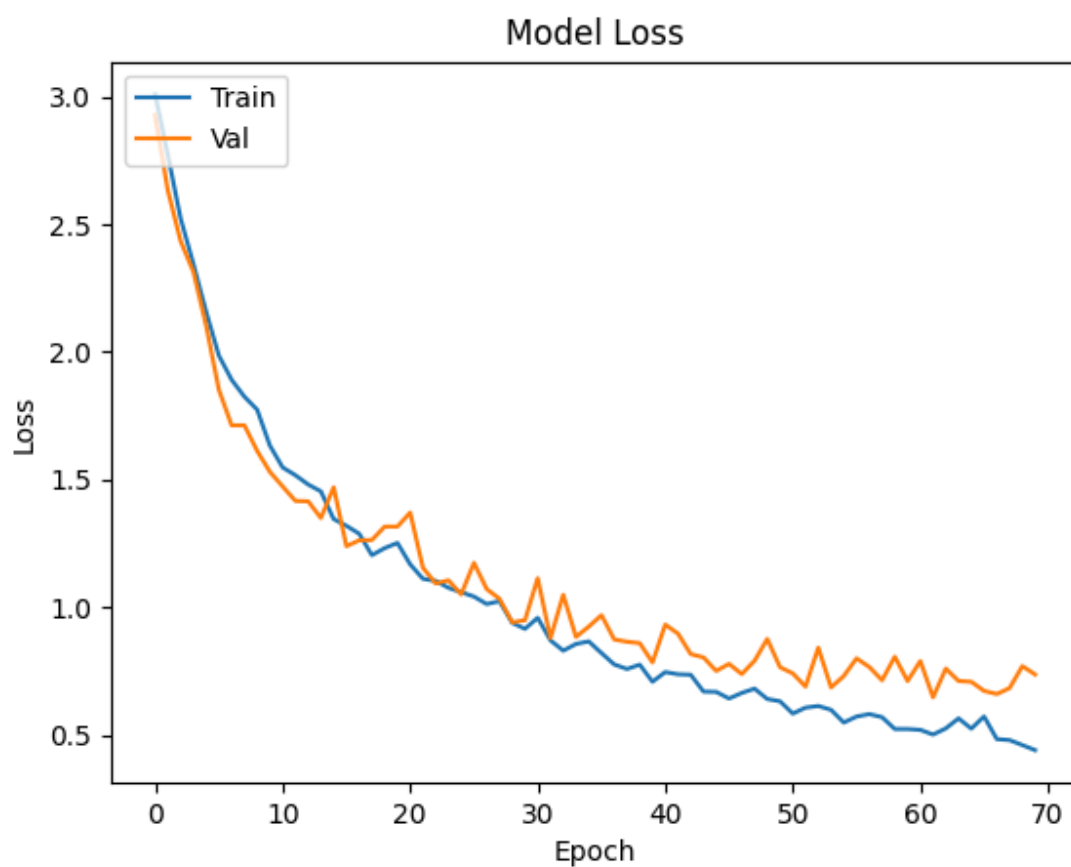


Figure 8. Training Loss Curve

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 256, 256, 32)	896
max_pooling2d (MaxPooling2D)	(None, 128, 128, 32)	0
conv2d_1 (Conv2D)	(None, 128, 128, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 64)	0
conv2d_2 (Conv2D)	(None, 64, 64, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 128)	0
conv2d_3 (Conv2D)	(None, 32, 32, 256)	295,168
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 256)	0
flatten (Flatten)	(None, 65536)	0
dropout (Dropout)	(None, 65536)	0
dense (Dense)	(None, 512)	33,554,944
dense_1 (Dense)	(None, 256)	131,328
dense_2 (Dense)	(None, 21)	5,397

Figure 9. Model Summary


Epoch 70/70
 46/46  26s 571ms/step - categorical_accuracy: 0.8622 - loss: 0.4173 - val_categorical_accuracy: 0.7778 - val_loss: 0.7388

Figure 10. Final Epoch Results

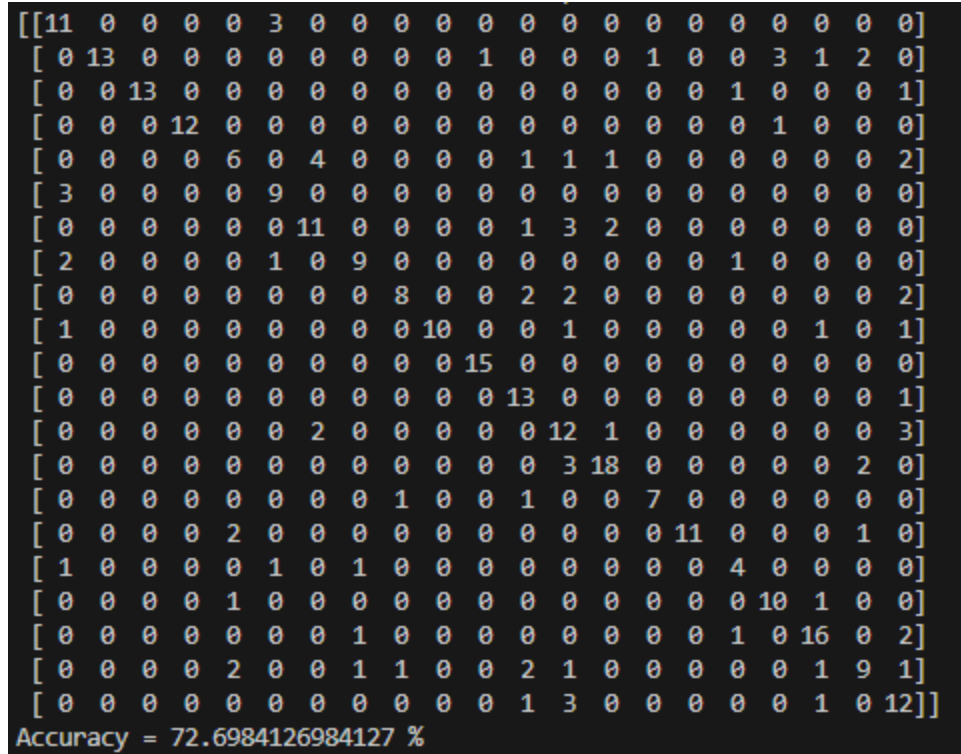


Figure 11. Confusion Matrix

Paired Convolutional Layer

Data Augmentation:

rotation_range=45,
width_shift_range=0.2,
height_shift_range=0.2,
zoom_range=0.2,
horizontal_flip=True,
vertical_flip=True,
fill_mode='nearest'

