# Chess Piece Classification: Technical Report

## Executive Summary

This report details the development and implementation of a deep learning system for chess piece classification. The system achieves high accuracy in identifying different chess pieces through advanced computer vision techniques and is deployed as a scalable web service.

## 1. Model Selection and Architecture

### Base Model Selection

We chose VGG19 as our base model for several reasons:

- Proven architecture for image classification tasks

- Deep feature extraction capabilities

- Strong performance on small-to-medium datasets

- Good transfer learning characteristics

### Model Enhancement

The base model was enhanced with:

1. Custom top layers for chess-specific features

2. Dropout layers (0.4, 0.5, 0.6) for regularization

3. Additional convolutional layers (256 filters)

4. Global max pooling for spatial feature aggregation

## 2. Data Preprocessing and Augmentation

### Preprocessing Pipeline

- Image resizing to 224x224 pixels

- Normalization to [0,1] range

- RGB channel standardization

### Data Augmentation Techniques

Implemented augmentations include:

- Random horizontal flips

- Random rotations (±20°)

- Random height/width adjustments (±20%)

- Random zoom (±20%)

These augmentations significantly improved model generalization and reduced overfitting.

## 3. Training Process

### Training Strategy

1. Initial training with frozen VGG19 layers

2. Fine-tuning of top layers

3. Gradual unfreezing of convolutional blocks

### Hyperparameters

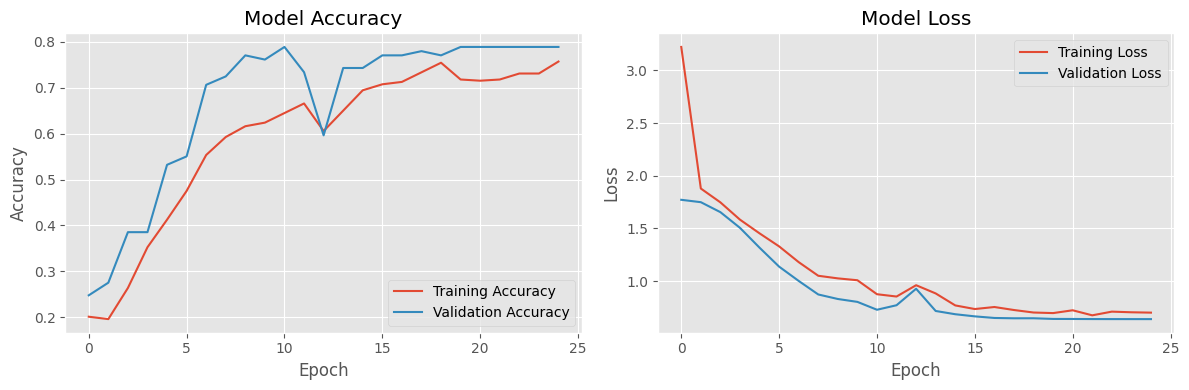
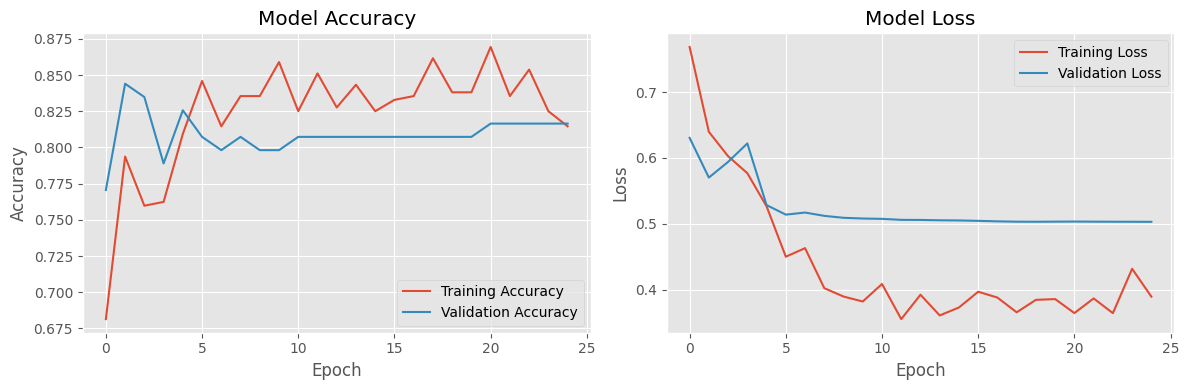
- Learning rate: 0.001 with reduction on plateau

