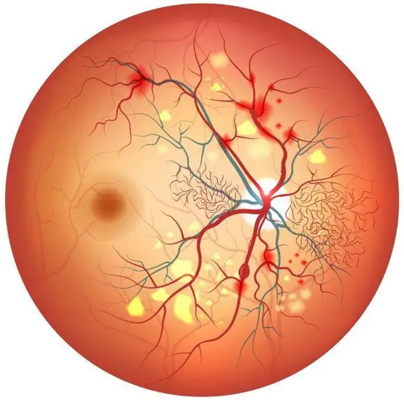
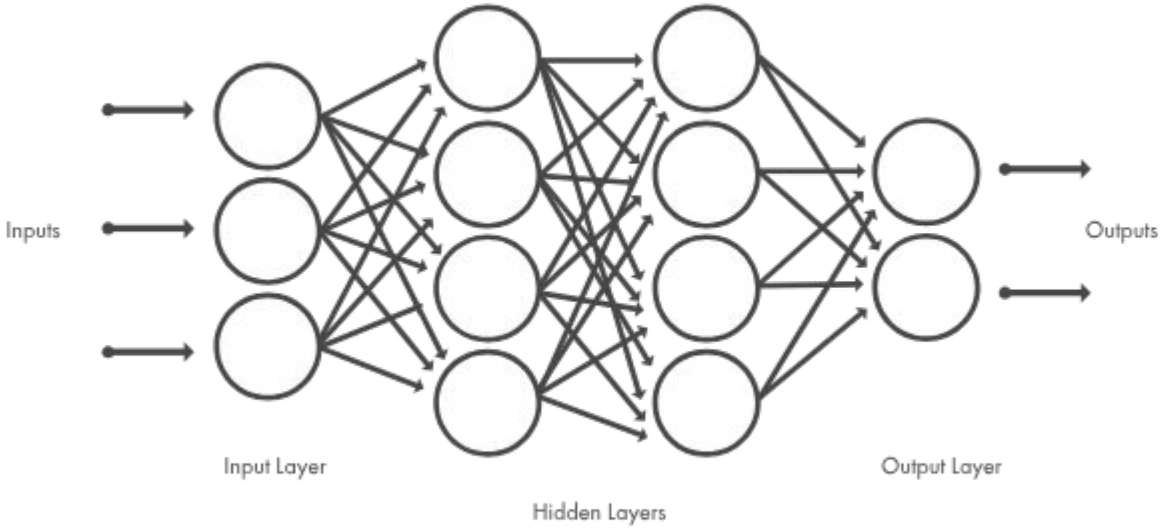
Classification of Diabetic Retinopathy using Deep Learning

