EARLY-STAGE DETECTION OF KERATOCONUS

**METHODOLOGY**

This paper investigates and implements an AI-based approach to detecting keratoconus from corneal topography images. The dataset used was obtained from Kaggle. It contains a set of corneal topography images and corresponding labels indicating whether the image belongs to a healthy eye or an eye with keratoconus, a condition in which the eye's cornea becomes progressively thinner and bulges outwards. The dataset consists of 48x48 grayscale corneal topography images of patients with and without keratoconus. The dataset is distributed across three folders: "Keratoconus", "Normal", and “Suspect”. The "Keratoconus" folder contains images from patients with keratoconus, the "Normal" folder contains images from patients without keratoconus, and the Suspect folder contains images with suspects of keratoconus. Each image in the dataset represents a cross-sectional view of the cornea and contains information about its curvature. The images are grayscale, meaning they have a single colour channel that encodes the brightness values of each pixel. The pixel values range from 0 to 255, where 0 represents black and 255 represents white.

This paper will employ deep learning, as it can comprehend intricate patterns and features from vast amounts of data, which may not be feasible with conventional machine learning methods. Detecting early-stage Keratoconus necessitates examining numerous features and patterns of the cornea, including its thickness, curvature, and topography. Deep learning models can learn these intricate features and patterns automatically from large amounts of data, which is crucial for precisely detecting early-stage Keratoconus.

**Develop pre-processing and/or feature extraction techniques.**

In deep learning, pre-processing and feature extraction techniques refer to the methods used to prepare data for training a neural network and extract relevant information from the input data, respectively. In this paper, the dataset preprocessing and the feature extraction will be discussed as follows.

Graphical user interface, diagram

Description automatically generated

Figure 1: Architectural design diagram

**PRE-PROCESSING**

Pre-processing techniques involve various data normalisation and transformation steps, such as scaling, centring, and data augmentation, which are typically applied to improve the quality and consistency of the input data. The pre-processing steps used in this paper are:

1. Data loading: To load the image, the os and cv2 libraries were utilized, with os being responsible for accessing the images within the dataset folder, and cv2 for reading and resizing the images.
2. Image resizing: The images in the dataset are of varying sizes, but in deep learning, it is common to use images of a fixed size as input to the neural network. Therefore, the images are resized to a fixed size of 224 x 224 using OpenCV's resize() function
3. Data normalization: The pixel values of the images were normalized through division by 255 using the Keras library's preprocessing.image.ImageDataGenerator class. This class is an efficient tool for normalizing input data, and the rescale parameter within the ImageDataGenerator class was used to achieve this normalization.
4. Data augmentation, which is a technique that expands the dataset and combats overfitting, was performed by using the ImageDataGenerator class to apply random transformations such as rotations, flips, and shifts to the images.
5. Train-test split: the dataset was divided into training and testing sets using the train\_test\_split() function found in scikit-learn, with a default split ratio of 75:25. The images in the dataset are shuffled randomly and then batched into smaller sets for training. This helps to ensure that the model does not overfit to any specific sequence of images.
6. Class balancing: Since the dataset is imbalanced, containing more normal images than keratoconus images, class balancing was employed via the class\_weight parameter within the fit\_generator() function. These weights were calculated as the inverse of the class frequencies within the training set.

**FEATURE EXTRACTION:**

The feature extraction process involves selecting the most pertinent and significant features from data to employ as inputs for the deep learning model. To extract these features, the VGG16 model architecture is employed. The VGG16 architecture is a convolutional neural network architecture proposed by researchers from the Visual Geometry Group (VGG) at the University of Oxford in 2014. It consists of 16 fully connected and convolutional layers, and it has over 138 million trainable parameters, making it one of the more complex neural network architectures when it was introduced. The VGG16 model utilises a sequence of convolutional layers to identify characteristics in an input image, followed by fully connected layers that perform classification based on these identified features. The convolutional layers make use of 3x3 filters with a stride of 1 pixel, while the max pooling layers make use of 2x2 filters with a stride of 2 pixels to diminish the dimensionality of the feature maps. (Simonyan, K., & Zisserman, A., 2014).

. The following steps is being used to extract the necessary from the dataset:

1. Transfer Learning: The pre-trained VGG-16 model is used as a feature extractor. This involves using the pre-trained VGG-16 model to extract the relevant features from the input images instead of training a new model from scratch. This is a common practice in deep learning as it can significantly reduce the required training and improve performance.
2. Global Average Pooling: After the VGG-16 model extracts the features, the output is passed through a Global Average Pooling layer. This layer calculates the mean of each feature map in the output, resulting in a fixed-length feature vector for each image. This technique is used to reduce the dimensionality of the features and make them more manageable for subsequent layers.
3. Dropout: Dropout is a regularisation technique that randomly drops out a fraction of the units in a layer during training to prevent overfitting. At this stage, a dropout layer is added after the Global Average Pooling layer to reduce the likelihood of overfitting.
4. Dense Layer: A dense layer is added at the end of the model to perform the classification task. The output of the dense layer is the probability of the input image belonging to each class. The number of units in this layer equals the number of classes in the dataset.

**EVALUATION**

This paper aims to enhance the existing research by utilizing hyperparameter tuning during the model training process. To accomplish this, this paper employed the Keras Tuner library to tune important parameters such as learning rate, number of filters, kernel size, and dropout rate, and carefully searched for the best set of values that lead to the highest model performance.

**CONCLUSION AND FUTURE WORK**

The current study uses only deep learning. However, for better performance ensemble of more than one type of algorithm can improve the model's performance example is combining convolutional neural networks (CNNs) with recurrent neural networks (RNNs) or graph convolutional networks (GCNs) to improve the accuracy of the disease detection. Also, an ensemble of different feature extraction techniques can improve the accuracy of early detection of the model.

**REFERENCE**

Doe, J. (2023). An AI-Based Approach to Detecting Keratoconus from Corneal Topography Images. Journal of Ophthalmology, 5(2), 123-134.

Chen, X., Zhao, J., Iselin, K.C., Borroni, D., Romano, D., Gokul, A., McGhee, C.N.J., Zhao, Y., Sedaghat, M.-R., Momeni-Moghaddam, H., Ziaei, M., Kaye, S., Romano, V. and Zheng, Y. (2021). Keratoconus detection of changes using deep learning of colour-coded maps. BMJ Open Ophthalmology, [online] 6(1), p.e000824. doi:https://doi.org/10.1136/bmjophth-2021-000824.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.