Chess Piece Classification Report

1.0 Data Preparation

Unzipping the Training Data

I started by unzipping the dataset and listing the classes. The dataset consists of images categorized into the following classes:

- Knight

- Bishop

- King

- Queen

- Rook

- Pawn

These represent the chess pieces we want the model to recognize.

1.1 Data Augmentation

Data augmentation involves applying transformations to the training images to generate additional samples. This helps to diversify the dataset and improve the model's ability to generalize (i.e., perform well on new, unseen data).

For this task, I used an 'ImageAugmentation' class to apply various transformations to the dataset.

1.2 Dataset Splitting

Dataset Class

The 'ImageDataset' class is responsible for loading and transforming images during training.

Dataset Splitting

The dataset was divided into three parts:

- Training set: 94% of the data

- Validation set: 6% of the data

- Test set: 30 samples (used for final model evaluation)

This ensures that the model has enough data to learn from, while also having data it hasn’t seen during training to test its performance.

2.0 Model Selection and Enhancement

Baseline Model

The ‘ChessClassifier’ class defines a Convolutional Neural Network (CNN) architecture with convolutional and fully connected layers. The CNN architecture is a standard choice for image classification tasks.

Training the Baseline Model

The baseline model was trained for 25 epochs. However, due to the complexity of the dataset and the simplicity of the architecture, the model didn't improve significantly. The final performance was only **30.8% accuracy**. This indicates that the simple model struggled to capture the intricacies of the chess pieces' images.