Nick Weda
COMP 542 : Machine Learning

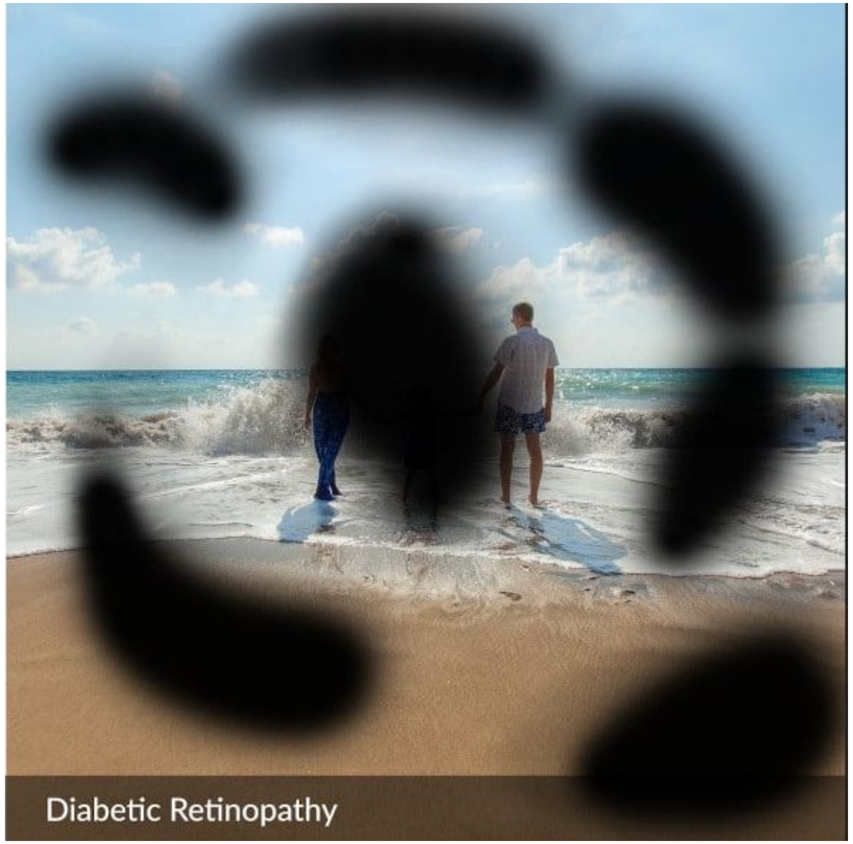




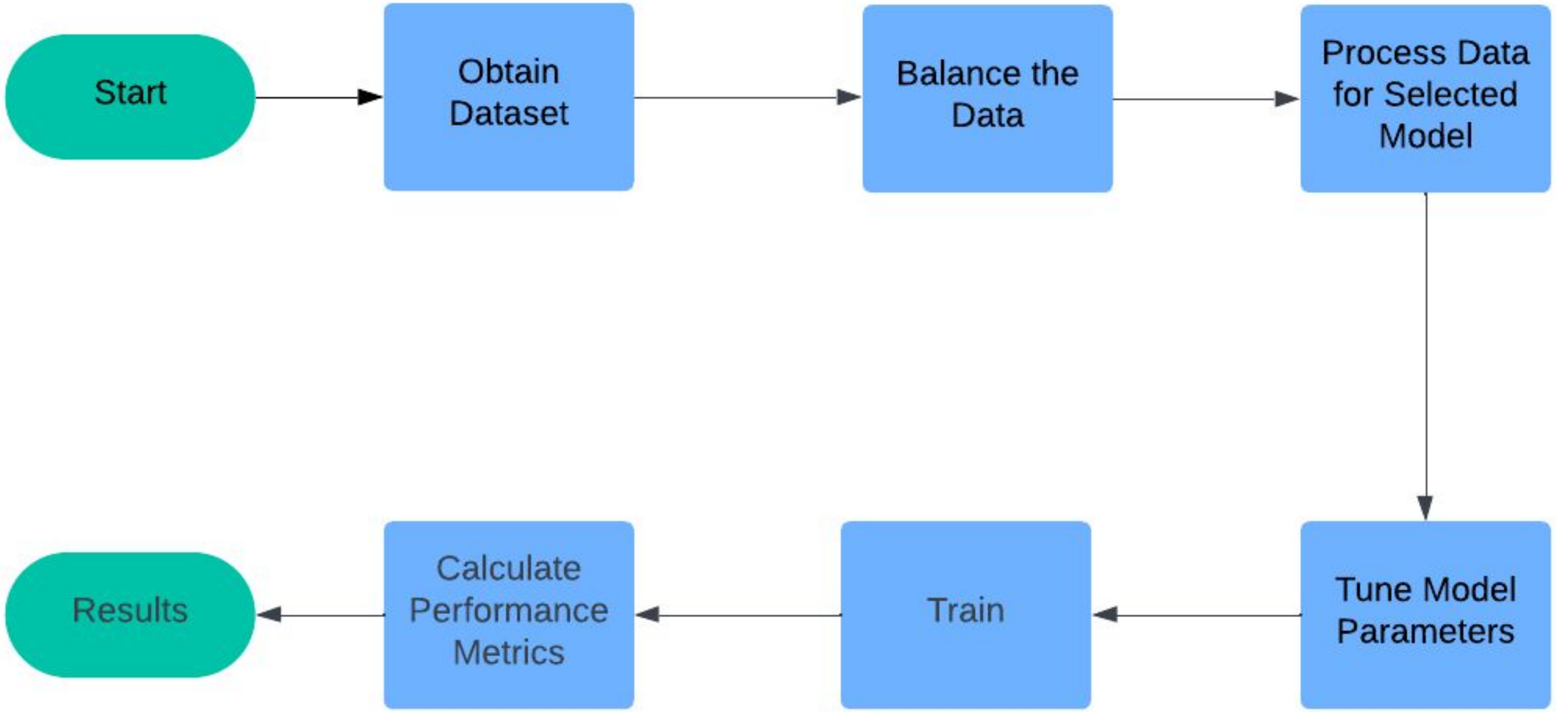
Department of Computer Science
California State University, Northridge

INTRODUCTION

- Diabetic Retinopathy is a complication of diabetes that can lead to blindness
- The disease is manageable if discovered and treated early
- Machine Learning can assist in the detection of early cases



METHOD


Start
Obtain Dataset
Balance the Data
Process Data for Selected Model
Tune Model Parameters
Train
Calculate Performance Metrics
Results

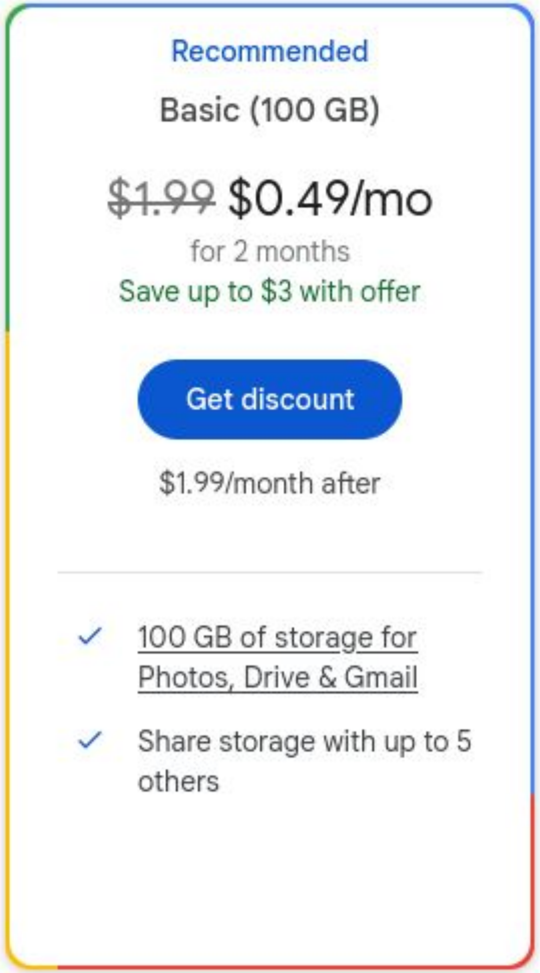
DATASET

- Dataset source
- Kaggle Competition - 2015 - $100,000 prize
- EyePacs - Provider of Data - Diabetic Retinopathy Screening Program in California