Table 3

Final Epoch Loss	0.8460
Final Epoch Validation Loss	0.9239
Number of Epochs	35
Learning Rate	0.0001
Optimizer	Adam

Loss Function	Categorical Cross Entropy
Number of Conv Layers	6
Training Accuracy	70.78%
Validation Accuracy	72.70%
Testing Accuracy	69.21%

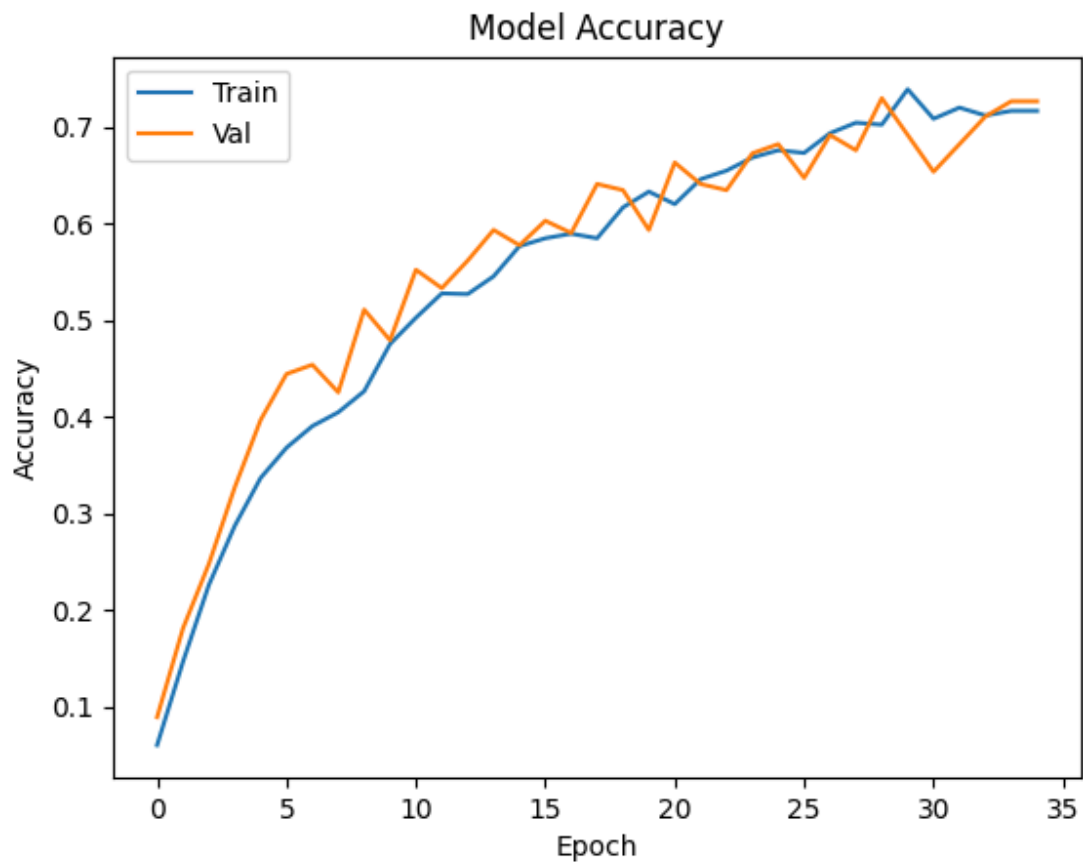


Figure 12. Training Accuracy Curve

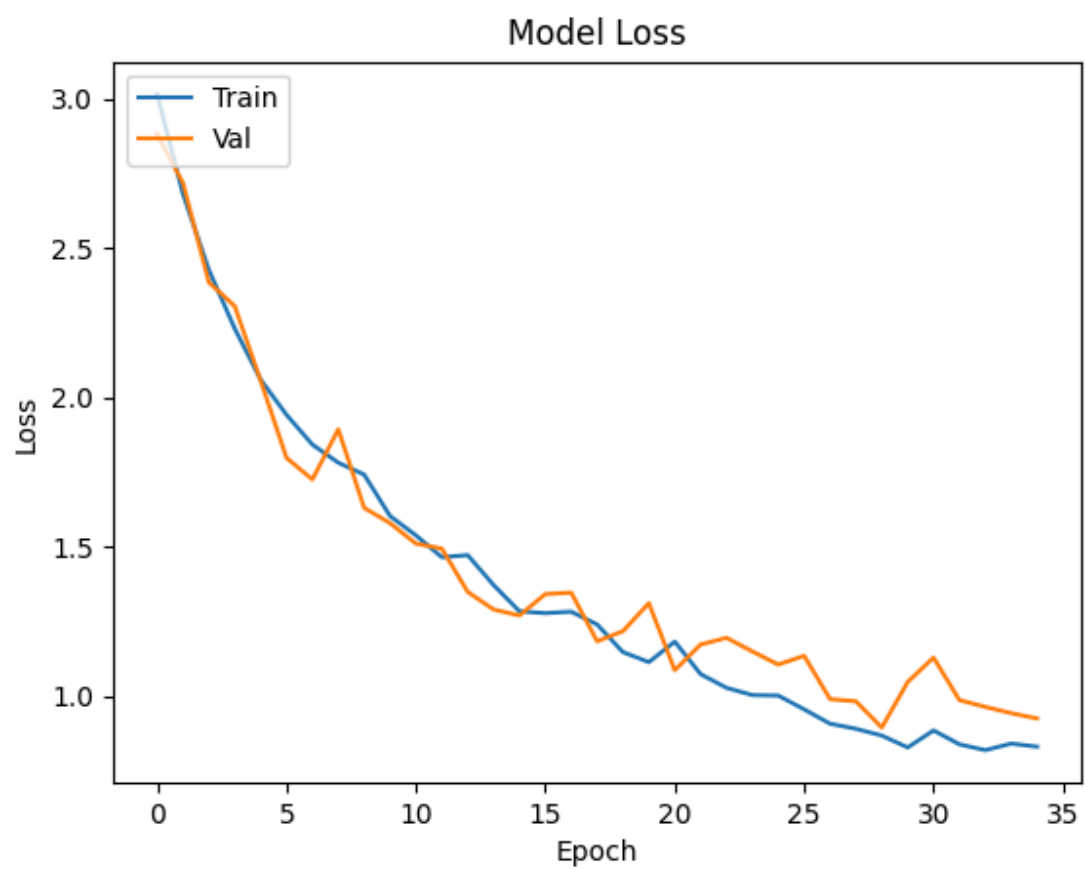


Figure 13. Training Loss Curve

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 256, 256, 32)	896
conv2d_1 (Conv2D)	(None, 256, 256, 32)	9,248
max_pooling2d (MaxPooling2D)	(None, 128, 128, 32)	0
conv2d_2 (Conv2D)	(None, 128, 128, 64)	18,496
conv2d_3 (Conv2D)	(None, 128, 128, 64)	36,928
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 64)	0
conv2d_4 (Conv2D)	(None, 64, 64, 128)	73,856
conv2d_5 (Conv2D)	(None, 64, 64, 128)	147,584
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 128)	0
flatten (Flatten)	(None, 131072)	0
dropout (Dropout)	(None, 131072)	0
dense (Dense)	(None, 512)	67,109,376
dense_1 (Dense)	(None, 256)	131,328
dense_2 (Dense)	(None, 21)	5,397