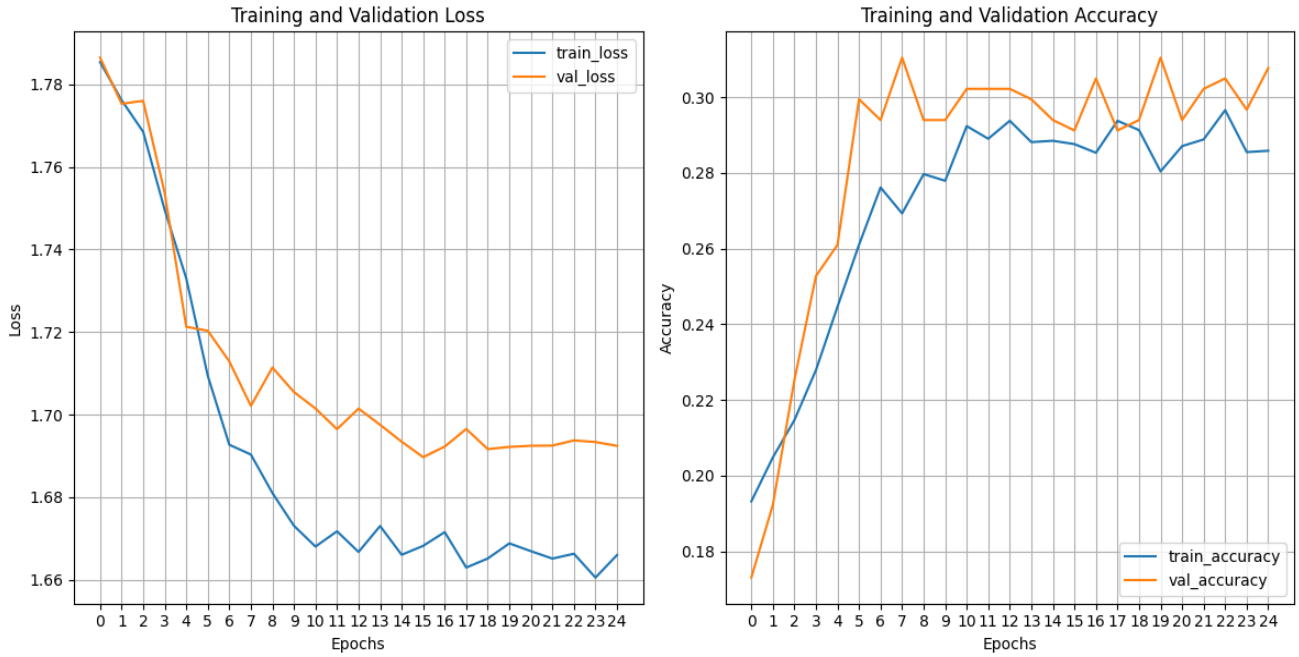

Fig 2.1: Training Metrics

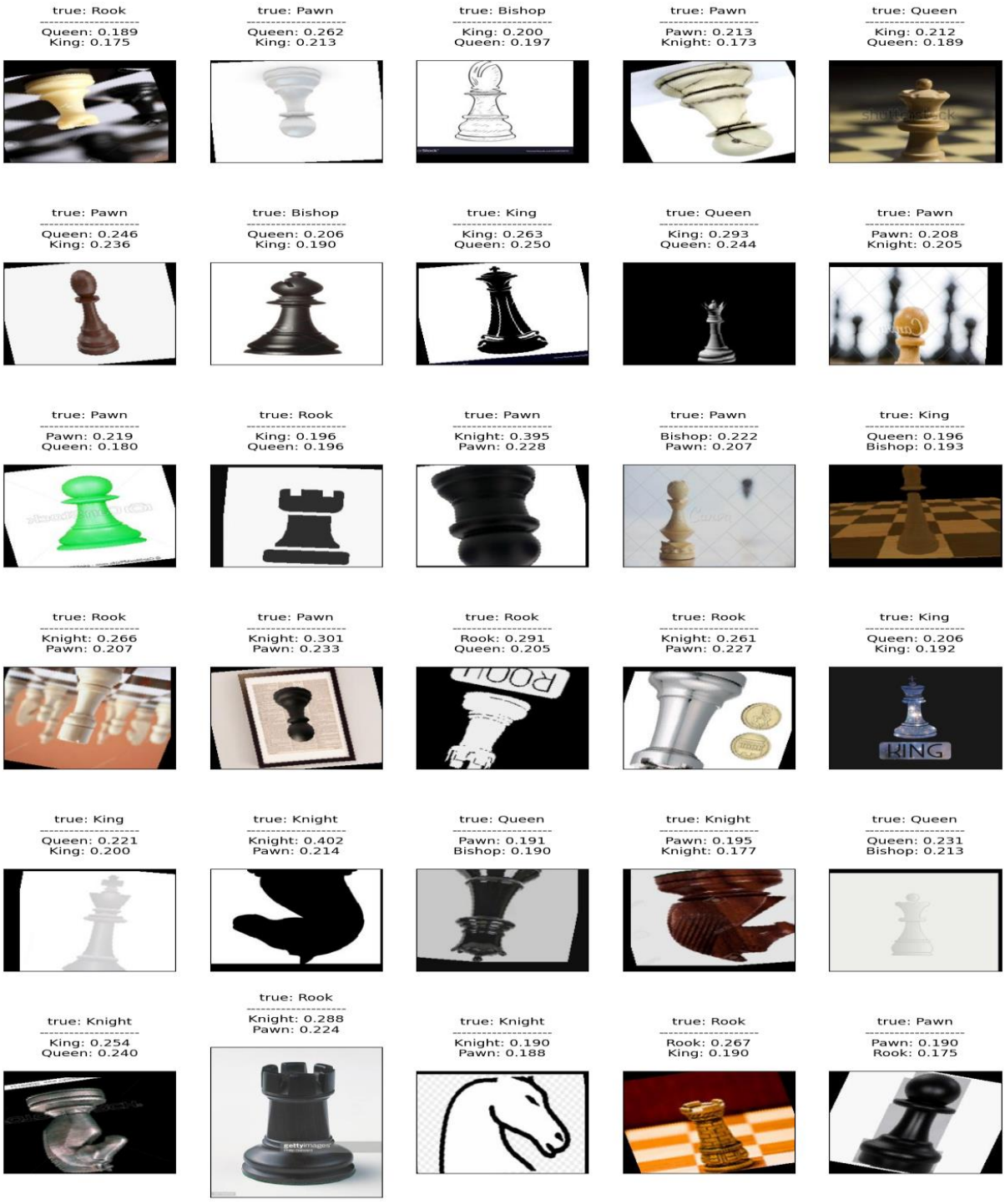

Fig 2.2: Baseline Model Predictions

3.0 Fine-Tuning Pre-Trained Model - VGG16

Model Setup

The `setup_vgg16_for_finetuning` function sets up the pre-trained VGG16 model to be fine-tuned for the chess piece classification task. Fine-tuning is an excellent approach when the dataset is small, as it allows the model to leverage knowledge learned from a much larger dataset (like ImageNet) to improve its performance on our task.

Training and Evaluation

The VGG16 model was trained for 20 epochs, and both training and validation losses and accuracies were recorded and plotted. This helped in visualizing the model's performance over time.