DATASET

- <u>Dataset Size</u>
- 37.9 GB
- 35,126 images.
- 17,563 people. Left and Right eye.
- High resolution images of varying size. 4k+
- Accessing the Dataset:

  Wrote a small script utilizing command line tools to save to the cloud

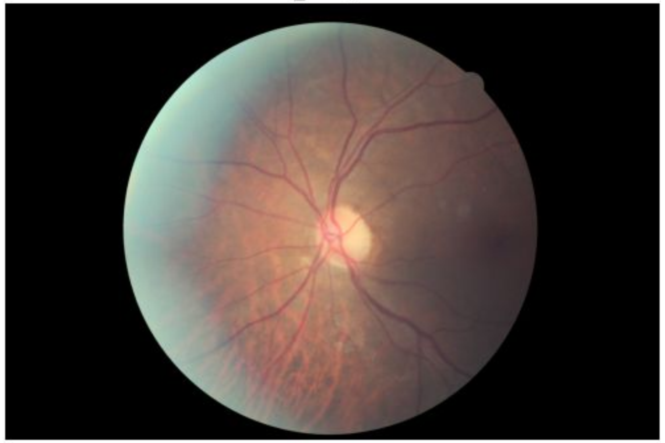


DATASET

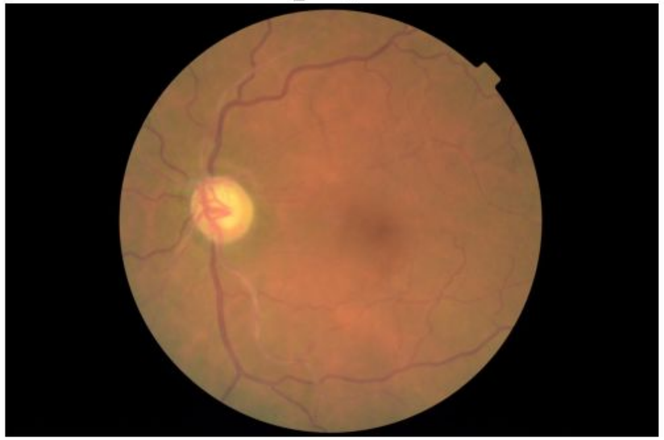
- <u>Data Exploration</u>
- The disease is on a severity scale of 0-4,where 0 is healthy and 4 is severe