Figure 14. Model Summary

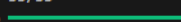
Epoch 35/35
 46/46  65s 1s/step - categorical_accuracy: 0.7078 - loss: 0.8460 - val_categorical_accuracy: 0.7270 - val_loss: 0.9239

Figure 15. Final Epoch Results

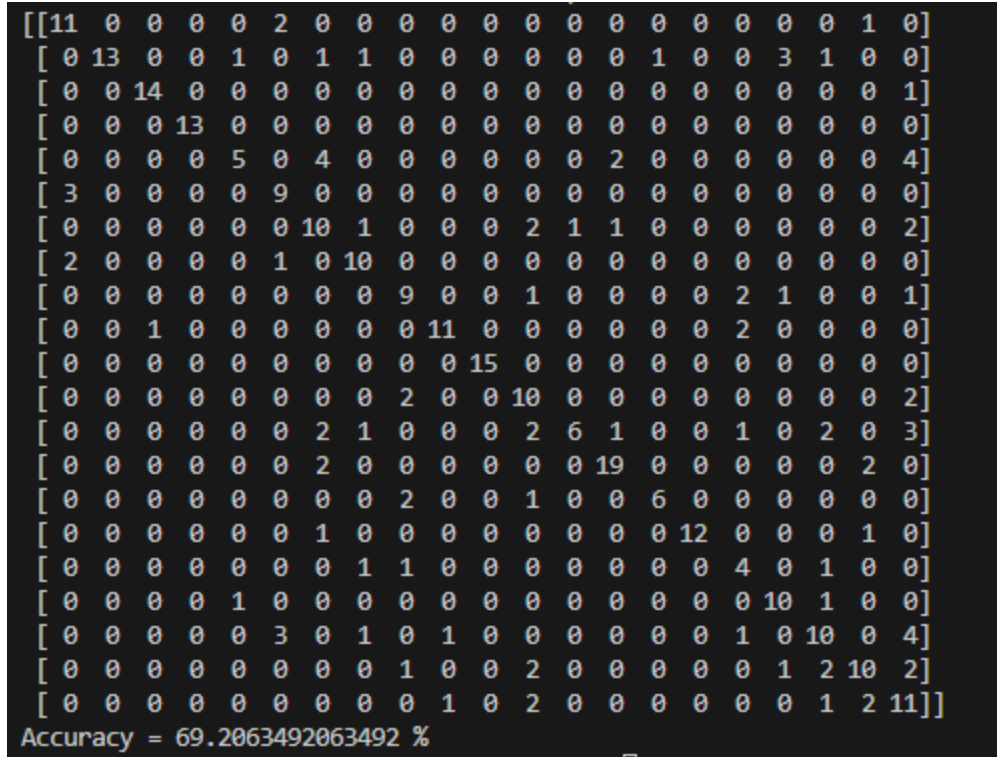


Figure 16. Confusion Matrix

Transfer Learning with VGG16

Data Augmentation:

rotation_range=45,
 width_shift_range=0.2,
 height_shift_range=0.2,
 zoom_range=0.2,
 horizontal_flip=True,
 vertical_flip=True,
 fill_mode='nearest'

Table 4

Final Epoch Loss	0.3390
Final Epoch Validation Loss	0.3218
Number of Epochs	10
Learning Rate	0.0001
Optimizer	Adam

Loss Function	Categorical Cross Entropy
Number of Conv Layers	13
Training Accuracy	88.71%
Validation Accuracy	90.16%
Testing Accuracy	84.76%

