Deep Learning: Theory and Practices

Siddik Ayyappa
Sudheer Reddy
Vithesh Reddy

The Curious Case of Convex Neural Networks

Deep Learning Final Project

Introduction

- The paper focuses on the class of neural networks where the output of the neural network is a convex function of the inputs. Input Output Convex Neural Networks (IOCNNs).
- With some ingenious constraints applied on the Vanilla Neural Networks, one would be able to impose the convexity constraints on the same.
- IOCNNs have been observed to overcome the problem of overfitting and to self-regularize. They have outperformed MLPs and have achieved the baseline accuracies of Convolutional Architectures. They have been robust to noise in the labels of the data.

Introduction

Related Work

- **Simple Convex Models** : This consists of learning a piecewise linear function (convex), this can be done with the help of multiple hyperplanes (linear classifiers).
- **Generalisation of Deep Networks** : General conventions of machine learning suggest that over-parametrisation leads to overfitting.
- A neural net that can fit any data, will have a worse generalisation, when compared to others which fit a restricted space of data. The paper proposes a way of restricting the data which the neural network can fit. The methods leads to improvement of generalisation in practice.

Input Output Convex NNs

IOCNNs

Getting Started

- The problem at hand is to design a neural net, where the output is a convex function of the input
- Looking the first hidden layer. We can say that the output of the first layer is an Affine Mapping of the Input. Hence Convexity is preserved.

$$h^{(1)} = \phi(Wx + b)$$

- For the subsequent layers, to preserve the convexity with respect to input. We have to ensure 2 things.
  - $w_{ij}^{(2:k+1)} \geq 0$
  - ϕ is a convex and non-decreasing function.

IOCNNs
More Details

- The proof of convexity comes from the properties of operators that …
  - $f(g(x))$ is convex if g is convex and f is non-decreasing and convex (activation)
  - Non-Negative sum of convex functions is convex (pre-activation)
- The activation function we use is ELU. ReLU function with the non-negative weight constraints restrict the hidden neurons from identity mapping. [Amos et al.]
- ELU allows negative values, and hence enables identity mapping. These constraints allow us use the convex counterparts of the normal NNs, without architectural changes.

IOCNNs

Convexity as Self Regularizer

- **Self-Regularization** : Property of network which allows itself some functional constraints.
- Consider a quadratic classifier:
  - $f(x_1, x_2) = w_1 x_1^2 + w_2 x_2^2 + w_3 x_1 x_2 + w_4 x_1 + w_5 x_2 + w_6$
- For a multivariable function to be convex with respect to all of it's inputs …
  - The second partial derivative with respect to all the variables has to be non-negative.
  - The Hessian Matrix of the function has to be positive semi-definite.

Continued …

- $\frac{\partial^2 f}{\partial x_1^2} = 2w_1 \geq 0 \qquad \frac{\partial^2 f}{\partial x_2^2} = 2w_2 \geq 0 \qquad \frac{\partial^2 f}{\partial x_1 x_2} = \frac{\partial^2 f}{\partial x_1 x_2} = w_3$
- Characteristic Polynomial of the Hessian is …
  - $\lambda^2 - 2(w_1 + w_2)\lambda + (4w_1 w_2) - w_3^2$
- If both the zeros are +ve, then $4w_1 w_2 - w_3^2 \geq 0$, therefore
- $-2\sqrt{w_1 w_2} \leq w_3 \leq 2\sqrt{w_1 w_2}$
- Which means we are reducing the hypothesis space.