File: 16_left | Label: 4

File: 10_left | Label: 0

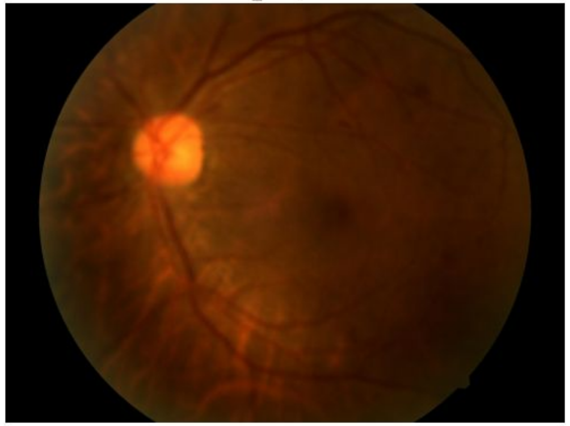


File: 15_left | Label: 1



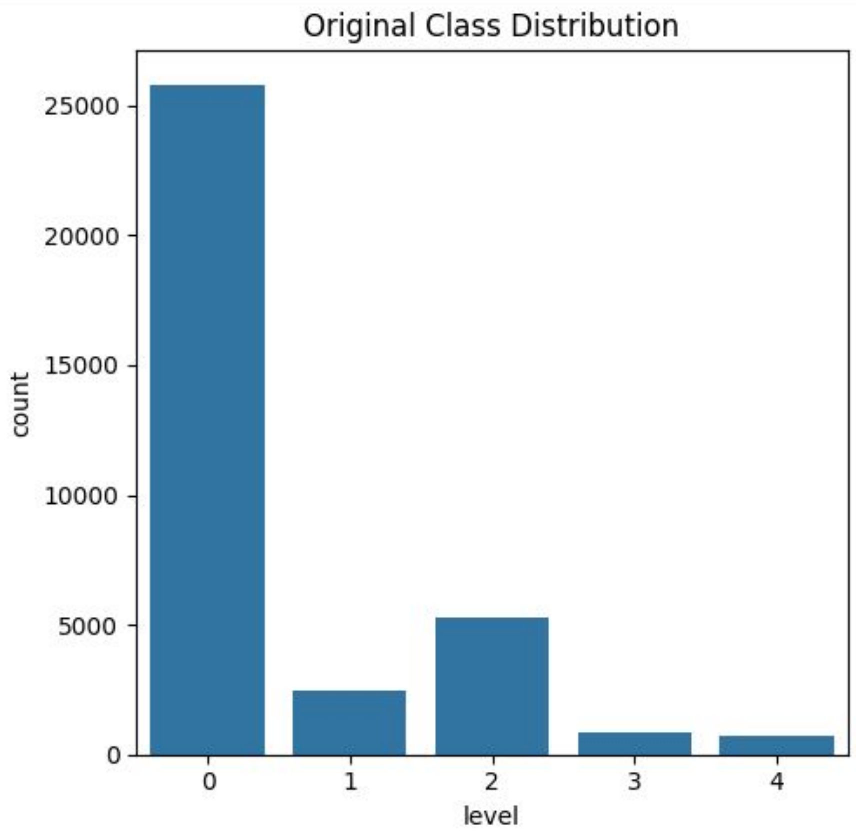
File: 15_right | Label: 2

File: 99_left | Label: 3



DATASET

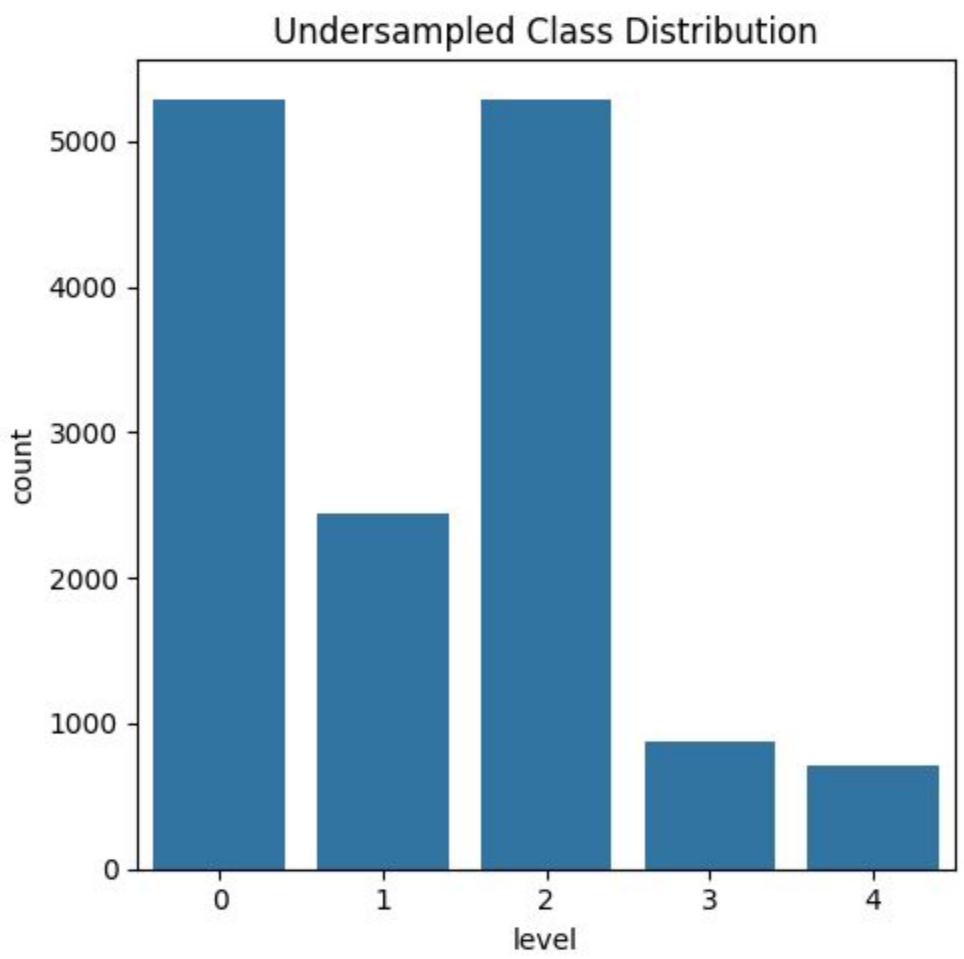
- <u>Number of Samples</u>
- Data was hand labeled by medical professionals
- Dataset is highly imbalanced
- Characteristic of Medical Data
- Most surveyed patients are healthy



DATASET

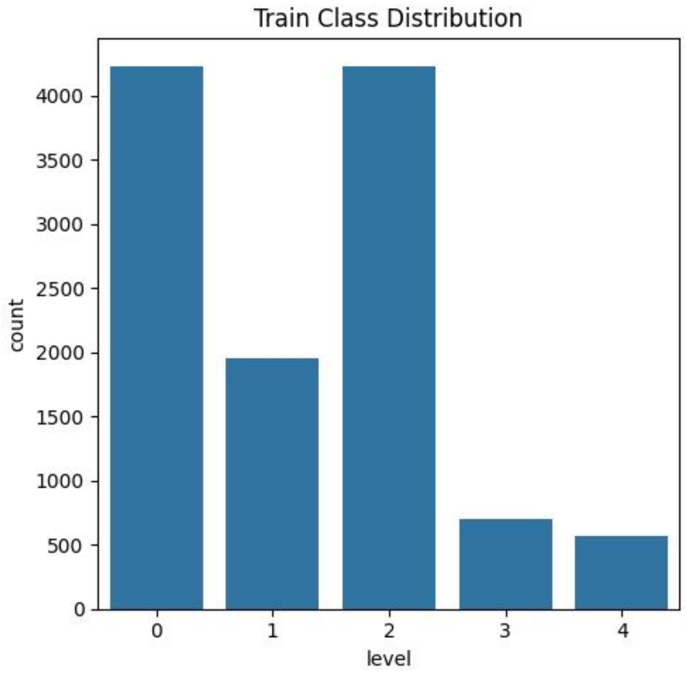
- <u>Combatting Imbalance</u>
- Undersample the majority class
- Use the Scikit Imbalanced-Learn Library
- Randomly select majority class examples to keep
- New Dataset Size: 14,608 (~40% Decrease)

