Retinal Disease Classification

CSE 472 Project

Anik Islam (1705104)

Soham Khisa (1705120)

Supervisor: Dr. Mohammad Saifur Rahman

Problem Definition

The project's objective is to use retinal images to recognize and classify various retinal illnesses, including age-related macular degeneration and diabetic retinopathy.

A label identifying the presence or absence of a certain illness will be produced by the system after receiving an input retinal picture.

Dataset

- Collected from Kaggle
- 1920 training images, 640 validation images, 640 test images
- Three csv files containing labels for training, validation and test images
- We used 1920+640 = 2560 images for training

Architecture

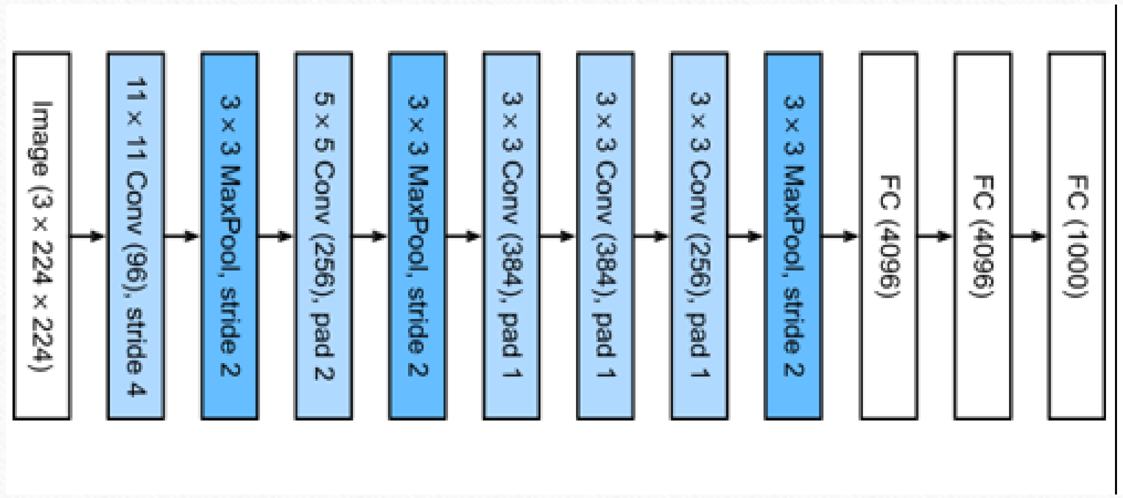
AlexNet

5 Convolutional Layers

3 Fully Connected Layers

3 Max Pooling Layers

ReLU Activation



We added an extra Fully connected layer at the end of size 45 and rounded the values of each node after activating by sigmoid function

Loss Function

The loss function we used is called **"CrossEntropyLoss"**

Cross-entropy loss measures the dissimilarity between the true labels and predicted probabilities output by a model. It is commonly used in classification problems where the model must predict one of multiple possible classes.

$$H(P^* | P) = -\sum_i P^*(i) \log P(i)$$

$P(i) = P(y | x; \theta)$ => Predicted class distribution
$P^*(i) = P^*(y | x)$ => True class distribution

Loss Function

CrossEntropyLoss

The goal is to make *P* and *P** distributions as close as possible.

KL (Kullback–Leibler) divergence

A type of statistical distance, a measure of how one probability distribution *P* is different from a second, reference probability distribution *Q*.

Naturally, we would like to minimize *KL divergence* between *P* and *P**.

Loss Function

$$
\begin{aligned}
D_{KL}(P^*||P) &= \sum_y P^*(y|x_i) \log \frac{P^*(y|x_i)}{P(y|x_i;\theta)} \\
&= \sum_y P^*(y|x_i) \left[\log P^*(y|x_i) \; - \log P(y|x_i;\theta)\right] \\
&= \sum_y \overbrace{P^*(y|x_i) \log P^*(y|x_i)}^{\text{DOESN'T DEPEND ON } \theta} - \sum_y P^*(y|x_i) \log P(y|x_i;\theta)
\end{aligned}
$$

$$
\underset{\theta}{\operatorname{argmin}}\, D_{KL}(P^*||P) \quad \equiv \quad \underset{\theta}{\operatorname{argmin}} \quad -\sum_y P^*(y|x_i) \log P(y|x_i;\theta)
$$