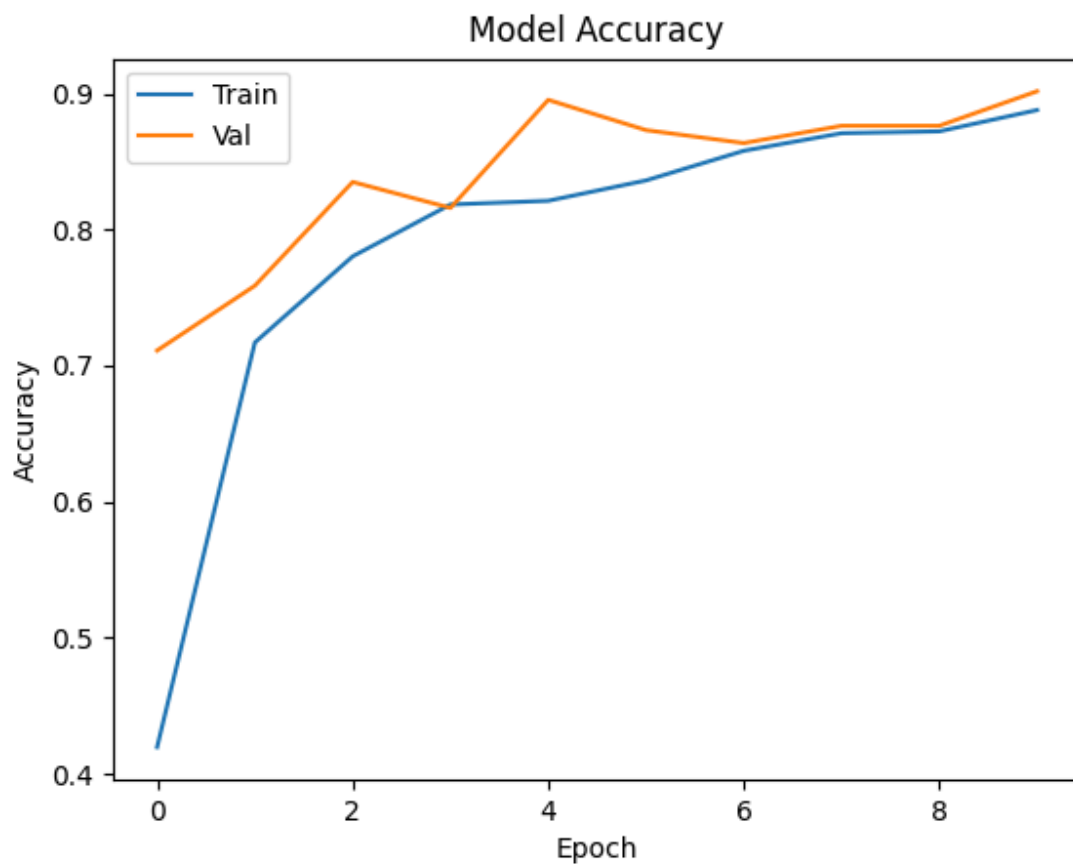


Figure 17. Training Accuracy Curve

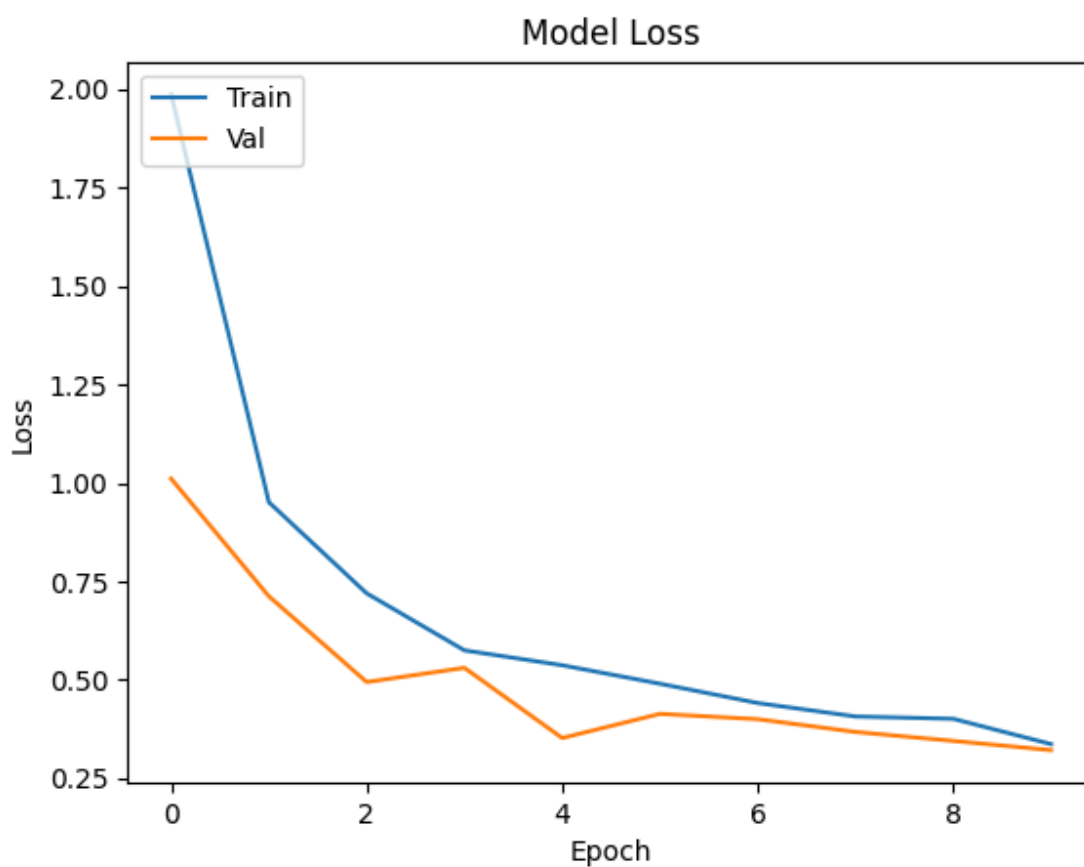


Figure 18. Training Loss Curve

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 8, 8, 512)	14,714,688
flatten (Flatten)	(None, 32768)	0
dropout (Dropout)	(None, 32768)	0
dense (Dense)	(None, 512)	16,777,728
dense_1 (Dense)	(None, 256)	131,328
dense_2 (Dense)	(None, 21)	5,397

Figure 19. Model Summary

Epoch 10/10
46/46 100s 2s/step - categorical_accuracy: 0.8871 - loss: 0.3390 - val_categorical_accuracy: 0.9016 - val_loss: 0.3218

Figure 20. Final Epoch Results

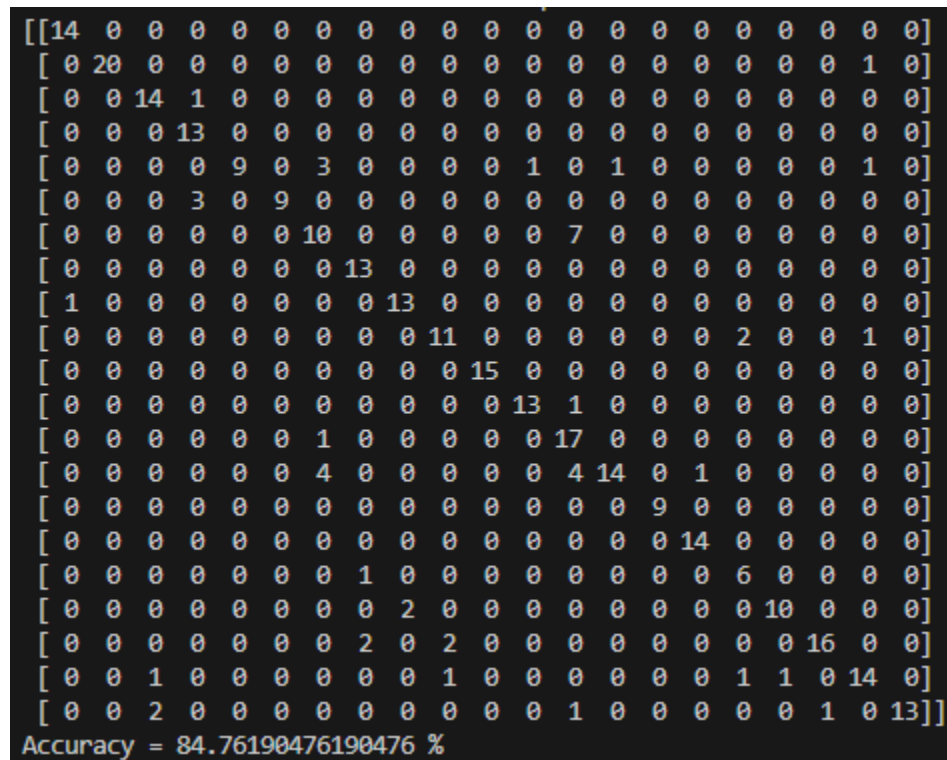


Figure 21. Confusion Matrix

Transfer Learning with Aggressive Data Augmentation

Data Augmentation:

```

rotation_range=90,
width_shift_range=0.3,
height_shift_range=0.3,
zoom_range=0.2,
shear_range=0.2,
horizontal_flip=True,
vertical_flip=True,
fill_mode='nearest'