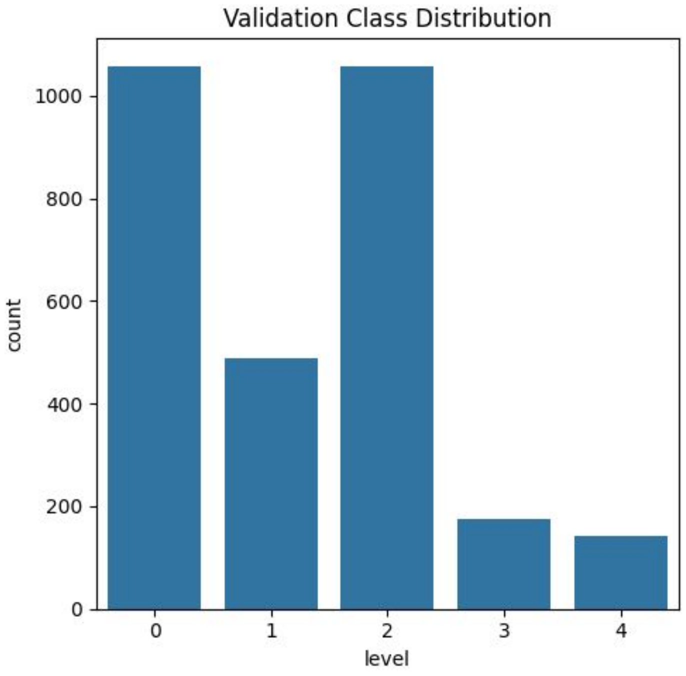
Imbalanced learn



DATASET

- <u>Combatting Imbalance</u>
- Train-Test Split
- Stratified Sampling to maintain the data's distribution





Choice of Model

- Keras vs. PyTorch APIs
- Keras has a simpler interface
- Includes a set of Convolutional Neural Network architectures
- Based on the dataset size:
- ResNet-50, MobileNetV2, **EfficientNetB0 - 237 Layers**

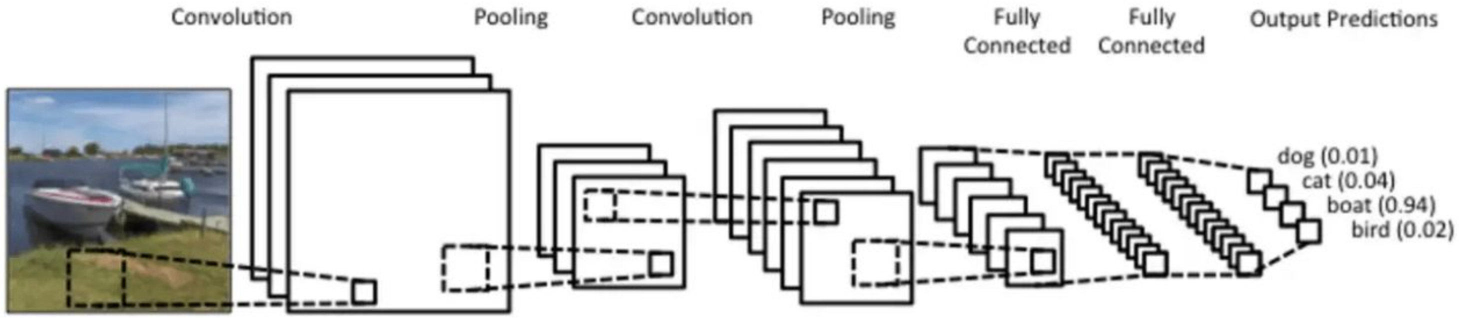


Image Pre-Processing

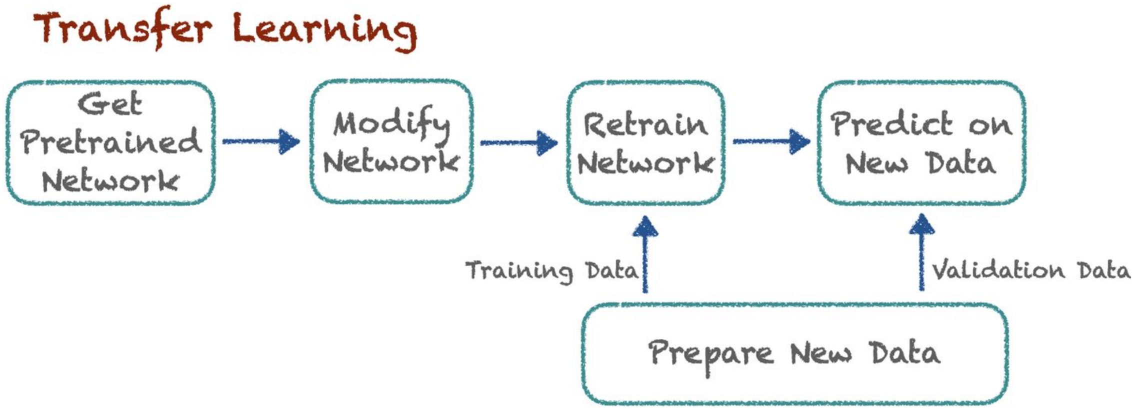
- All 3 previously mentioned models require 224x224 images
- Careful cropping to avoid distorting the image
- Crop to **Region Of Interest** (Capture the retina)
- Retain **Aspect Ratio** (Width/Height Proportion)
- Add Black Border if necessary to preserve aspect ratio

