Title: Binary Cataract Classification Using Deep Learning : EfficientNetB0 & CNN Model

Introduction

The objective of this project is to develop a deep learning model for automated detection of cataracts from eye images. Cataracts are a common eye condition that can lead to vision impairment if left untreated. Automated screening tools can assist healthcare professionals in early detection and intervention, potentially improving patient outcomes.

Dataset

2.1 Dataset Overview

The dataset consists of eye images classified into two categories: normal and cataract.

Training set:

Total images: 491
Normal images: 246
Cataract images: 245

Test set:

Total images: 121
Normal images: 60
Cataract images: 61

2.2 Image Analysis

We analyzed the dimensions of images in both classes:

Normal image example size: 171 x 242 x 3 (Height x Width x Channels)
Cataract image example size: 335 x 493 x 3 (Height x Width x Channels)

The scatter plots (Image 1 and Image 2) show the distribution of image sizes for normal and cataract images respectively. These plots reveal:

- Significant variation in image sizes within each class
- Cataract images tend to be larger on average compared to normal images
- No clear separation between classes based on image size alone

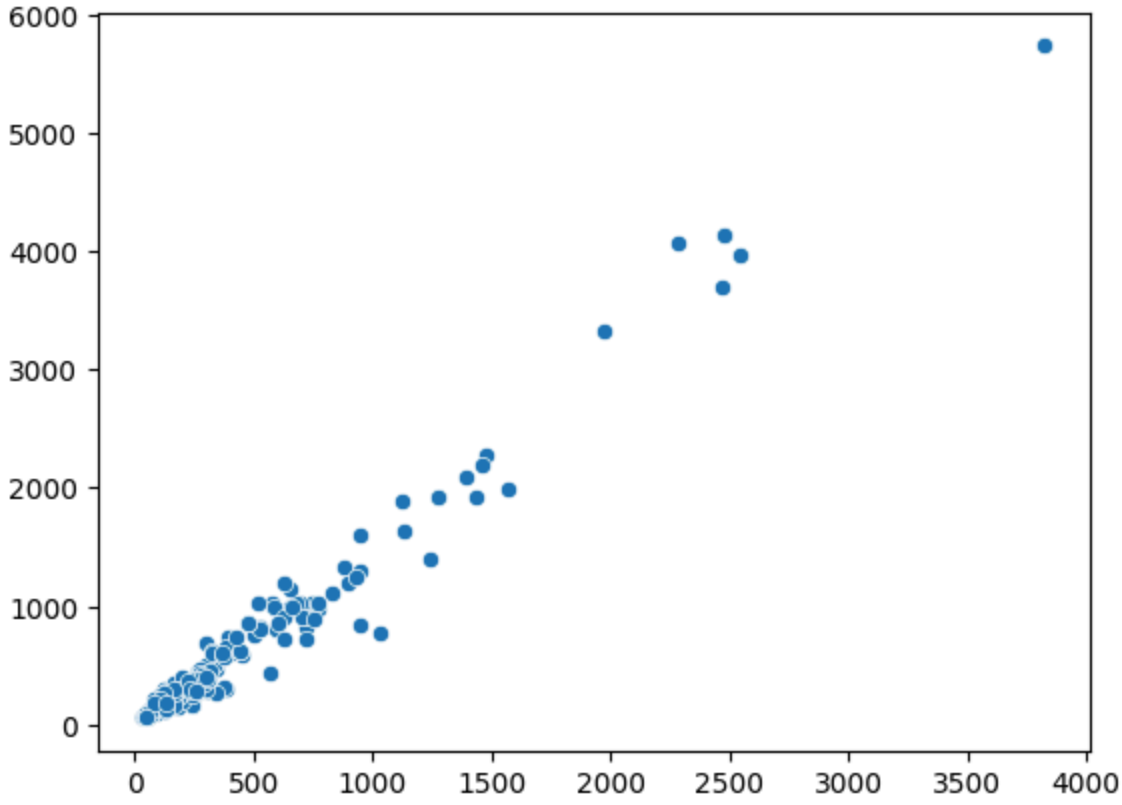

Fig : Normal shape mean

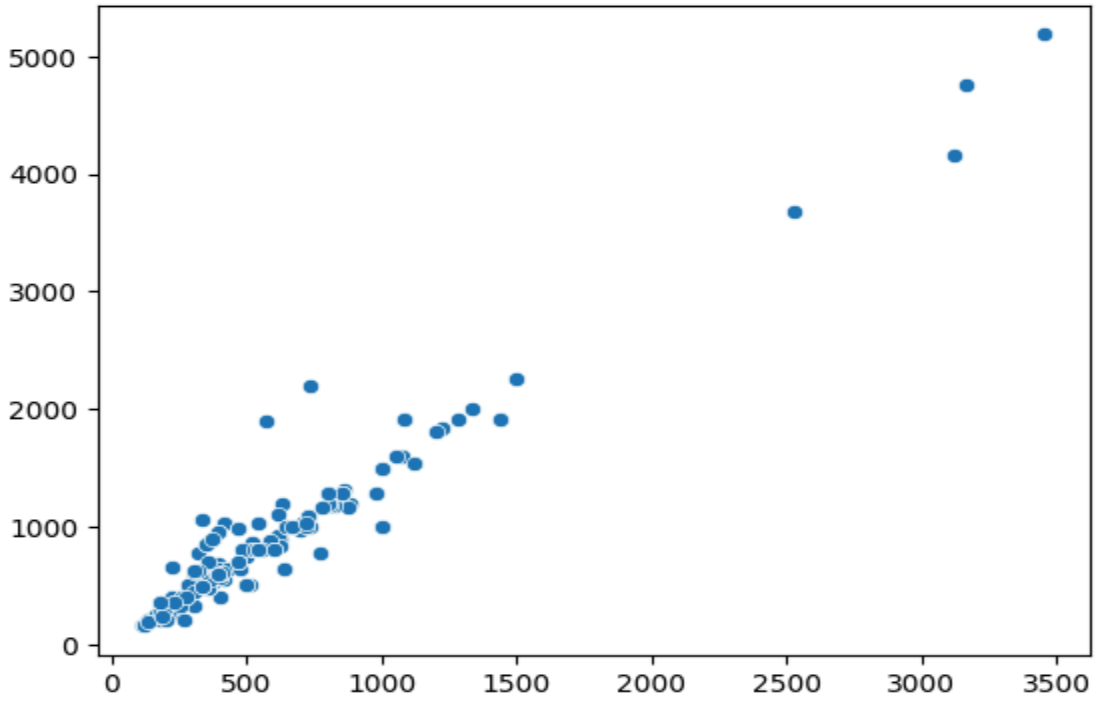

Fig : Cataract shape mean

Data Preprocessing

- Rotation:
  Images are rotated randomly within a range of 20 degrees (rotation_range=20).

- Width Shift:
  Images are shifted horizontally by a fraction of the total width. This shift is random and within a range of 10% of the width (width_shift_range=0.1).

- Height Shift:
  Images are shifted vertically by a fraction of the total height. This shift is random and within a range of 10% of the height (height_shift_range=0.1).

- Shear:
  A shear transformation is applied to the images, with a shear intensity up to 10 degrees (shear_range=0.1).

- Zoom:
  Images are zoomed in or out by a factor within a range of 10% (zoom_range=0.1).

- Horizontal Flip:
  Images are randomly flipped horizontally (horizontal_flip=True).

- Fill Mode:
  Points outside the boundaries of the input are filled according to the nearest mode (fill_mode='nearest').

- Validation Split:
  20% of the data is set aside for validation (validation_split=0.2).