```

Table 5

Final Epoch Loss	0.3306
Final Epoch Validation Loss	0.3536

Number of Epochs	20
Learning Rate	0.0001
Optimizer	Adam
Loss Function	Categorical Cross Entropy
Number of Conv Layers	13
Training Accuracy	90.34%
Validation Accuracy	88.57%
Testing Accuracy	85.71%

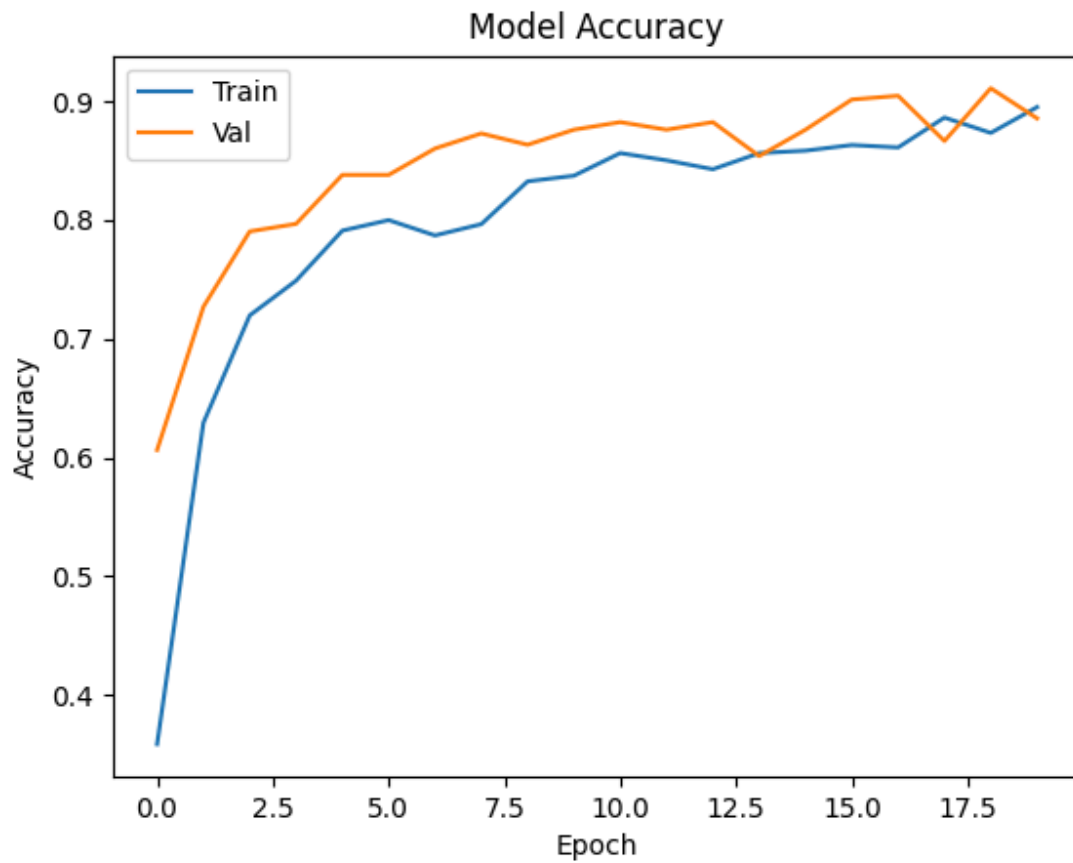


Figure 22. Training Accuracy Curve

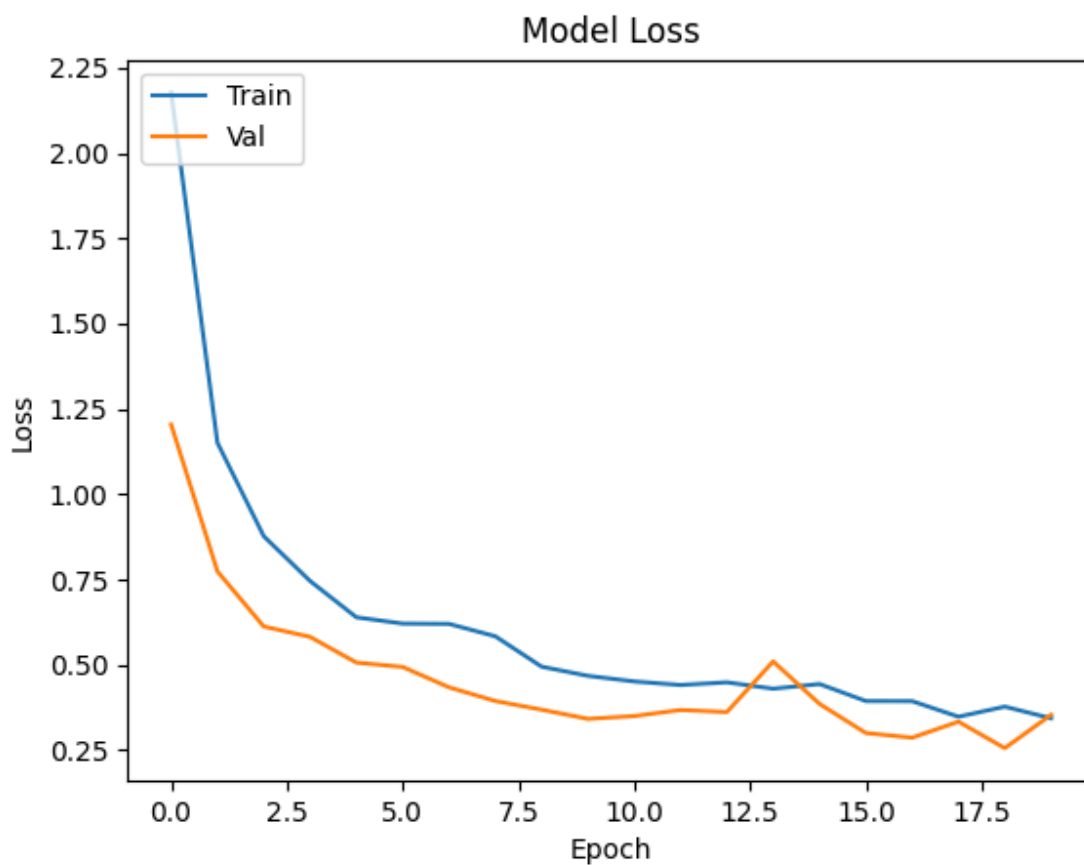


Figure 23. Training Loss Curve

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 8, 8, 512)	14,714,688
flatten (Flatten)	(None, 32768)	0
dropout (Dropout)	(None, 32768)	0
dense (Dense)	(None, 512)	16,777,728
dense_1 (Dense)	(None, 256)	131,328
dense_2 (Dense)	(None, 21)	5,397