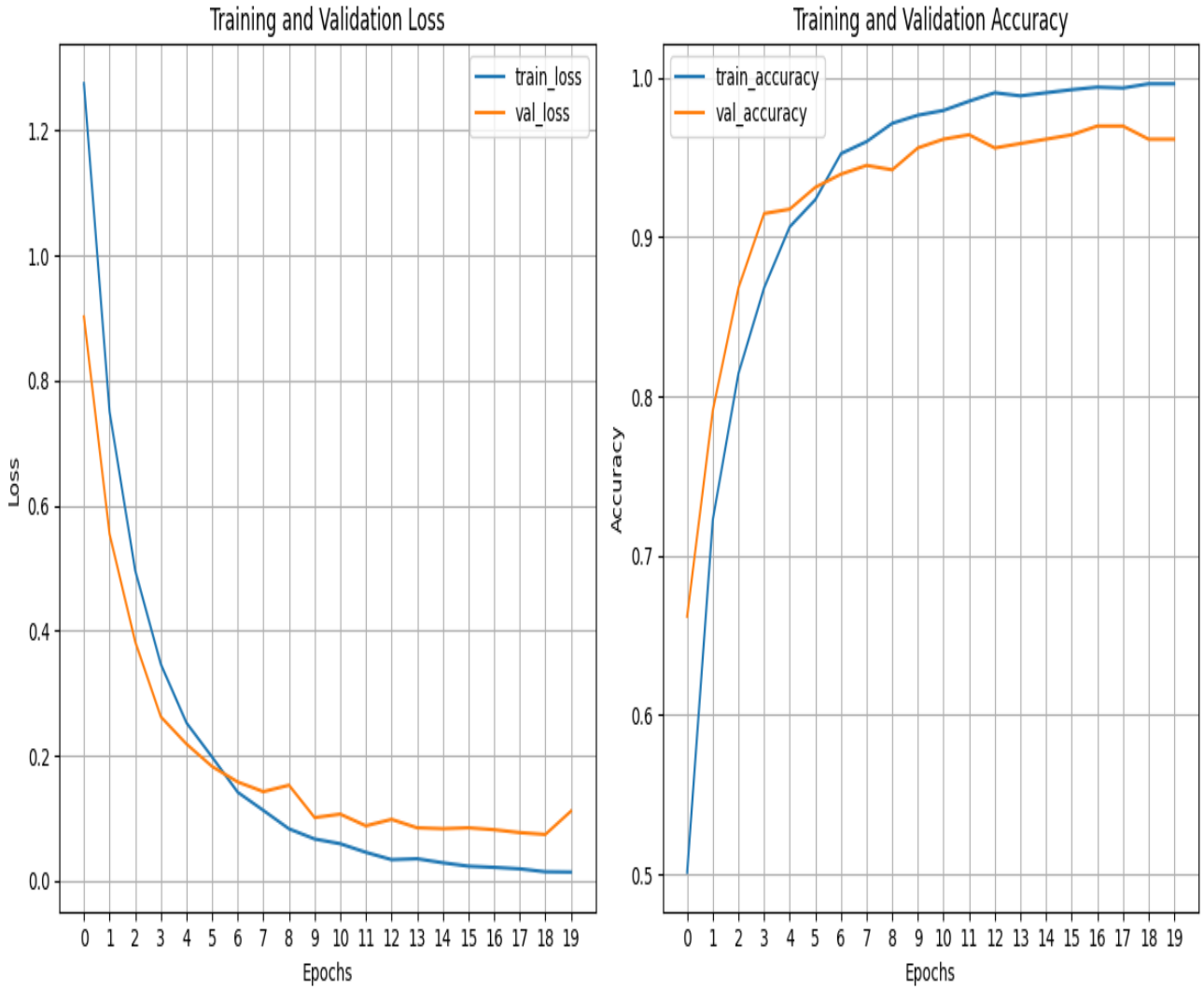

Fig 3.1: Training Metrics

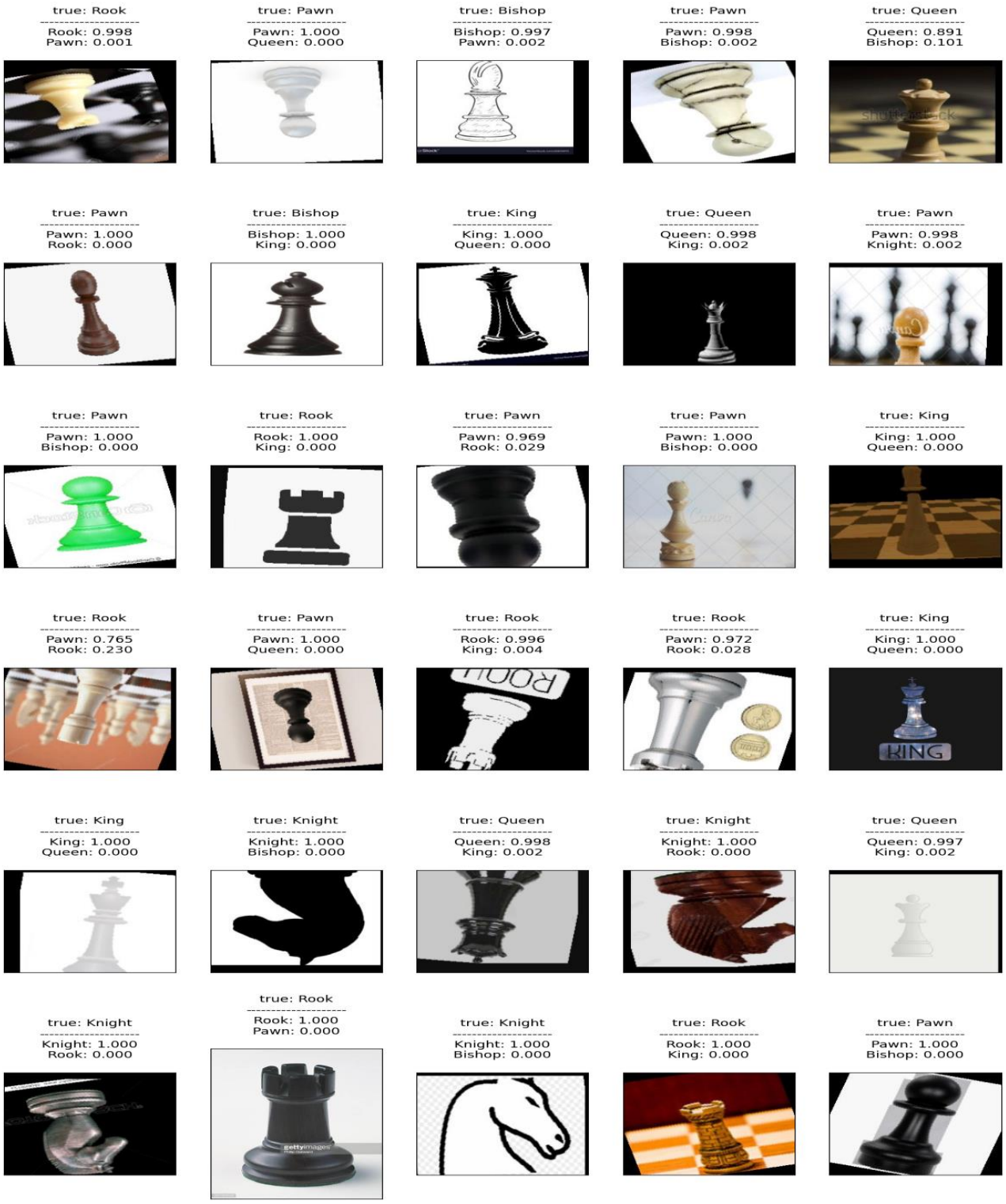

Fig 3.2: Baseline Model Predictions

Results

The VGG16 model significantly outperformed the baseline model, achieving a final validation accuracy of **96.2%**. This indicates that the fine-tuned model can generalize well to unseen data and is much more effective in recognizing chess pieces.

4.0 Analysis and Improvements

Data Augmentation

While the data augmentation process was applied, additional transformations (e.g., scaling) could further diversify the dataset and improve the model's robustness. It's important to ensure that these augmented images still resemble realistic chess pieces.

Model Architecture

Future improvements could involve experimenting with deeper or more complex network architectures like ResNet. Transfer learning (using pre-trained models) also proved successful with VGG16, and similar approaches could be tested with other models.

Training Enhancements

To avoid overfitting (where the model performs well on training data but poorly on new data), it's recommended to implement early stopping. This technique stops training once the model's performance on the validation set stops improving.

Evaluation Metrics

In addition to accuracy, it's important to use other evaluation metrics like precision, recall, and F1-score. These metrics provide a more detailed understanding of how well the model is performing, especially if there's an imbalance in the classes. Visualizing the confusion matrix would help in identifying which classes the model struggles with.

Hyperparameter Tuning

Experimenting with different learning rates, batch sizes, and other hyperparameters could optimize the model's performance further.

5.0 Conclusion

The notebook is well-structured and successfully demonstrates the steps required to build a chess piece classifier. The VGG16 model performed the best, achieving a final validation accuracy of **96.2%**. By implementing the suggested improvements and fine-tuning further, the model's performance can be enhanced even more, ensuring it provides reliable and accurate predictions.