Loss Function

Advantages of **CrossEntropyLoss**

- Cross-entropy loss can handle multi-class classification problems with multiple output classes. It does this by using a softmax function that produces class probabilities for each output class.
- It can lead to fast convergence in the training process of neural networks.
- It provides a probabilistic output, which can be useful in applications such as image segmentation or object detection.
- Cross-entropy loss is differentiable, which allows for backpropagation and optimization of the model parameters during training

Performance

Testing Performance:

Epoch: 20 , Learning rate : 0.001 , Batch size=100

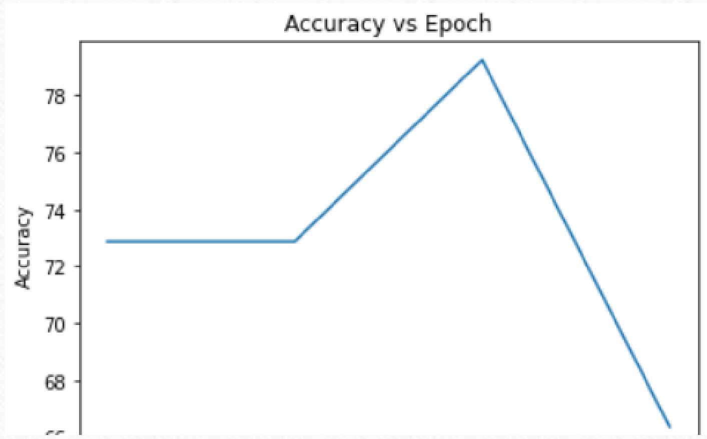
- Correctly predicted cells: 22472
- Total cells: 28800
- Accuracy : 78.02777777777777 %
- Sensitivity/Recall: 0.8415545590433483
- Specificity: 0.7788205182894316

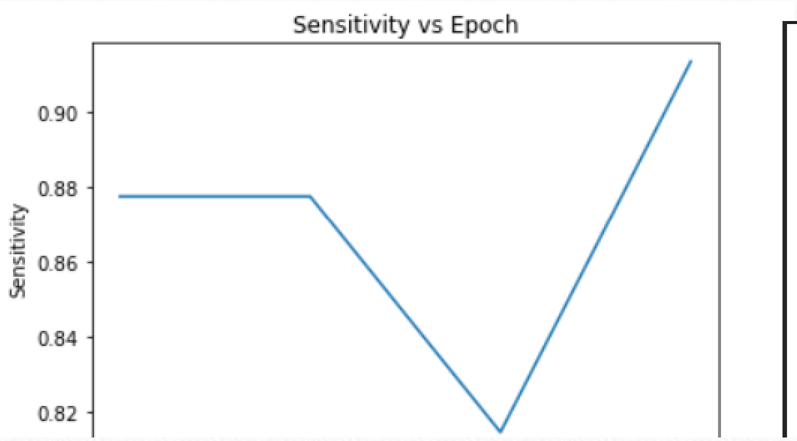
$$\text{Specificity} = \frac{\text{True Negative}}{\text{True Negative} + \text{False Positive}}$$

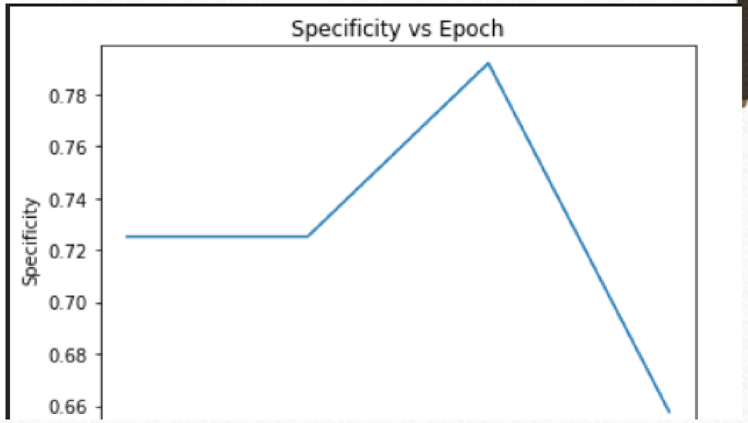
$$\text{Sensitivity} = \frac{\text{True positive}}{\text{True positive} + \text{False Negative}}$$