IOCNNs

Convexity as Self-Regularizer

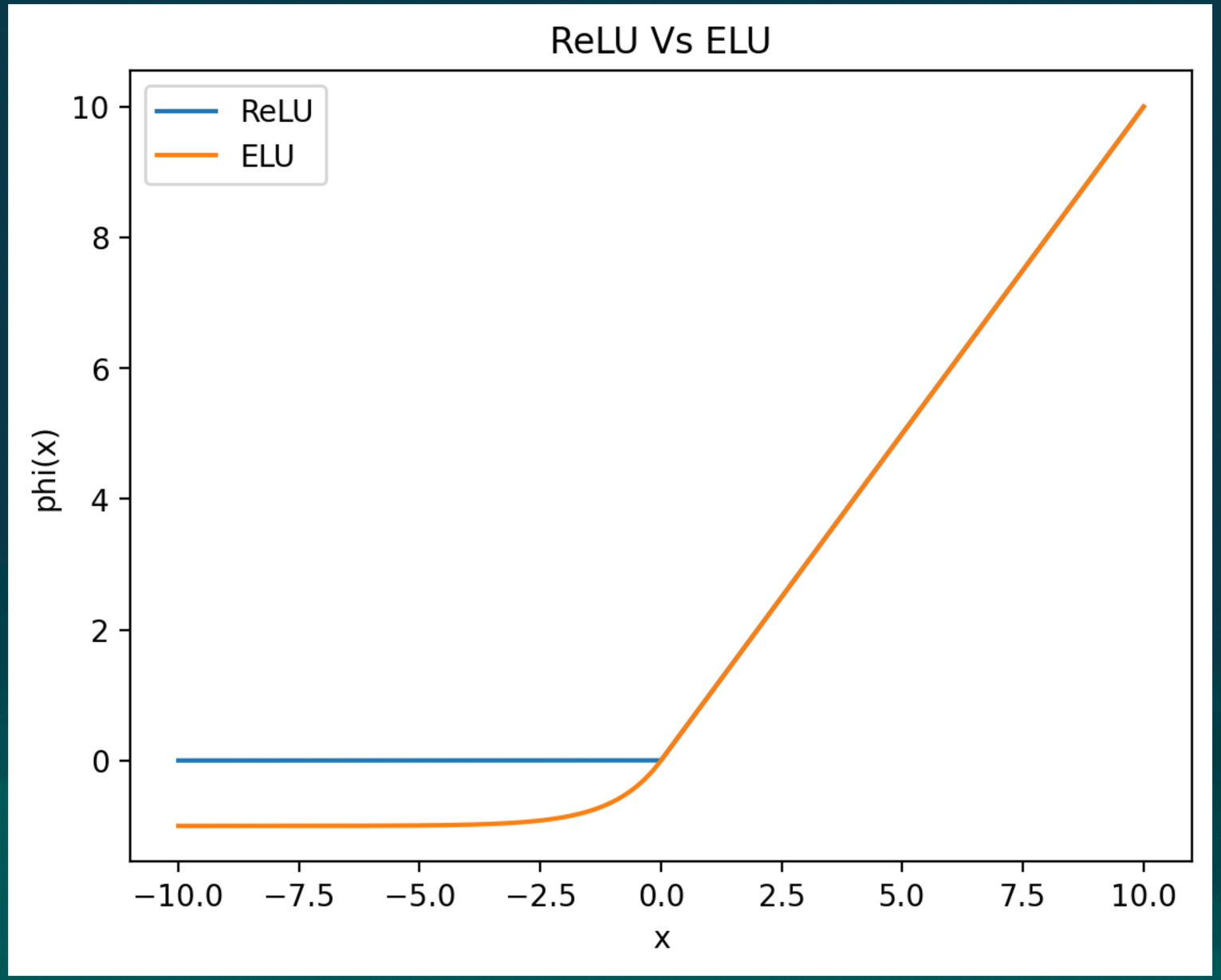
- **VC - Dimesnion** : The highest number of points that can be shattered using a set of functions is the VC - Dimesnion of the same.
- Higher VC dimension, implies that the classifier would be able to fit higher number of points, meaning more potential overfitting.
- On adding a sign constraint, the VC-dimension of a linear classifier in $\mathbb{R}^d$ reduces from $d + 1$ to d.
- This can be looked as regularisation at work since, the weight constraints have been reducing overfitting by producing a simpler model, (lower VC dimension).

IOCNNs

Learning Non Convex Decision Boundaries

- Since the output is convex w.r.t the input, we can say that the neural net learns n convex functions (discriminant functions) in the output layer, each corresponding to a class. During inference, the label is assigned to the highest function.
  - It must be noted that using softmax function in the output does not distort the convexity.
- Function would also be able to learn complex (non-convex) decision boundaries too. Even if each discriminant function of each class is convex.
- If $f(x)$ and $g(x)$ are convex $h_{fg}(x) = f(x) - g(x) = 0$, is not necessarily convex.

ReLU Vs ELU



Experiments and Results

Experiments

- We have tested the convex counterparts of AllConv and MLP of specified architecture in the paper.
- **Datasets** : MNIST, FMNIST, Spambase (From archives), CIFAR-10, CIFAIR - 10
- **Randomised Labelling** : We have trained AllConv, MLP, and their respective convex counterparts on CIFAR-10 with 100% random labels.
- **Weight Constraints** : We have tried 3 types of weight constraints …
  - Clipping
  - Absolute
  - Exponentiation
- Exponentiation gave the best results among all the other ones.