Figure 24. Model Summary

Epoch 20/20
46/46 108s 2s/step - categorical_accuracy: 0.9034 - loss: 0.3306 - val_categorical_accuracy: 0.8857 - val_loss: 0.3536

Figure 25. Final Epoch Results

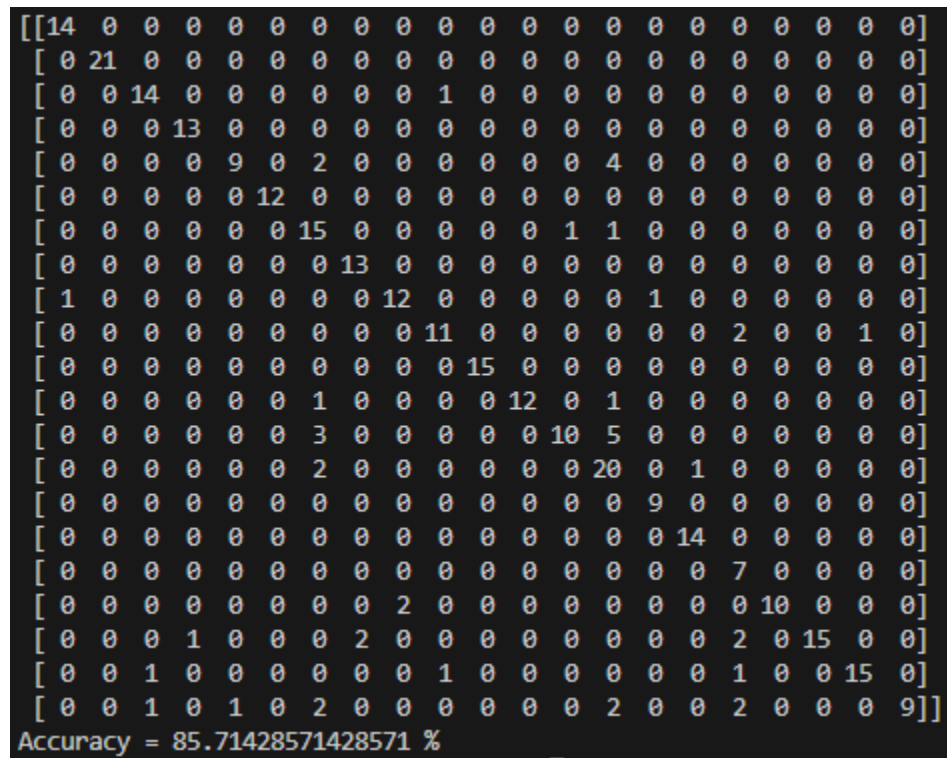


Figure 26. Confusion Matrix

Transfer Learning with extra Custom Layers

Data Augmentation:

rotation_range=45,
width_shift_range=0.2,
height_shift_range=0.2,
zoom_range=0.2,
horizontal_flip=True,
vertical_flip=True,
fill_mode='nearest'

Table 6

Final Epoch Loss	0.5892
Final Epoch Validation Loss	0.5318
Number of Epochs	10

Learning Rate	0.0001
Optimizer	Adam
Loss Function	Categorical Cross Entropy
Number of Conv Layers	15
Training Accuracy	79.93%
Validation Accuracy	81.59%
Testing Accuracy	80.32%