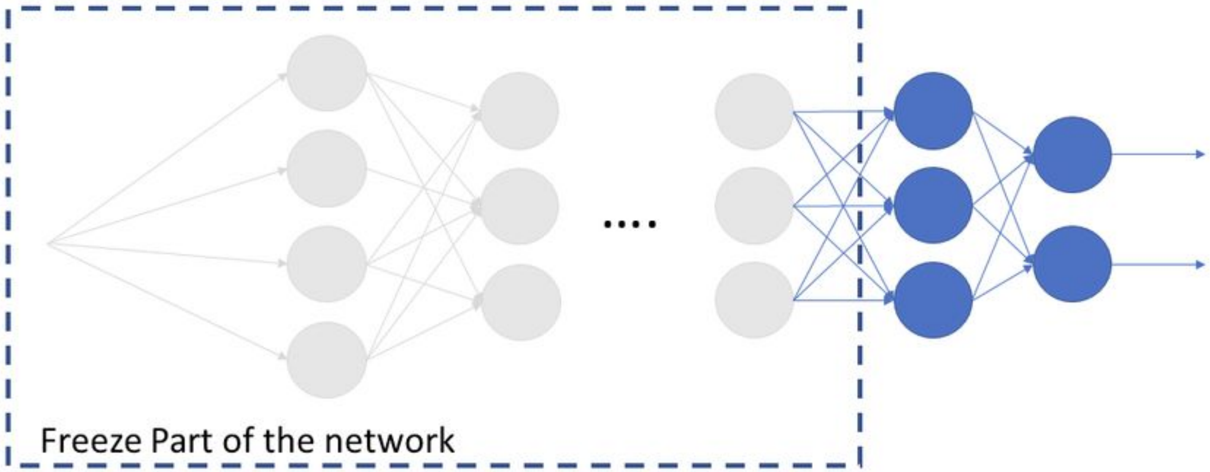
Image Pre-Processing

- We can take the opportunity to apply some light **augmentations**
- Random Horizontal Flip, Random Brightness, Random Zoom
- Save future computation by transforming once and then saving

Model Parameters

- Use pre-trained weights on ImageNet. Use **Transfer Learning**. Saves computation.
- Freeze beginning layers, basic pattern recognition.
- Weights will not be updated.
- Unfreeze the last layers (50/237), domain specific application.

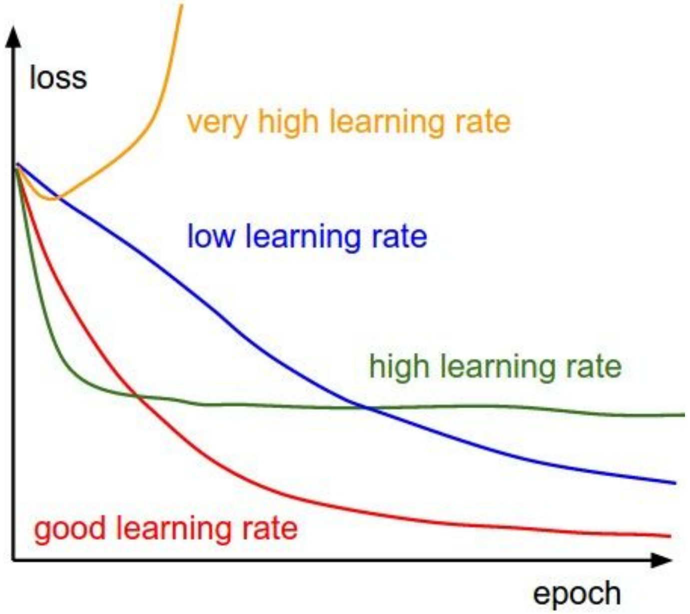




Model Parameters

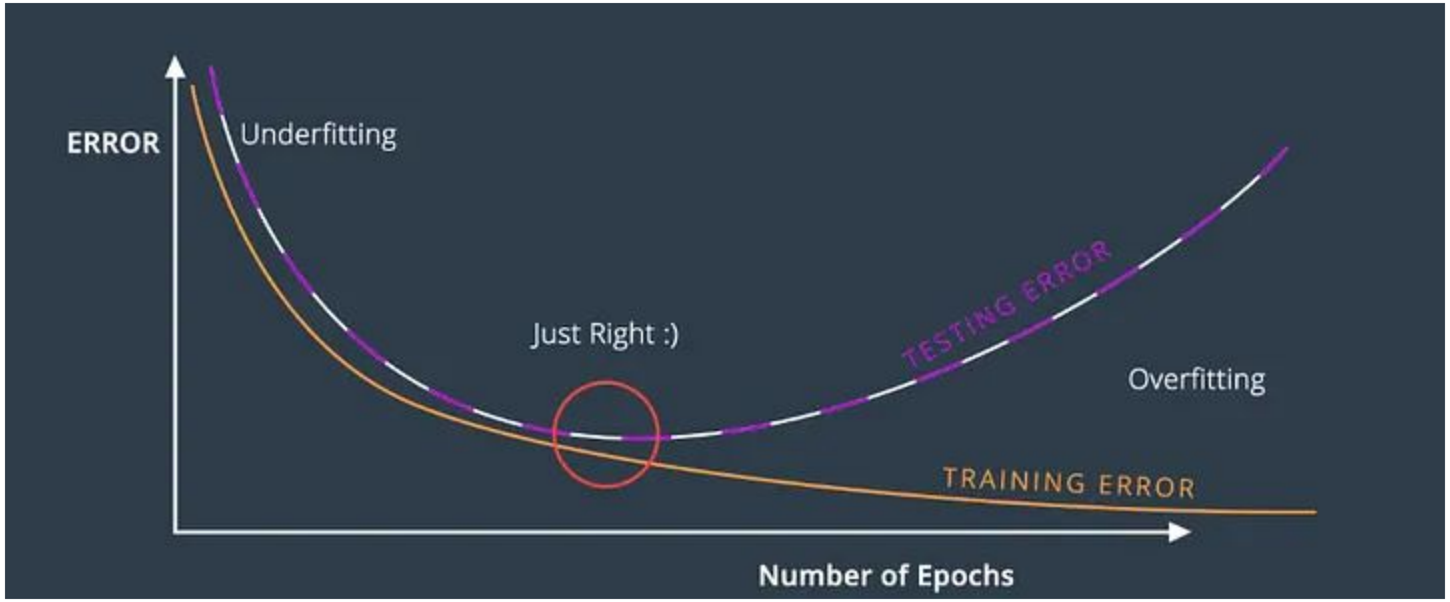
- Transfer learning adjusts what our **learning rate** should be. How much the weights change on each pass.
- Conduct a Learning Rate Range Test (LRRT). Tests the model on several rates.
- Ideal in this case = 0.0001 (1e-4)
- **Loss** is the difference between predicted and actual and should decrease over time