- Class Labels -The class labels are defined as follows:
  0: Cataract
  1: Normal

4. Model Architectures

4.1 CNN Model

The CNN model has the following architecture:

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
batch_normalization_2 (BatchNormalization)	(None, 111, 111, 32)	128
conv2d_1 (Conv2D)	(None, 109, 109, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 64)	0
batch_normalization_3 (BatchNormalization)	(None, 54, 54, 64)	256
conv2d_2 (Conv2D)	(None, 52, 52, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 128)	0
batch_normalization_4 (BatchNormalization)	(None, 26, 26, 128)	512
conv2d_3 (Conv2D)	(None, 24, 24, 256)	295168
max_pooling2d_3 (MaxPooling2D)	(None, 12, 12, 256)	0
batch_normalization_5 (BatchNormalization)	(None, 12, 12, 256)	1024
flatten (Flatten)	(None, 36864)	0
dense_3 (Dense)	(None, 512)	18874880
batch_normalization_6 (BatchNormalization)	(None, 512)	2048
dropout_2 (Dropout)	(None, 512)	0
dense_4 (Dense)	(None, 256)	131328
batch_normalization_7 (BatchNormalization)	(None, 256)	1024
dropout_3 (Dropout)	(None, 256)	0
dense_5 (Dense)	(None, 1)	257

Total params: 19,399,873 (74.00 MB)

Trainable params: 19,397,377 (74.00 MB)

Non-trainable params: 2,496 (9.75 KB)

4.2 EfficientNetB0-based Model

The model architecture is based on the EfficientNetB0 as the backbone, with additional layers for fine-tuning and classification:

Model: "sequential"

Layer (type)	Output Shape	Param #
efficientnetb0 (Functional)	(None, 7, 7, 1280)	4049571
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
dense (Dense)	(None, 512)	655872
batch_normalization (BatchNormalization)	(None, 512)	2048
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131328
batch_normalization_1 (BatchNormalization)	(None, 256)	1024
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 1)	257

Total params: 4,840,100 (18.46 MB)
Trainable params: 788,993 (3.01 MB)
Non-trainable params: 4,051,107 (15.45 MB)

Key aspects of the architecture:

- EfficientNetB0 as the base model, known for its efficiency and performance in image classification tasks.
- Global Average Pooling to reduce the spatial dimensions of the features.
- Two Dense layers (512 and 256 units) with Batch Normalization and Dropout for regularization.
- Final Dense layer with a single unit and sigmoid activation for binary classification.

