Technical Exercise Report: Creating a Two-Class Image Classifier for Fields and Roads

Introduction

Computer Vision plays a crucial role in solving various real-world problems, and as part of the Trimble/Bilberry technical exercise, I was tasked with creating a two-class image classifier to distinguish between "Field" and "Road" images. This report outlines the methodology, choices made, and results obtained during the process.

1. Methodology

1.1. Data Preparation

The provided dataset consisted of images categorized as "fields" and "roads." These images were collected and organized for analysis. The dataset was split into training and validation subsets using a standard 80-20 split to ensure a robust evaluation of the model.

1.2. Model Architecture:

For this classification task, I opted to use a transfer learning approach. I selected the VGG19 model, a well-established architecture known for its ability to extract intricate features from images. The base layers of VGG19 were pre-trained on ImageNet, and I added custom layers on top for classification. This architecture was chosen to leverage the deep features already learned by VGG19.

I used the TensorFlow framework to implement the model. The model was compiled with the "adam" optimizer and "categorical_crossentropy" loss function, given the two-class classification problem. A dropout layer was introduced to mitigate overfitting. The model was trained for 10 epochs with a batch size of 32. A validation set was used to monitor model performance during training and prevent overfitting.To train the model quickly, I used google colab GPU.

2. Model Choices and Evaluation

2.1. Model Choices

The decision to use the VGG19 architecture was based on its proven track record in image classification tasks. Its deep layers and learned features are well-suited for detecting complex patterns in images. The inclusion of dropout layers aimed to enhance the model's generalization ability.

2.2. Hyperparameter Tuning

The chosen hyperparameters were selected through experimentation and validation performance. Hyperparameters like batch size, optimizer, and dropout rate were fine-tuned to strike a balance between training speed and model performance.

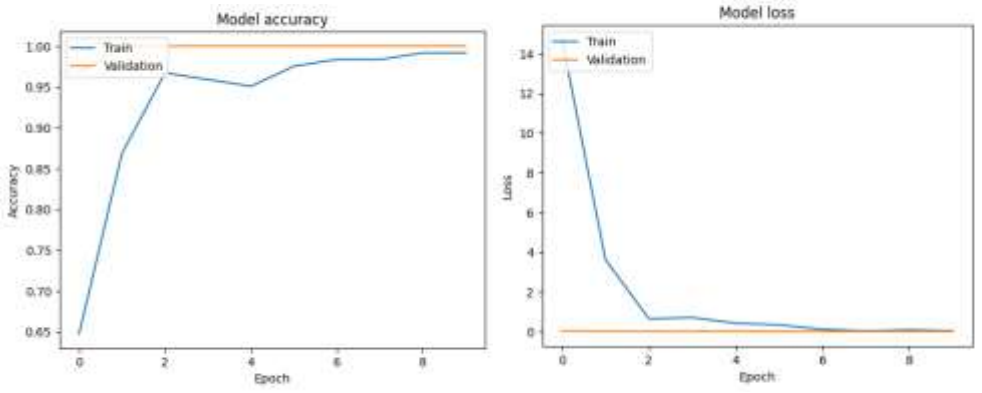
2.3. Model Evaluation

The model's performance was evaluated not only based on its accuracy but also on its ability to generalize to new, unseen data. While the provided test dataset was used for initial evaluation, it's important to note that the model's true performance will be assessed on a different dataset, as indicated in the exercise.

3. Results

3.1. Model Performance

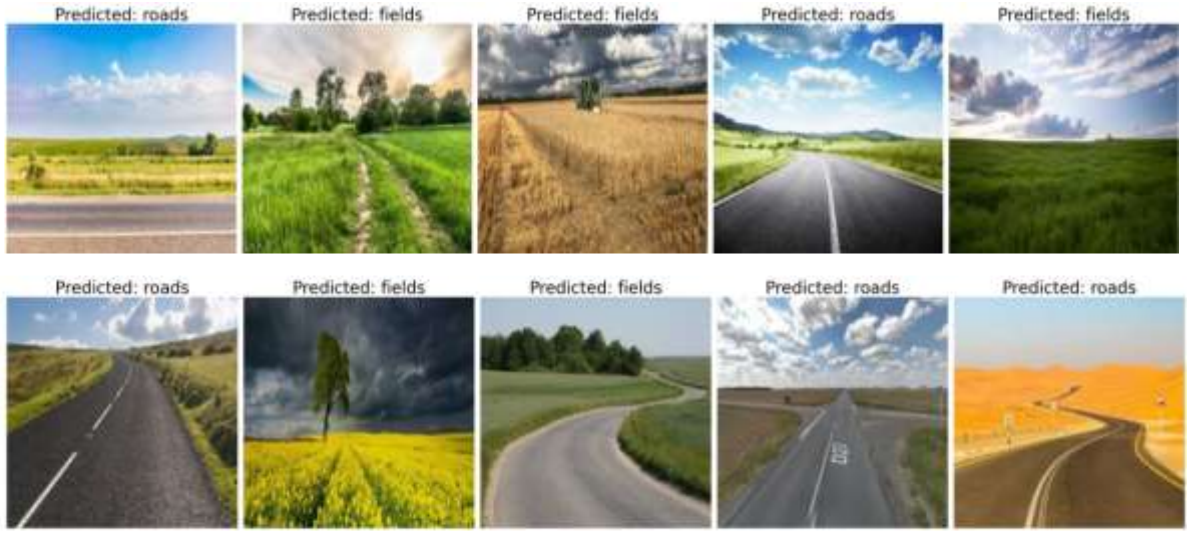
The trained model exhibited promising performance on the provided test dataset. The accuracy on the test set reached 99%.



These 2 curves show that the model learns efficiently and generalizes well to unknown data. The loss curve reaches a plateau (convergence), indicating that the model has reached its best possible fit to the training data. This demonstrates the model's capability to accurately classify new images as "Field" or "Road."

3.2. Visualization

To visualize the model's predictions, I used a subset of the test images and displayed them alongside their predicted labels. This visualization provided insights into the model's classification behavior.



The prediction results on the test data provided are quite efficient.

Conclusion

In conclusion, the successful creation of a two-class image classifier for "Field" and "Road" images demonstrates the potential of computer vision techniques in solving real-world problems. The chosen approach of transfer learning using the VGG19 architecture, along with careful hyperparameter tuning, yielded a well-performing model.

NB: the model has been saved in the trained_model.h5 file for easy use. To load this, you can run the following code:

“from tensorflow.keras.models import load_model

model = load_model('trained_model.h5')”