Keras Tuner



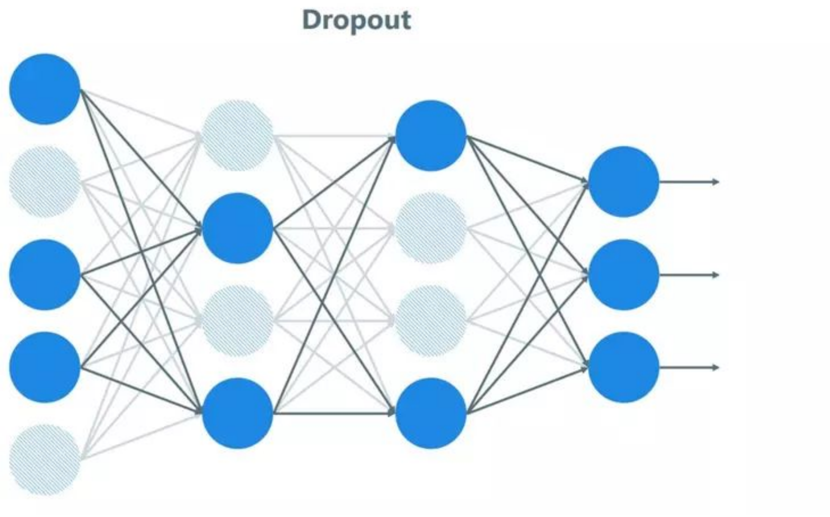
Model Parameters

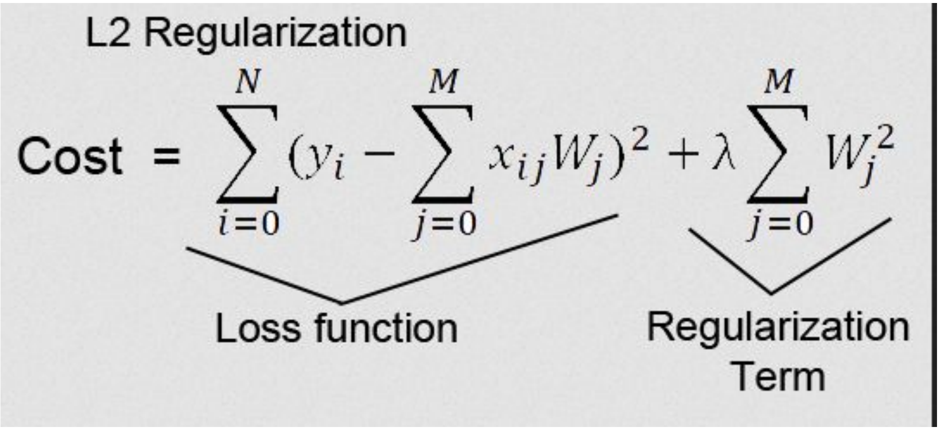
- Other tuning parameters
- **Batch Size** - How many images fed in at once. Higher = faster training. (128)
- **Epochs** - How many entire passes of the training data through the network. (15)



Control Overfitting

- EfficientNetB0 is a fairly complex model. 14,000 images on on the smaller end for it.
- **Dropout:** Randomly dropout a fraction of neurons on every forward pass (20%)
- **L2 Regularization:** Add a penalty term to the loss function (cross entropy), pushing the weights to a smaller value