Performance

Epoch : 1-5 , Learning rate: 0.001 , Batch size : 100

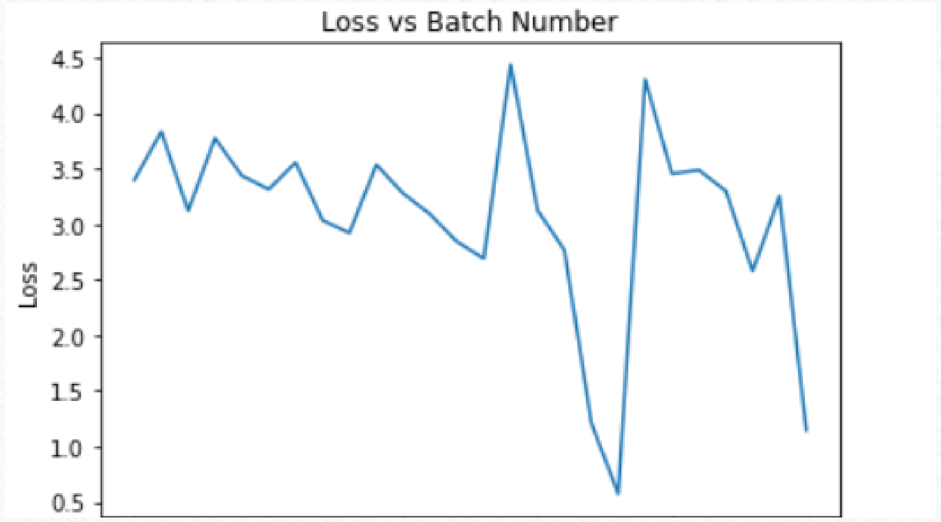


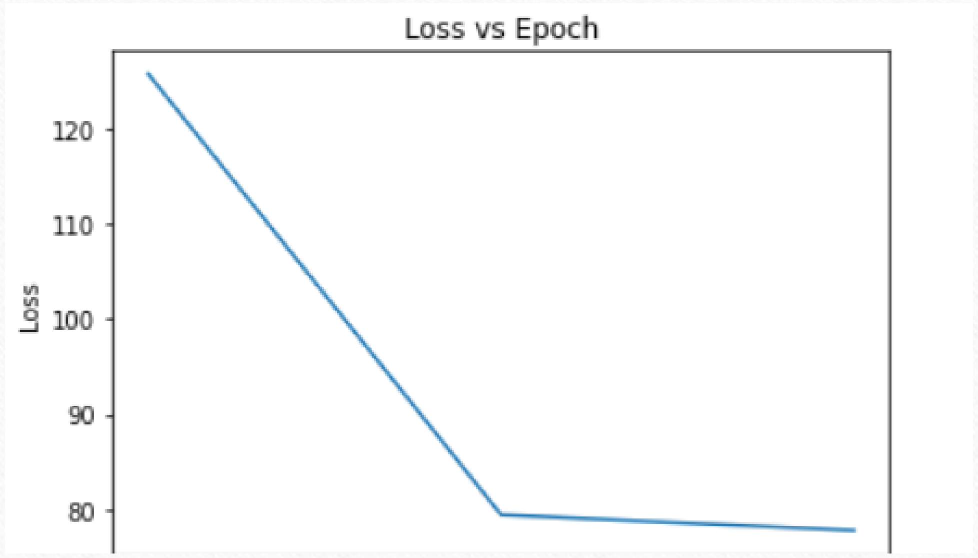




Performance

- Epoch = 3 , Batch size =100





Comparison

Method	Sensitivity	Specificity	Accuracy
MR	0.7344	0.9764	0.9456
SVM	0.7567	0.9788	0.9527
DRDL	0.7932	0.9789	0.9673
AlexNet	0.8416	0.7788	0.7803

Discussion

- Initially, our attempt was to create a Deep Belief Network, which is essentially a series of Restricted Boltzmann Machines (RBM) stacked on top of each other. RBMs are typically utilized for unsupervised learning. However, the output we obtained from the DBN model was not satisfactory.

- Then we implemented AlexNet architecture which is a CNN(Convolutional Neural Network) architecture.

Discussion

- We used BCEWithLogitsLoss as loss function at first.It combines 'BCEloss' and 'sigmoid' function together. But somehow all the binary output turned out zero.We tuned parameters like learning rate,epochs but there was not seen much improvement

- Then we used CrossEntropyLoss as the loss function which provided reasonable output

Discussion

- We did some preprocessing on images. We reshaped them to (224,224,3) shape where 224 is the height and weight and 3 is the number of channels
- We calculated three metrics – Accuracy, Sensitivity and Specificity