Experiments

- All the weight constraints have been experimented on the convex counterparts of the specified MLP or rather the IOC-MLP, with CIFAR-10.
- Instead of ReLU, we use ELU as activation function in IOC-NNs.
- The MNIST and FMNIST datasets have been used to compare the performances of MLP Vs IOC-MLP.
- The CIFAIR-10 (Duplicate Free Image) dataset has also been used in the experiments.
- The effectiveness of the exponentiation of weights can be explained by preservation of the rank of weights after changing, while others do not.
- The redundancy of weights is lower in exponentiation, than the other, which would endorse the constraint’s effectiveness.

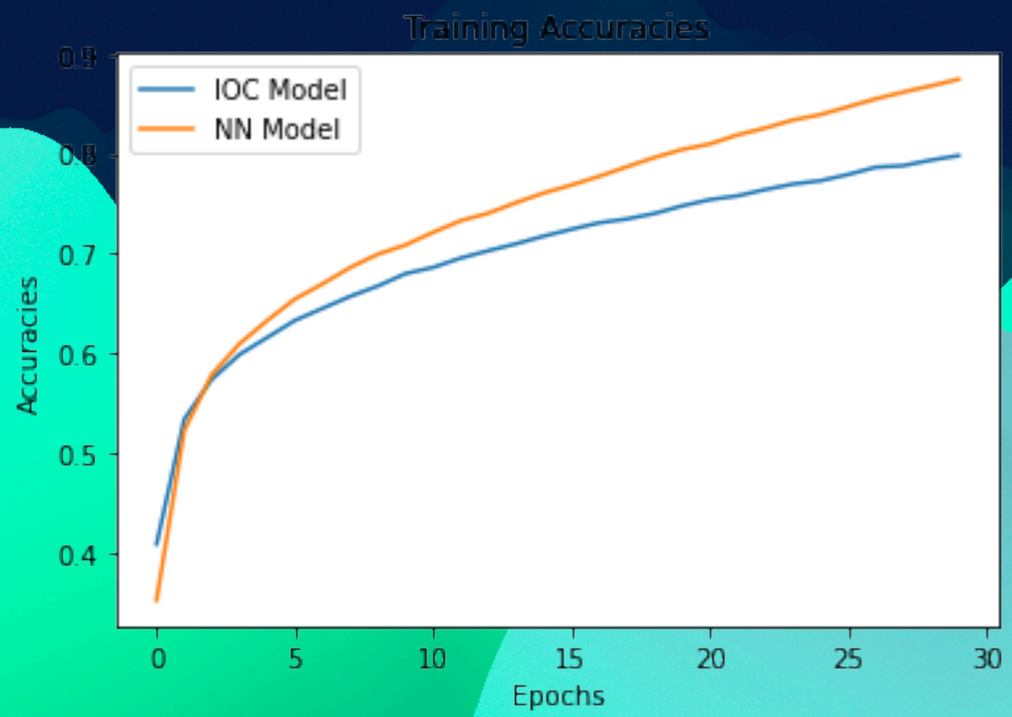
Architecture

MLP and IOC-MLP

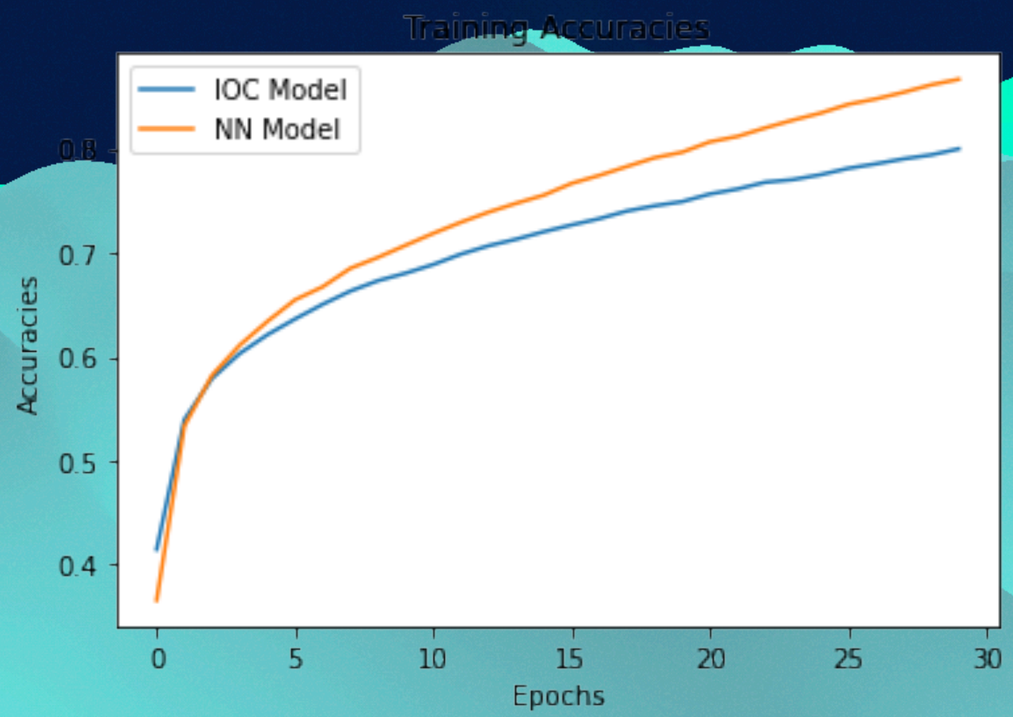
- 3 Hidden Layers (800 Neurones each)
- ReLU and ELU are used as activations for NN and IOC respectively.
- Batch Normalisation is used between every layer.
- Softmax activation is used in the last layer.
- Optimiser - Adam, Learning rate = 10^{-4}

