Comprehensive Evaluation Report: Conjunctivitis Eye Disease Classification (Phase 3)

Jazmine Brown, Javon Darby, Katherine Stanton, Jeffery Dirden

In this phase of our capstone project, we worked collaboratively to design, train, and evaluate a deep learning model capable of identifying conjunctivitis in eye images. Our objective was to create a reliable image classification system that could distinguish between *healthy eyes* and *infected eyes* using computer vision and convolutional neural networks (CNNs). We implemented and tested our model using Google Colab with TensorFlow and tracked the entire experimental workflow through MLflow.

1. Dataset and Preprocessing

We obtained the Conjunctivitis Dataset from Kaggle via KaggleHub. The dataset contained 358 labeled images evenly split between healthy and infected eyes, which made it well-suited for binary classification. After downloading, we confirmed the directory structure and explored the dataset through exploratory data analysis (EDA). This step involved visualizing random samples and analyzing class distributions to ensure that both categories were balanced.

To prepare the data for training, we split it into training (80%), validation (10%), and testing (10%) subsets using stratified sampling to preserve class balance. We then used TensorFlow's ImageDataGenerator to preprocess and augment the data. The training generator applied transformations such as rotation, zoom, width/height shifts, and horizontal flips, which improved generalization by exposing the model to variations in image positioning and lighting. The validation and test sets were only rescaled to maintain data integrity.

2. Model Architecture

We designed a lightweight CNN architecture optimized for speed and accuracy on modest GPU hardware. The model included:

- Three convolutional layers with increasing filter depth (32, 64, 128) for hierarchical feature extraction.
- Batch normalization layers after each convolution to stabilize learning.
- MaxPooling layers to reduce spatial dimensions and computational load.
- A GlobalAveragePooling2D layer to replace fully connected layers, reducing overfitting and model complexity.
- A dense layer with ReLU activation followed by a dropout layer (rate 0.35) for regularization.
- A final output layer using a sigmoid or softmax activation depending on configuration (binary vs. categorical).

This architecture achieved a balance between model expressiveness and computational efficiency, which was important for maintaining fast training cycles in Colab.

3. Model Training and Optimization

We trained the model using the Adam optimizer with a learning rate of 0.001 and binary cross-entropy loss for two-class classification. Our early tests with fewer than 30 epochs resulted in underfitting, where the model failed to capture complex patterns in the data. After increasing the training duration to 30 epochs, we observed a steady improvement in both training and validation accuracy. The final model achieved approximately 93% test accuracy, showing a strong ability to distinguish between healthy and infected eyes.

To further enhance performance, we integrated MLflow for automated experiment tracking. This tool allowed us to log hyperparameters, training metrics, and model artifacts for each run. We then used Keras Tuner to perform hyperparameter optimization on the learning rate, batch size, and dropout rate. The tuner automatically searched for the best configuration that maximized validation accuracy, which streamlined our experimentation process and provided deeper insight into how small parameter changes affect model outcomes.

4. Model Evaluation

Once training was complete, we evaluated the model on the test set using several performance metrics:

- Test Accuracy: ~93.0%
- ROC-AUC Score: ~0.987
- PR-AUC (Average Precision): ~0.987
- Cohen’s Kappa: ~0.86
  These high scores indicate that the model performed well at distinguishing between classes, with minimal false positives and false negatives.

We also visualized the confusion matrix, which confirmed that most predictions were accurate and evenly distributed between the two classes. The ROC and Precision-Recall curves showed high area under the curve values, further validating the model’s strong discriminative capability.

5. Error and Misclassification Analysis

To better understand the model’s weaknesses, we conducted an error analysis on misclassified samples. By visualizing these images, we noticed that most misclassifications occurred on images with poor lighting, glare, or low contrast. To quantify this, we analyzed brightness and contrast levels across correctly classified and misclassified samples. Our results showed that darker and blurrier images had a higher tendency to be predicted incorrectly. This finding emphasized the importance of consistent image quality and suggested that additional preprocessing (e.g., histogram equalization or denoising) could help improve robustness.

6. Experiment Tracking and Comparison

Using MLflow, we tracked multiple experiments and generated a model comparison table summarizing key parameters such as learning rate, batch size, validation accuracy, test accuracy, and AUC scores. This systematic tracking helped us identify that the configuration with a batch size of 32, learning rate of 0.001, and dropout of 0.3 achieved the best performance. Having this experiment log also made it easier to justify design decisions and document the evolution of our model.

7. Reflections and Future Improvements

Working as a team on this project gave us valuable experience in collaborative model development, data handling, and AI experiment management. We learned how CNNs can effectively capture visual features from medical images and how tuning hyperparameters can drastically change model behavior.

For future iterations, we plan to:

- Expand the dataset to include more diverse eye conditions.
- Incorporate transfer learning using pre-trained models like MobileNetV2 to improve generalization.
- Implement better preprocessing to address lighting inconsistencies.
- Deploy the model into a simple web-based interface for real-time prediction.

8. Conclusion

Overall, our project successfully met the objective of creating a functional prototype capable of detecting conjunctivitis with high accuracy. The combination of CNN architecture, data augmentation, and systematic experiment tracking provided a clear path from data collection to performance evaluation. This experience not only improved our understanding of deep learning in healthcare applications but also taught us how reproducibility, documentation, and collaboration play crucial roles in real-world AI development.