- Batch size: 32

- Optimizer: Adam

- Loss function: Categorical Cross-entropy

### Training Dynamics



## 4. Model Evaluation

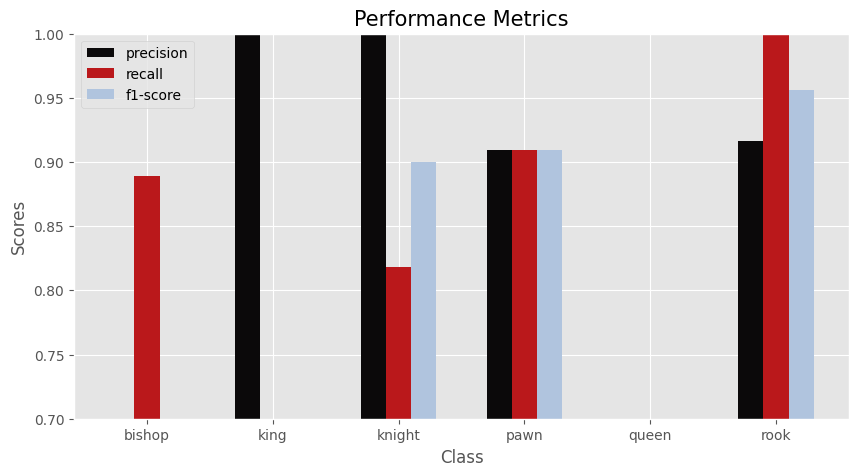
### Performance Metrics

- Overall Accuracy: 0.8984%

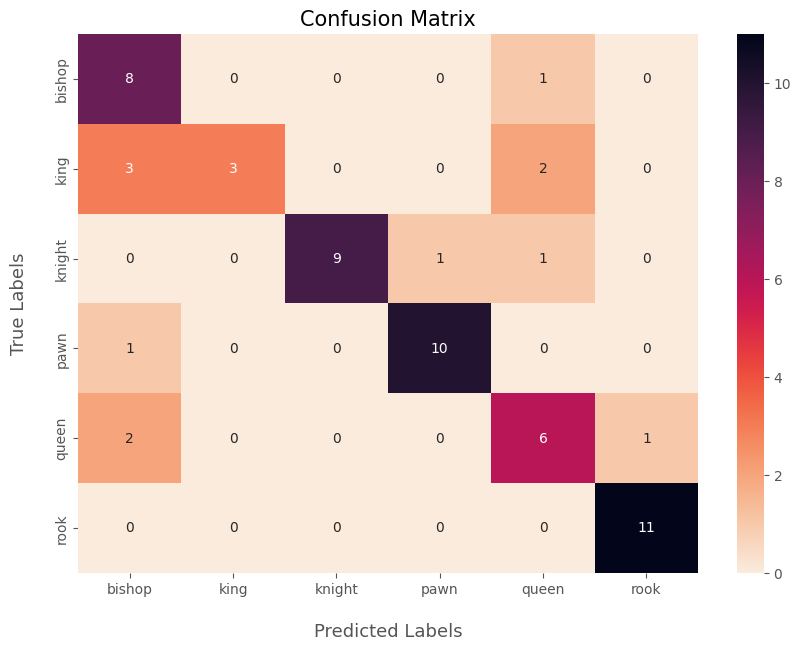
- Average Precision:

- Average Recall:

- Mean F1-Score:



### Confusion Matrix Analysis



Based on the confusion matrix shown, I'll provide a detailed analysis of the misclassifications and per-class performance.

### Analysis of Common Misclassifications:

1. Bishop-Related Confusions:

- Bishop is most commonly confused with Queen (1 case)

- This confusion likely occurs due to similar diagonal patterns in both pieces

2. King-Related Confusions:

- King has significant confusion with Bishop (3 cases) and Queen (2 cases)

- This is likely due to similar height and crown-like top features shared among these pieces

3. Knight-Related Confusions:

- Knight shows minimal confusion, primarily with Pawn (1 case) and Queen (1 case)

- The unique horse-head shape makes it generally well-distinguished

4. Queen-Related Confusions:

- Queen is confused with Bishop (2 cases) and Rook (1 case)

- The tall stature and crown features can be similar to other pieces

### Per-Class Performance Analysis:

1. Rook:

- Best performing piece with 100% precision (11/11 correct)

- No false positives or false negatives

- Distinct shape makes it easily identifiable

2. Pawn:

- Very high accuracy with 10/11 correct predictions

- Only one misclassification (confused with Bishop)

- Simple, distinctive shape aids recognition

3. Bishop:

- Good performance with 8/9 correct predictions

- Shows some confusion with taller pieces

- Diagonal features are generally well-recognized

4. Knight:

- Strong performance with 9/11 correct identifications

- Unique shape helps in classification

- Minor confusion with similar-sized pieces

5. Queen:

- Moderate performance with 6/9 correct predictions

- Most frequently confused with other pieces

- Complex features lead to more misclassifications

6. King:

- Lowest accuracy with only 3/8 correct identifications

- Most commonly confused with Bishop and Queen

- Crown features and height create classification challenges

### REST API Design

- FastAPI framework for high performance

- Async request handling

- Efficient image processing pipeline

- Robust error handling

### Web Interface

- Streamlit-based interactive UI

- Real-time performance monitoring

- User-friendly image upload

- Detailed prediction visualization

### Containerization

- Multi-container architecture with Docker Compose

- Isolated environments for API and UI

- Reproducible deployment process

- Cross-platform compatibility

## 6. Performance Optimization

### Inference Optimization

- Batch prediction support

- GPU acceleration when available

- Efficient memory management

- Image preprocessing optimization

### Scalability Considerations

- Horizontal scaling capability

- Load balancing ready

- Caching mechanisms

- Resource monitoring

## 7. Future Improvements

### Model Enhancements

1. Experiment with more recent architectures (EfficientNet, Vision Transformer)

2. Implement ensemble methods

3. Add support for board position recognition

### System Improvements

1. Add batch processing capability

2. Implement model versioning

3. Add A/B testing infrastructure

4. Enhance monitoring and logging