Results and Conclusion

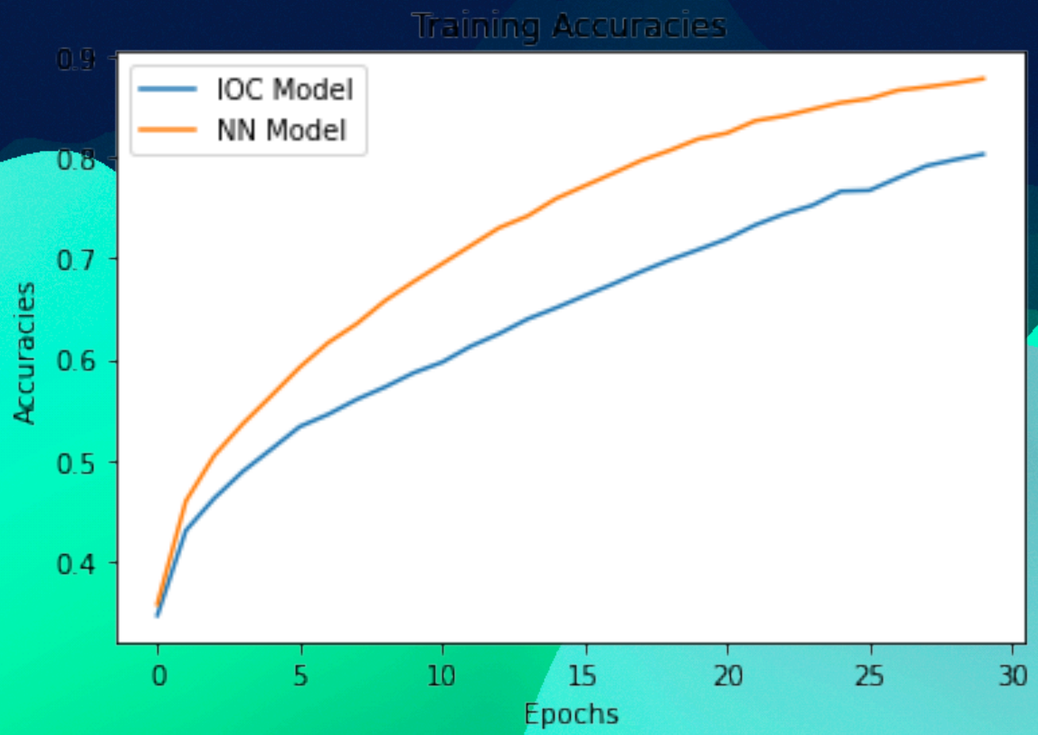
AllConv - CIFAR10



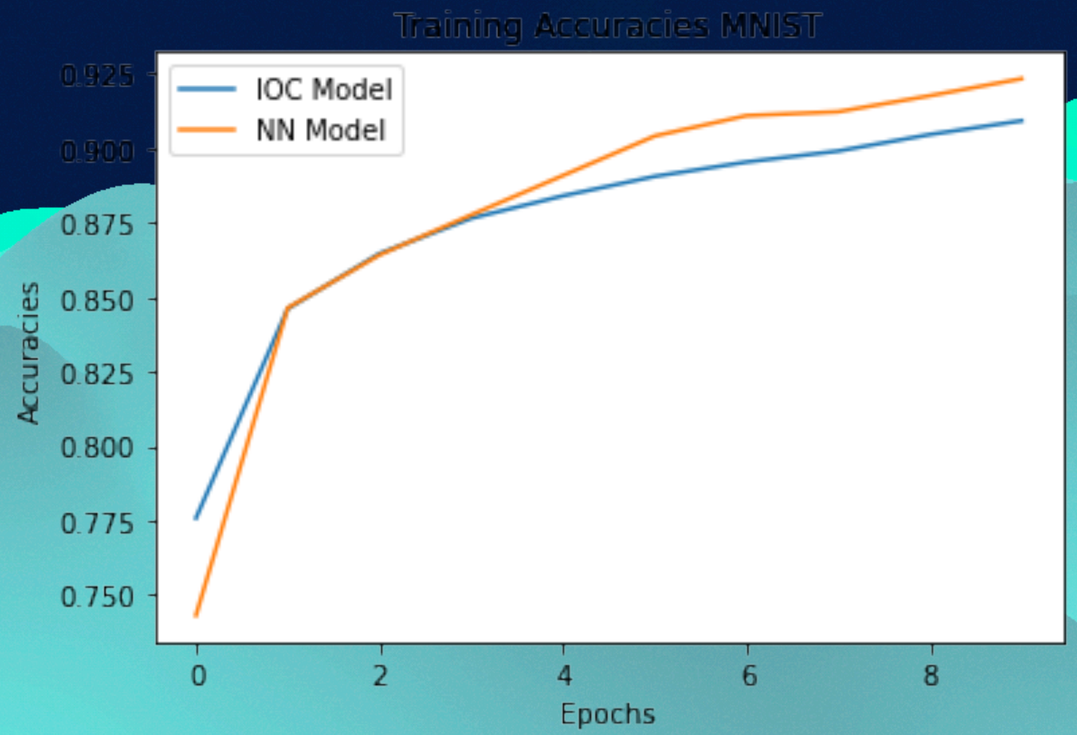
Randomised Labelling - AllConv - CIFAR10