Training Process

- Both models were trained for 30 epochs with similar parameters:
- Optimizer: Adam
- Loss function: Binary Cross-Entropy
- Batch size: Approximately 30
- The custom CNN model had a learning rate reduction:
  Initial learning rate: 1e-4 (Reduced to 2e-5 from epoch 25 onwards)

5. Results and Evaluation

5.1 Custom CNN Model

Training and Validation Performance:

Final training accuracy: 93.89%
Final validation accuracy: 93.88%
Final training loss: 0.1607
Final validation loss: 0.1321

Test Set Performance:

	precision	recall	f1-score	support
0	0.93	0.92	0.93	61
1	0.92	0.93	0.93	60
accuracy			0.93	121
macro avg	0.93	0.93	0.93	121
weighted avg	0.93	0.93	0.93	121

Overall accuracy: 93%
Cataract (0): Precision 0.93, Recall 0.92, F1-score 0.93
Normal (1): Precision 0.92, Recall 0.93, F1-score 0.93

Confusion Matrix:

```
[[56  5]
 [ 4 56]]
```

5.2 EfficientNetB0-based Model

- Training Process
- The model was trained for 30 epochs with the following parameters:
- Optimizer: Adam with a learning rate of 1e-4
- Loss function: Binary Cross-Entropy
- Batch size: Approximately 30 (inferred from 13 steps per epoch for 393 training images)

Results and Evaluation

Training and Validation Performance

Final training accuracy: 94.66%
Final validation accuracy: 96.94%
Final training loss: 0.1142
Final validation loss: 0.1220

Test Set Performance

	precision	recall	f1-score	support
0	0.95	0.93	0.94	61
1	0.93	0.95	0.94	60
accuracy			0.94	121
macro avg	0.94	0.94	0.94	121
weighted avg	0.94	0.94	0.94	121

Overall accuracy: 94%
Class 0 (cataract): Precision 0.95, Recall 0.93, F1-score 0.94
Class 1 (normal): Precision 0.93, Recall 0.95, F1-score 0.94

Confusion Matrix
[[57 4]
[3 57]]

6. Comparison and Analysis

6.1 Model Performance

- The EfficientNetB0-based model slightly outperforms the custom CNN model:
    - EfficientNetB0: 94% accuracy on the test set
    - Custom CNN: 93% accuracy on the test set

- Both models show balanced performance across classes, which is crucial for a reliable diagnostic tool.

6.2 Model Convergence

Both models show good convergence, with the custom CNN exhibiting more fluctuations in validation accuracy during training. The EfficientNetB0 model appears to have a smoother learning curve.

6.3 Model Complexity

EfficientNetB0: 4,840,100 total parameters (788,993 trainable)
Custom CNN: 19,399,873 total parameters (19,397,377 trainable)

The custom CNN is significantly larger and more complex than the EfficientNetB0-based model.

6.4 Training Efficiency
The custom CNN required a learning rate reduction in the later epochs to improve performance, while the EfficientNetB0 model maintained a constant learning rate throughout training.

Additional Notes

The EfficientNetB0-based model is preferred for several reasons:

- Performance: It achieves slightly higher accuracy on the test set (94% vs. 93%).
- Efficiency: With fewer parameters, it's more computationally efficient and requires less memory.
- Generalization: The EfficientNetB0 model shows better generalization, with final validation accuracy (96.94%) higher than training accuracy (94.66%).
- Stability: The training process appears more stable, without the need for learning rate adjustments.

The custom CNN, while performing well, is overparameterized for this task. Its larger size increases the risk of overfitting, especially with a relatively small dataset.
The EfficientNetB0 model leverages transfer learning, benefiting from pre-trained weights on a large dataset. This allows it to achieve high performance with fewer trainable parameters and potentially better generalization to unseen data.

Conclusion and Future Work

The EfficientNetB0-based model demonstrates superior performance and efficiency in classifying cataract and normal eye images. Its high accuracy, balanced performance across classes, and computational efficiency make it more suitable for practical applications in automated cataract screening.

Future work could include:

- Fine-tuning the EfficientNetB0 model further, potentially with more advanced data augmentation techniques.
- Exploring other efficient architectures like MobileNet or EfficientNetV2.
- Investigating the model's performance on a larger and more diverse dataset.
- Implementing explainable AI techniques to provide insights into the model's decision-making process.

In conclusion, this comparison demonstrates the advantages of using a well-designed, pre-trained model like EfficientNetB0 for medical image classification tasks, especially when working with limited data.