RESULT

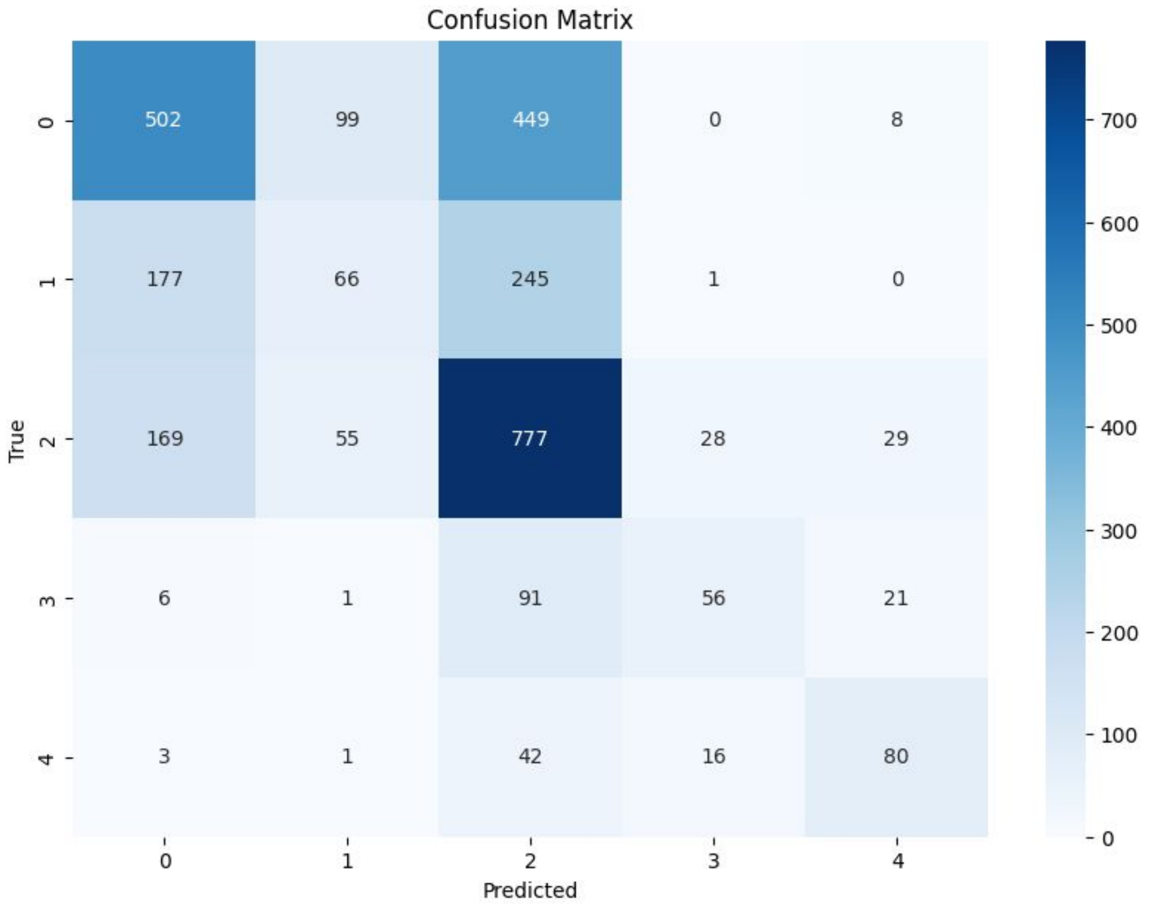
- <u>Performance Metrics</u>
- Accuracy = (TP + TN) / Total
- Precision (of believed positive) = TP/ (TP +FP)
- Recall (of actual positive) = TP / (TP+FN)
- F1 Score - Harmonic Mean
- Support - occurrence of class
- Macro Average - across each class
- Weighted Average - across each class (proportion)

Classification Report:

	precision	recall	f1-score	support
0	0.5504	0.6248	0.5852	1058
1	0.2767	0.2331	0.2531	489
2	0.5504	0.5520	0.5512	1058
3	0.4500	0.3600	0.4000	175
4	0.6019	0.4577	0.5200	142
accuracy			0.5089	2922
macro avg	0.4859	0.4455	0.4619	2922
weighted avg	0.5011	0.5089	0.5031	2922

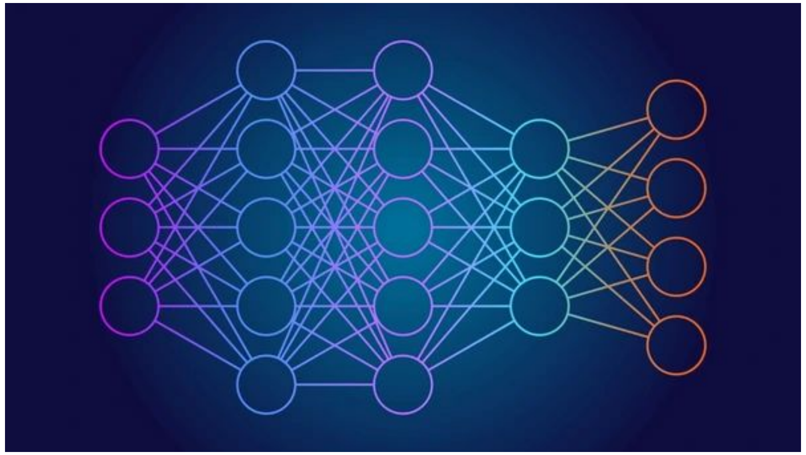
RESULT

- <u>Performance Metrics</u>
- Only predicted the dominating classes with the most accuracy.
- Need more data for 1,3,4.
- It doesn't exist for 3,4.
- Training time ~30 min.
- Model was overfitting, train accuracy was high at 87%+.

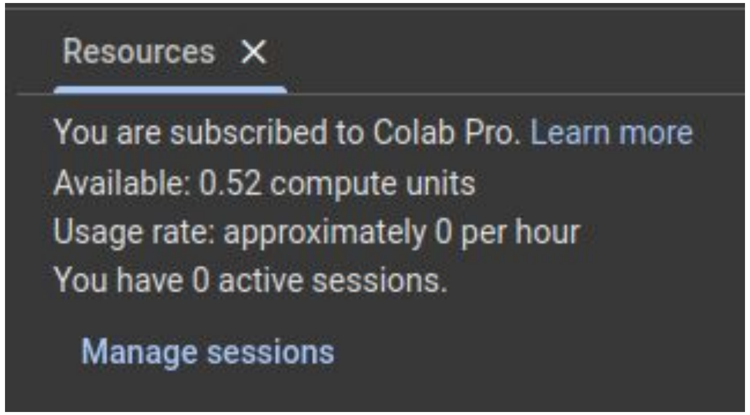


RESULT

- <u>Possible Improvements:</u>
- Build a network from scratch instead of relying on transfer learning.
- Implement proper SMOTE oversampling of images, creating synthetic images.
- Apply augmentations on the fly for more variability of data.
- Develop a novel feature extraction algorithm that can be used for accurate classification.



Closing Remarks



Thank You