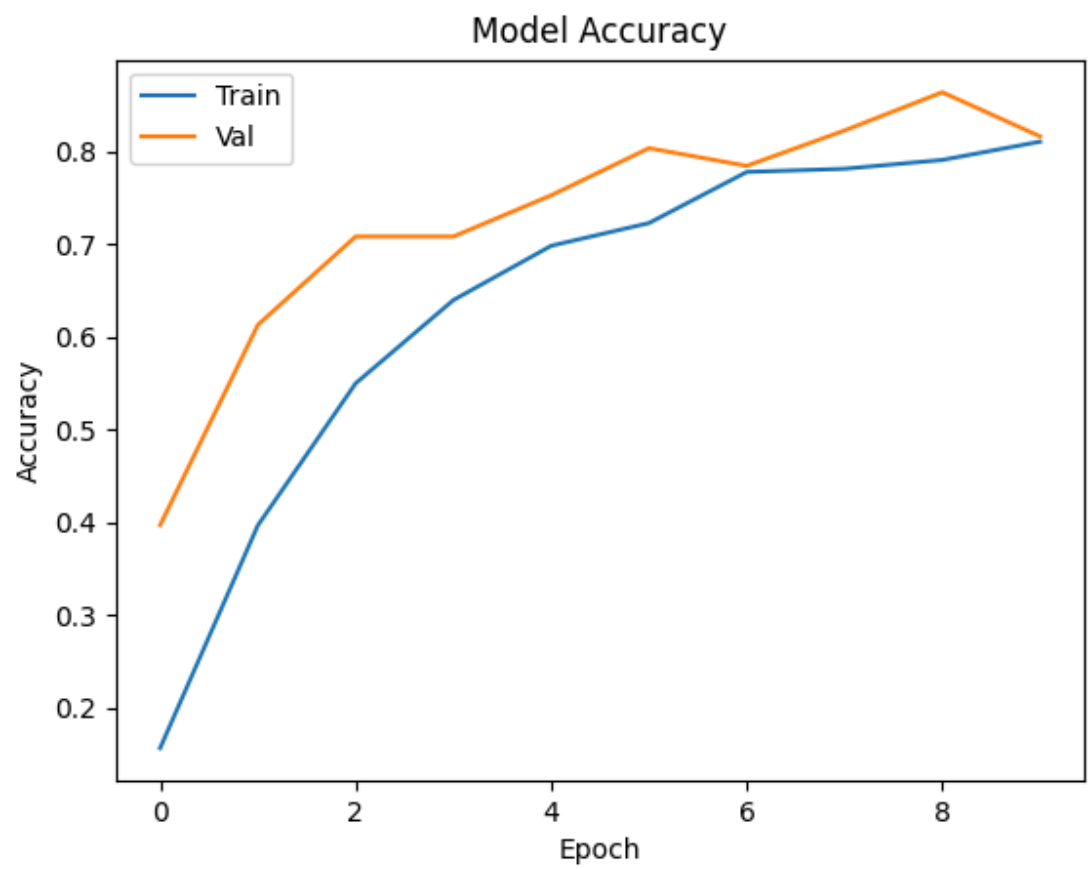


Figure 27. Training Accuracy Curve

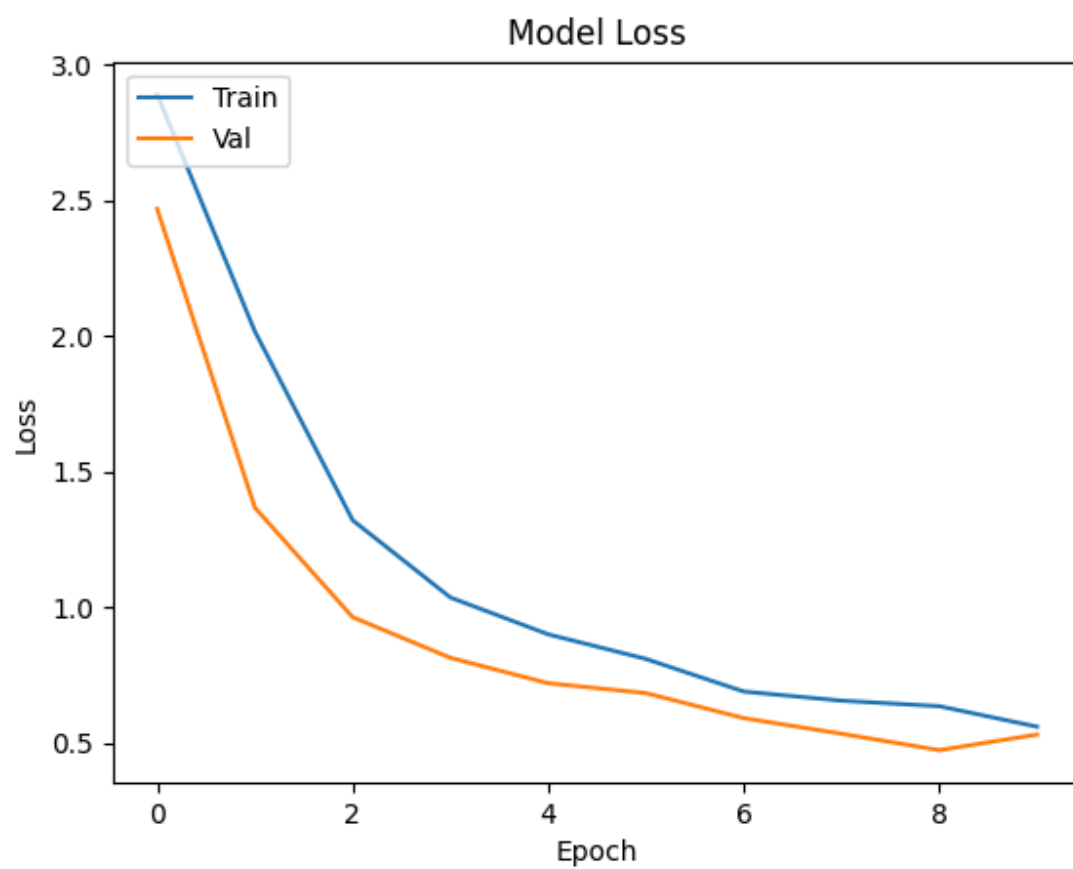


Figure 28. Training Loss Curve

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 8, 8, 512)	14,714,688
conv2d (Conv2D)	(None, 8, 8, 128)	589,952
max_pooling2d (MaxPooling2D)	(None, 4, 4, 128)	0
conv2d_1 (Conv2D)	(None, 4, 4, 256)	295,168
max_pooling2d_1 (MaxPooling2D)	(None, 2, 2, 256)	0
flatten (Flatten)	(None, 1024)	0
dropout (Dropout)	(None, 1024)	0
dense (Dense)	(None, 512)	524,800
dense_1 (Dense)	(None, 256)	131,328
dense_2 (Dense)	(None, 21)	5,397

Figure 29. Model Summary

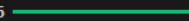
Epoch 10/10
 46/46  111s 2s/step - categorical_accuracy: 0.7993 - loss: 0.5892 - val_categorical_accuracy: 0.8159 - val_loss: 0.5318

Figure 30. Final Epoch Results

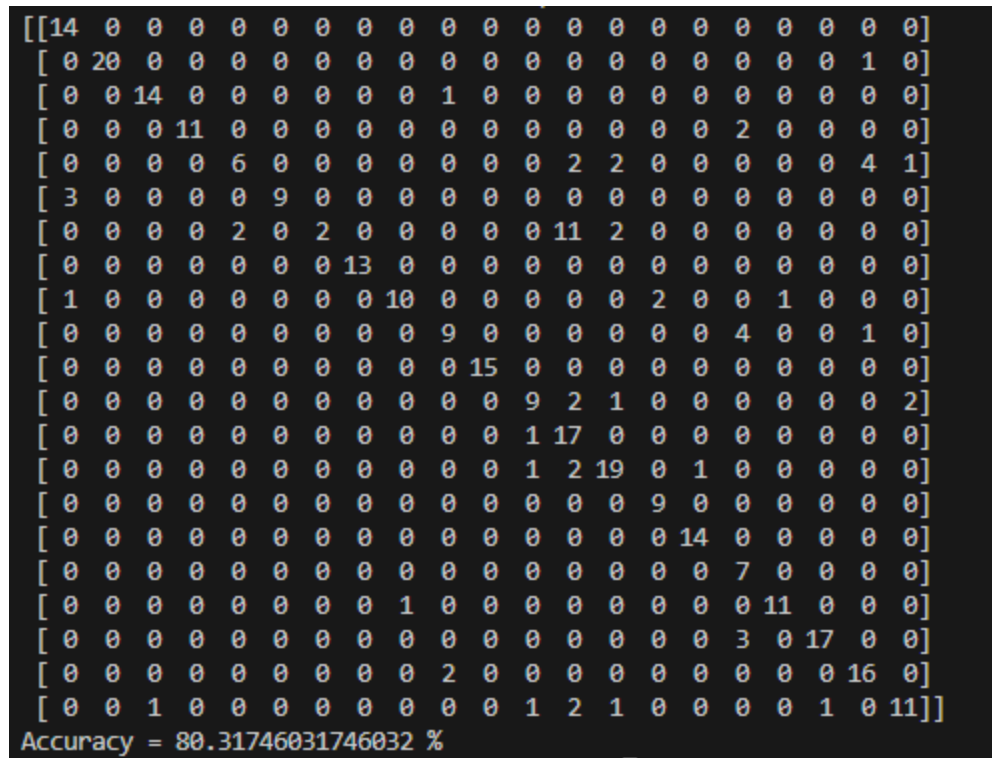


Figure 31. Confusion Matrix

Final Model Results

The model chosen to be my final was the Transfer Learning approach with the heavy data augmentation. To fully test this model I tried to predict all 2100 images in the dataset to see the performance in the entire set, not just the 15% tested and had an accuracy of 89.52% .Training and testing results follow table 5 and figures 22-26.