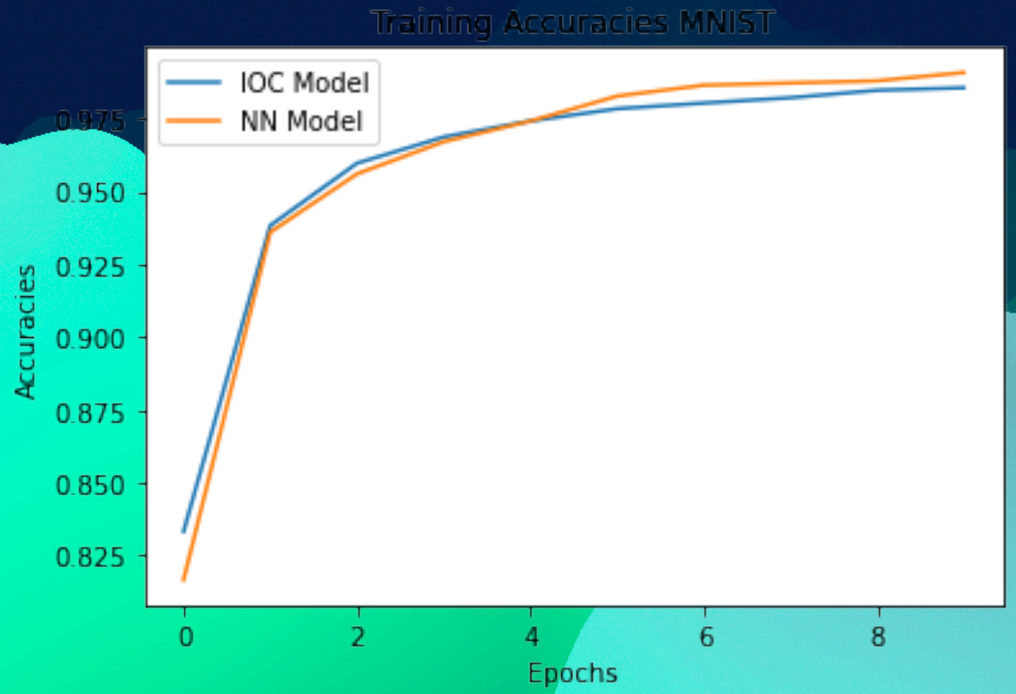
CIFAR10 - MLP



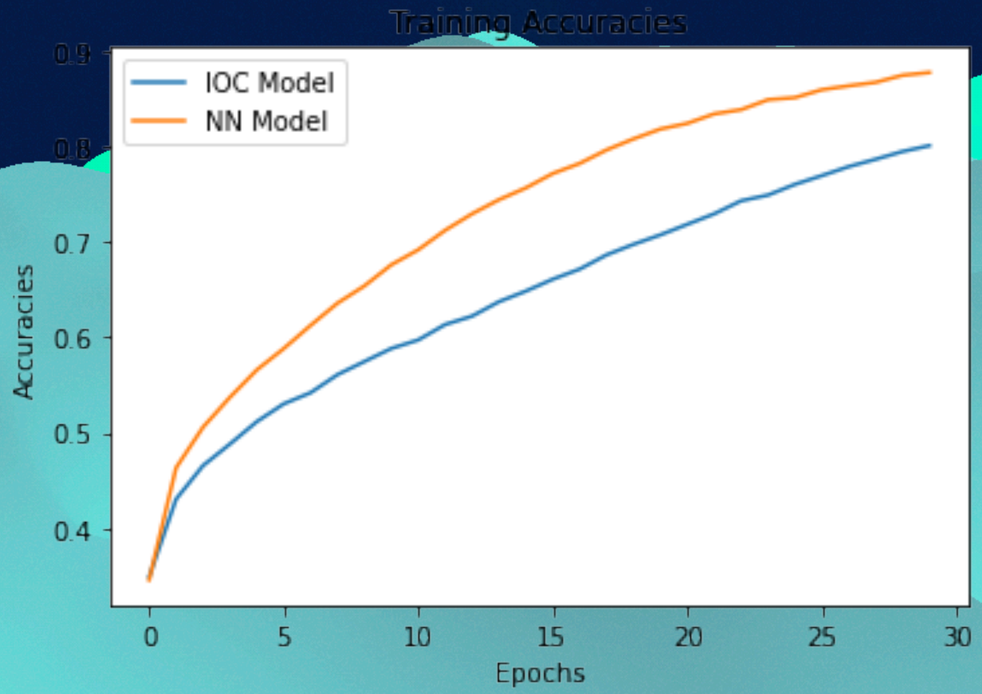
FMNIST



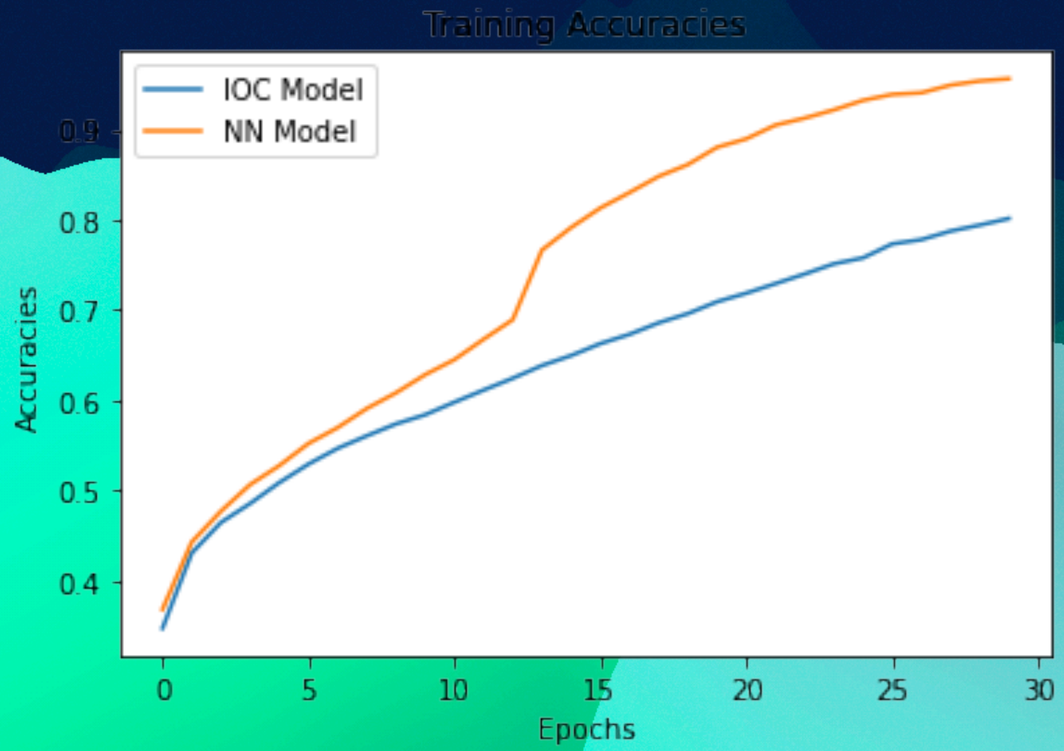
MNIST



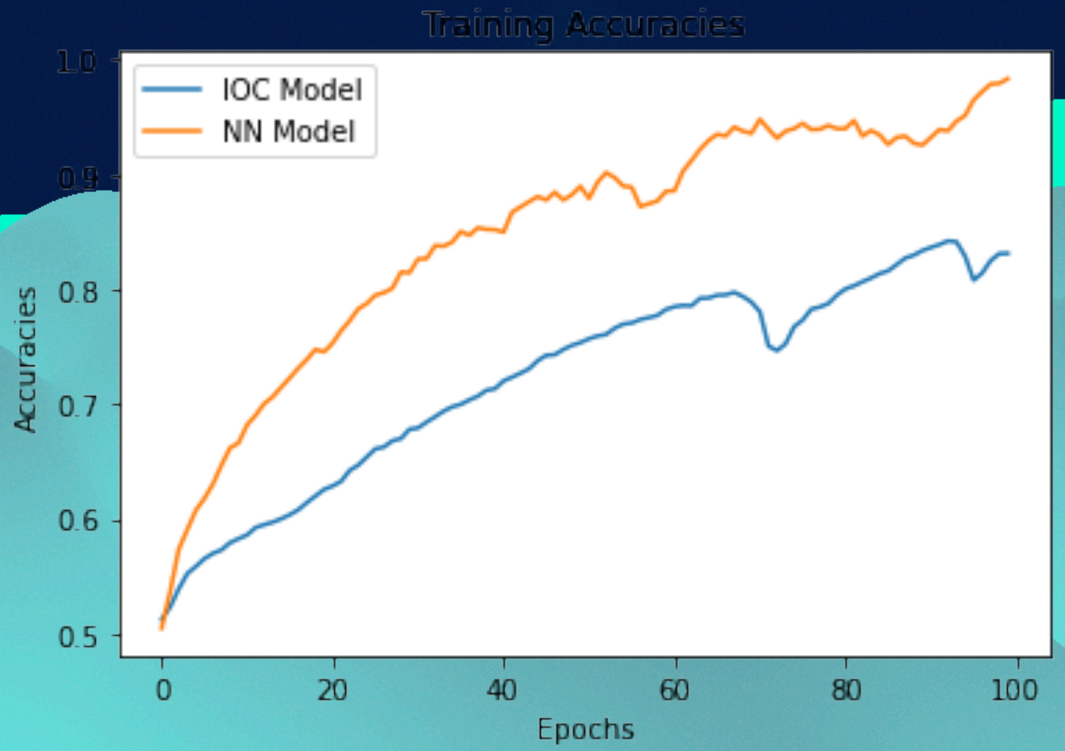
Randomised Labelling - MLP - CIFAR10



CIFAIR10



Spambase



Results

Nitty Gritty Details

dataset\architecture	MLP	AllConv	IOC-MLP	IOC-AllConv
MNIST	1.34%	-	0.76%	-
FMNIST	4.6%	-	3.04%	-
CIFAR-10 Random Noise Labelling	37.1%	18.43%	28.5%	12.76%
CIFAR-10	40.9%	19.32%	28.2%	12.01%
Spambase Random Labelling	46.33%	-	32.45%	-

Thank You