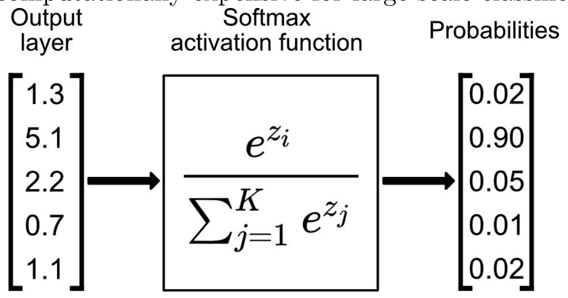
SAiDL Summer 2023 Induction Assignment 1

Vimarsh Shah

May 2023

1 Introduction

Neural networks have revolutionized image classification tasks, and the softmax function has become a popular choice for generating probabilistic outputs. However, its time complexity grows linearly with the number of classes, making it computationally expensive for large-scale classification problems.


Output layer
Softmax activation function
Probabilities
1.3
5.1
2.2
0.7
1.1
$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
0.02
0.90
0.05
0.01
0.02

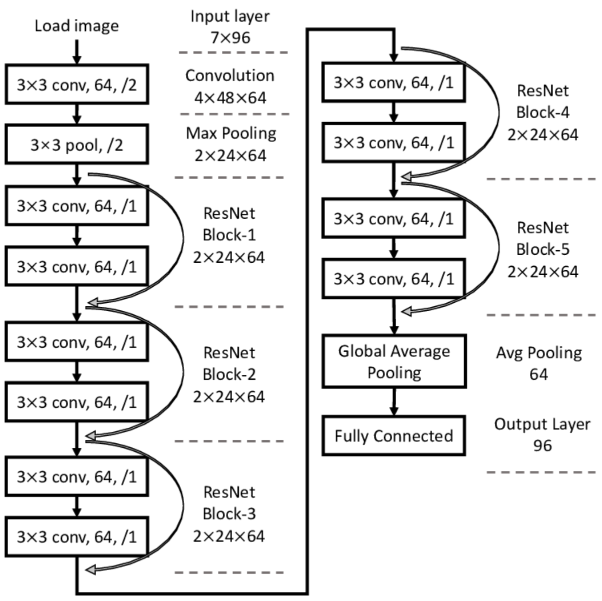
To address this issue, alternative softmax implementations have been proposed, such as logsoftmax and Gumbel-Softmax, which aim to reduce the computational complexity while maintaining performance. In this article, I try to implement and understand the performance differences between various implementations.

2 Methodology

In this study, we employ the CIFAR-100 dataset, consisting of 100 classes of images, to compare the performance of different softmax variations in CNN models. First I developed a baseline CNN model with the standard softmax implementation. Then, created additional models using the same architecture but replacing the softmax layer with logsoftmax and Gumbel-Softmax, etc.

Architecture of the model

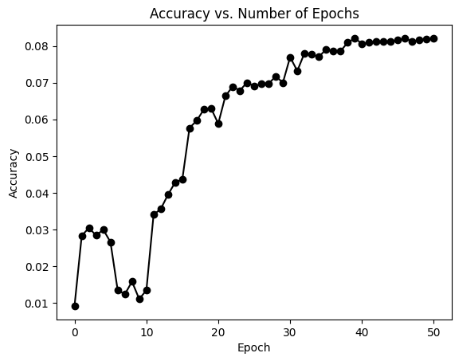
The CNN is architected on Resnet12 model. I tried creating my own CNN with several Convolution and activation layers, but accuracy was less compared when compared to already established layer architectures as that of Resnet12.

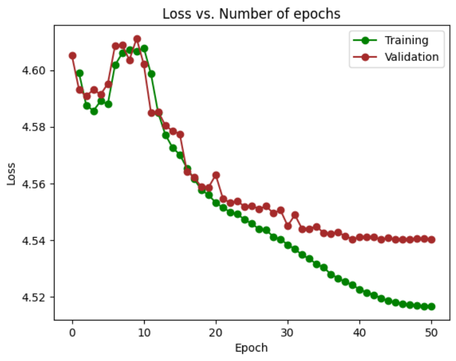


2.1 Standard Softmax

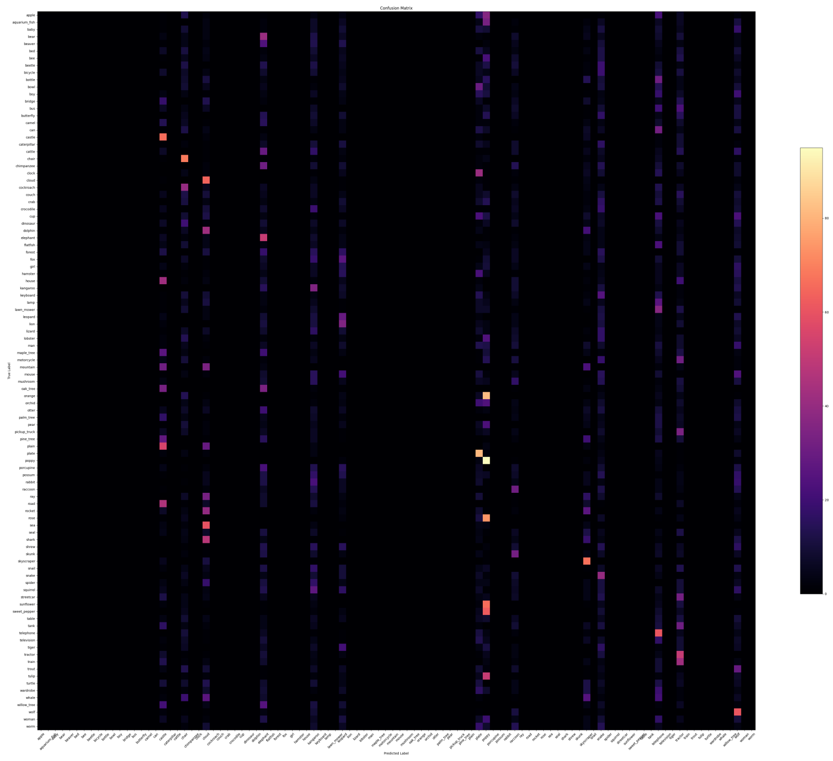
Results on training on standard softmax layer:

Accuracy	Precision	Recall	F1 Score
8.09%	0.0115	0.0817	0.02





Confusion Matrix



Using just 20 Super Classes

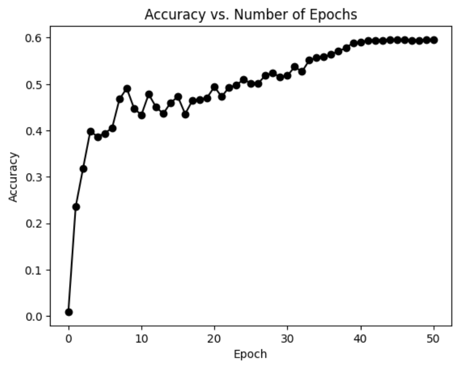
When I trained the model on just the 20 super classes, the accuracy of Softmax for 50 epochs was 45 , which decreases drastically on increasing it to 100 classes. Further showing that the complexity increases drastically with increasing the number of classes.

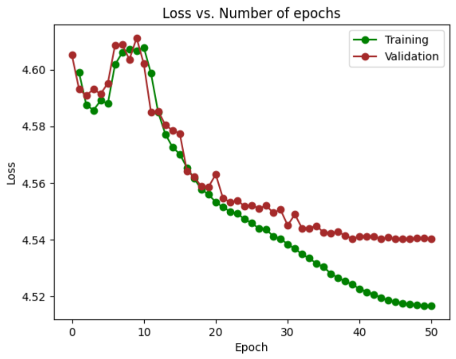
2.2 LogSoftmax

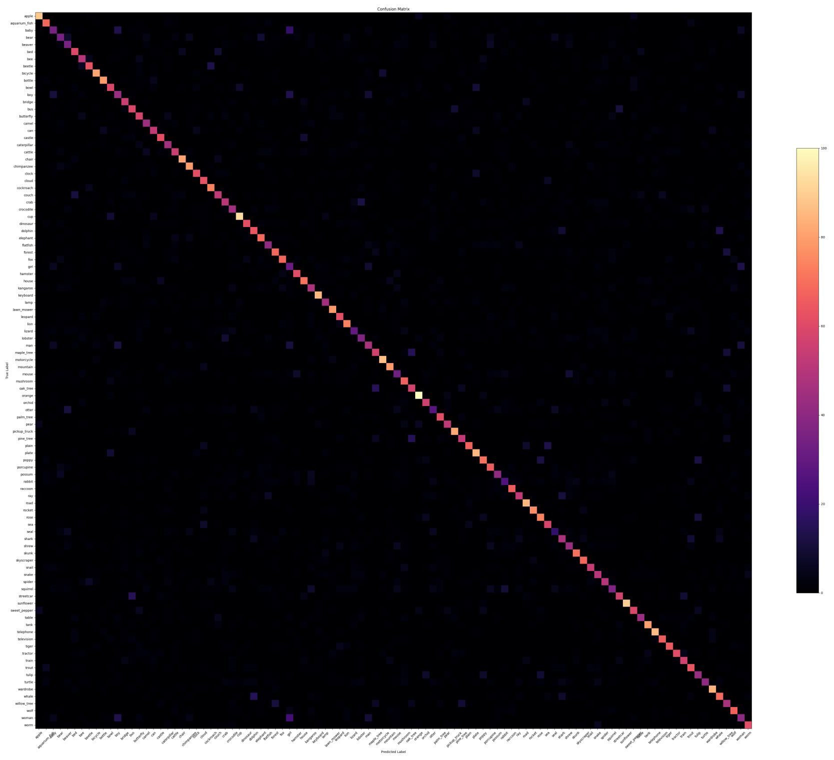
On implementing LogSoftmax instead of standard softmax, the accuracy improved drastically. This seems to be due to the fact that when using normal softmax, the vectors being exponentiated, increase the error by a lot; heavily hindering the performance of the model. But log softmax, simply converts vectors into probability, rather then exponential average. Not only does this reduce complexity but maintains the relative difference, which in this case seems to have helped.

Accuracy	Precision	Recall	F1 Score
60.16%	0.5922	0.5987	0.5931

Confusion Matrix







2.3 Gumbel Softmax

This variation of softmax is just like standard softmax with a temperature term, ”reducing” the vectors.

![](assets/lolgumbelsoftmax.png)

On training the model on this I could see an increase in the accuracy of the model on increasing the value of ****.

Accuracy	Precision	Recall	F1 Score
60.16%	0.5922	0.5987	0.5931

The above accuracy was with the value of = 3, on having value ¡=1 the accuracy didn't even reach 8%. Overall the learning rate wrt the number of epochs didn't change much.

![](assets/models/gumbel-accuracyepoch.png)

![](assets/models/gumbel-lossepoch.png)

2.4 Log Gumbel Softmax

2.5 Pseudo Softmax

What if instead of have e^z we have 2^z in the softmax function. This does not change a lot in terms of mathematical operations, but does help in computational scaling. As by the use of bitwise operator computers can quickly calculate 2^z.

3 Hypothesis

3.0.1 Reason for low performance of Softmax, when compared to others

4 References

http://cs231n.stanford.edu/reports/2017/pdfs/130.pdf