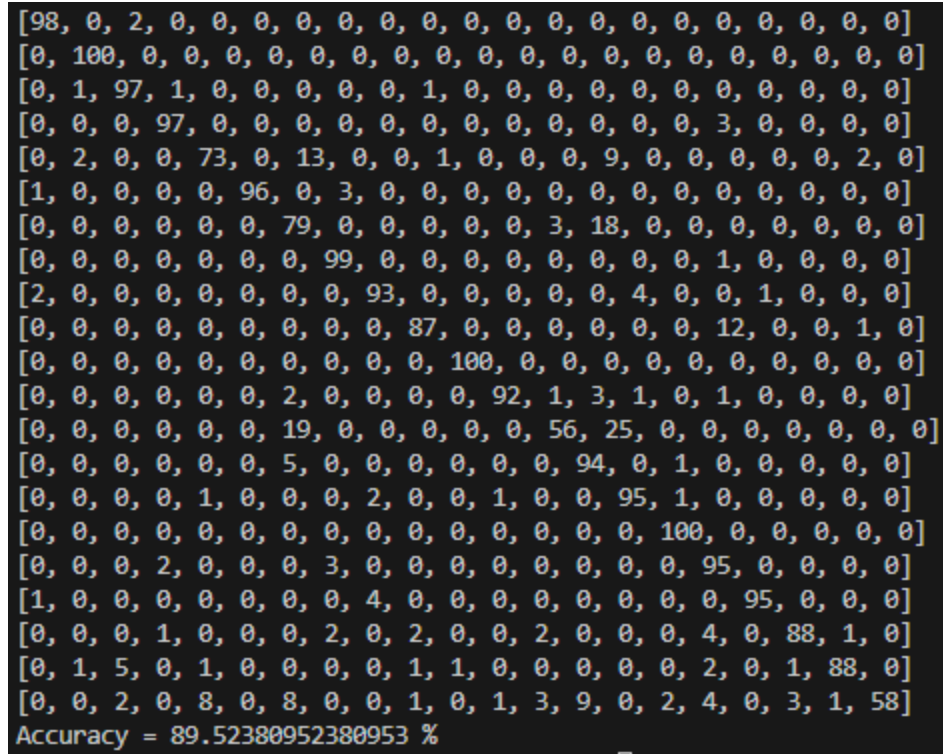


Figure 32. Final Confusion Matrix

Discussion

Analysis and Explanation

Initial Development

During the development of the image classifier, I ran into several problems and used different techniques to counteract. One of the primary limitations which caused issues was the size of the dataset. The dataset contained 2100 images with 100 images per class. I believe that this size of 100 images per class sits on the lower end and using 70% to train is not sufficient enough to give good performance on my models. The first problem that appeared from such an issue was overfitting. With the limited data, the model tended to train on the training set too well, and although provided good training accuracy, caused the validation accuracy to plateau as well as have low testing accuracy approximately around 50%-60%.

To mitigate overfitting, I implemented data augmentation. By applying different transformations such as rotation, shifting, zooming, and flipping. I was able to synthetically increase the size of the training set, and this variability allowed the model to not only avoid overfitting but improve testing accuracy on unseen data to approximately 70%.

Performance Increase

Initial approaches with my own convolutional layers resulted in subpar results. The data sample size and time/computing constraint were insufficient in training a model with effective classification. To find better performance I took the approach of using transfer learning, to leverage a pre-trained model's weights to improve results in my own image. I chose to use VGG16 with trained weights on the ImageNet Dataset [6] to provide a starting point. By using VGG16's convolution base with my own custom top layer, to classify images into 21 classes vs the original 1000, I found significant improvements not only in accuracy, but training time, with a testing accuracy of approximately 80%+ and training time taking around less than the previous methods.

To further improve the performance, I applied more aggressive data augmentation to the transfer learning approach and found that enhancing the training data's diversity gave minor improvements, with no negative impact on training time. The combination of increased data augmentation and transfer learning leveraging VGG16 proved to be effective, showing good performance on the image classification task, with an accuracy on predicting all 2100 images of 89.52%

Effectiveness

The effectiveness of my network appears to be successful in classifying the 2100 images with an accuracy reaching almost 90%. I believe that each iteration of my methods performed reasonably because of the limitations and enhancements discussed previously, of low sample size and techniques used.

I predict that if my network were tested on images outside the dataset, I believe it would generalize well due to the techniques of leveraging transfer learning and VGG16's pretrained weights on ImageNet as a starting point, as well as training on data that was augmented to synthetically increase the sample size. I believe results would be similar or have a minor decrease in performance.

Conclusion

The goal of the project was to develop a convolutional neural network (CNN) capable of classifying aerial images into 21 distinct land use categories using the UC Merced Land Use Dataset [1]. To approach the task, I tried many different methods and tested the results to see which provided the best performance and ultimately chose a transfer learning model. Initial challenges such as overfitting due to the limited dataset size were overcome by implementing data augmentation techniques to enhance data diversity and leveraged transfer learning with the

pre-trained VGG16 model to improve performance. The final model achieved nearly 90% accuracy on the entire dataset, demonstrating an effective classifier.

References

- [1] “UC Merced Land Use Dataset,” *Kaggle*, Jan. 07, 2024.
<https://www.kaggle.com/datasets/abdulhasibuddin/uc-merced-land-use-dataset/data>
- [2] “Training a neural network on MNIST with Keras,” *TensorFlow*.
https://www.tensorflow.org/datasets/keras_example
- [3] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesús, “Neural Network Design (2nd Edition)”. Pearson, 2002.
- [4] “Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names,” May 23, 2018.
https://gomburu.github.io/2018/05/23/cross_entropy_loss/
- [5] 365 Team, “What is Cross-Entropy Loss Function?,” *365 Data Science*, Jun. 15, 2023.
<https://365datascience.com/tutorials/machine-learning-tutorials/cross-entropy-loss/#2>
- [6] Great Learning, “Everything you need to know about VGG16 - Great Learning - Medium,” *Medium*, Jan. 05, 2022. [Online]. Available:
<https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918>