**Methodology**

This section outlines the methodology for developing and evaluating a convolutional neural network (CNN)-based model for glaucoma detection using OCT images. The workflow involves multiple stages, including data acquisition, preprocessing, model design, training, and evaluation, ensuring high diagnostic accuracy.

The dataset used for this study was sourced from Kaggle's **Glaucoma OCT Images** dataset. It contains labeled images representing normal and advanced glaucoma conditions. The dataset was downloaded and extracted programmatically to enable seamless integration with the preprocessing pipeline.

**1. Data Preprocessing**

To enhance the quality of the images and prepare them for model training, the following preprocessing steps were applied:

**a. Image Loading and Resizing**

* Images were loaded using OpenCV and resized to a uniform dimension of 224×224224 \times 224 pixels to ensure compatibility with the CNN model.

**b. Image Enhancement**

* Noise reduction was applied using Gaussian blur.
* Edge detection was performed using the Canny method.
* Sharpening was applied using unsharp masking techniques to enhance the key features of glaucoma-related patterns.

**c. Image Normalization**

* Pixel intensity values were normalized to a range of [0, 1], improving convergence during training.

**d. Label Encoding**

* Labels were encoded into binary values: 0 for normal glaucoma and 1 for advanced glaucoma, enabling binary classification.

**2. Model Development**

The model was implemented using TensorFlow and Keras, focusing on extracting hierarchical features through convolutional operations.

**a. Architecture Design**

A custom CNN architecture was developed with:

* **Convolutional Layers**: Extracting hierarchical features using 3×33 \times 33×3 kernels and ReLU activation.
* **Pooling Layers**: Reducing spatial dimensions using 2×22 \times 22×2 and 3×33 \times 33×3 max pooling.
* **Dropout Layers**: Mitigating overfitting with dropout rates of 20%, 25%, and 50%.
* **Fully Connected Layers**: Integrating features with a dense layer of 128 neurons and L2L\_2L2​-regularization, followed by a softmax output layer for classification.

**b. Hyperparameter Selection**

Key hyperparameters such as learning rate, batch size (60), and number of epochs (60) were empirically tuned. The Adam optimizer and sparse categorical cross-entropy loss were used for optimal performance.

**c. Early Stopping**

Early stopping was applied to terminate training when validation loss failed to improve for three consecutive epochs, ensuring model generalization.

**3. Training and Validation**

* The dataset was split into training (70%) and validation (30%) subsets using stratified sampling to ensure balanced class representation.
* The model was trained using the Adam optimizer and a sparse categorical cross-entropy loss function. The validation subset was used for monitoring performance during training.

**4. Evaluation and**

The trained model was evaluated using the following methods:

**a. Performance**

* Validation accuracy and loss were computed to assess the model's ability to generalize to unseen data.
* A confusion matrix was generated to calculate precision, recall, F1-score, and AUC-ROC for a detailed evaluation.

**A black background with white and orange squares

Description automatically generated**