Neural Networks for Optics Pattern Recognition

Ajay Mehra
Department of Physics
University of Delhi
Delhi, India
ajaymehra@duck.com

***Abstract*—This document is a model and instructions for LaTeX. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract. Often only 100 to 300 words, the abstract generally provides a broad overview and is never more than a page. It describes the essence, the main theme of the paper. It includes the research question posed, its significance, the methodology, and the main results or findings. Remember to take great care in composing the abstract. It's the first part of the paper the instructor reads. It must impress with a strong content, good style, and general aesthetic appeal.**

***Index Terms*—Neural networks, Convolution, Optics pattern, Deep Learning**

I. Introduction

Deep learning is a subset of machine learning, which is essentially a neural network with some layers. These neural networks attempt to simulate the behavior of the human brain, and make predictions from data. In this paper, the implementation of neural networks in case of Hall C data of *Jefferson lab* is summarised.

A neural network is a system that learns how to make predictions by following these steps:

1) Taking the input data
2) Making a prediction
3) Comparing the prediction to the desired output
4) Adjusting its internal state to predict correctly the next time

A neural network comprises of layers, which helps in feature extraction. Concept of weights and biases are used in building these layers, the aim is to change and tune these weights and biases such to minimise the prediction error. Each neuron takes inputs and have some values of weights and biases, and it gives some output.

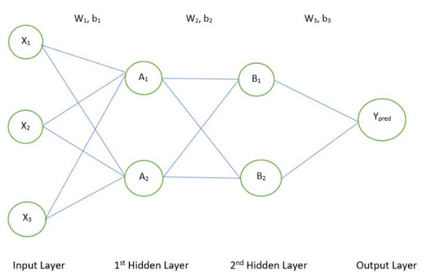

Fig. 1. Neural network [1]

II. Methodology

In this research, we analyze a total of 185 distinct simulated optics patterns from the *Super High Momentum Spectrometer* (SHMS) at Hall C of Jefferson Lab. There were six different optics correlations x_{fp} vs. y_{fp} , x_{fp} vs. $y^{'}_{fp}$, x_{fp} vs. $x^{'}_{fp}$, $x^{'}_{fp}$ vs. y_{fp} , $x^{'}_{fp}$ vs. $y^{'}_{fp}$, $y^{'}_{fp}$ vs. y_{fp} each pattern had 31 optics images with varying optics tunes [Q1, Q2, Q3], corresponding to the spectrometer quadrupole magnets, here summarized in Table I:

Quadrupole Magnet	Range	Stepsize
$Q1$	[0.945, 1.055]	0.01
$Q2$	[0.945, 1.055]	0.01
$Q3$	[0.945, 1.055]	0.01

TABLE I
Quadrupoles Input Data Details

Each of the six 2D SHMS optics pattern correlations were trained separately, using 31 different optics tunes per correlation plot for a total of 185 images. The optics patters for testing the network consisted of only varying Q2 from 0.945 to 1.055 in steps of 0.01, while keeping Q1 and Q3 tunes fixed at unity. To test the neural network after it had been trained, a set of 10 images were used for each 2D optics correlation, where Q1 and Q3 tunes were kept fixed at unity while Q2 was varied from 0.955 to 1.055 in steps of 0.01 for a total of 10 Q2 tunes. [2] Private communication. C. Yero. August 2021.

III. Data Analysis Procedure

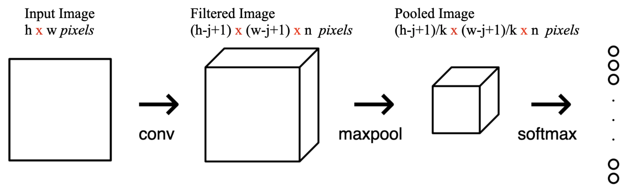

Fig. 2. CNN layers

The Neural Network used in this research consists of 5 layers in total (See Fig.2). The input and output layers, represent the raw input image and output model prediction, respectively, and intermediate hidden layers, *convolutional*, *pooling*, and *activation* layers, each with a specific image analysis task as described in the subsections below. [3]

A. Convolutional Layer

The convolutional layer is the core building block of a CNN, and it is where the majority of computation occurs. It requires a few components, which are input data, a filter, and a feature map. The feature map will move across the receptive fields of the image, checking if the feature is present. This process is known as a convolution.

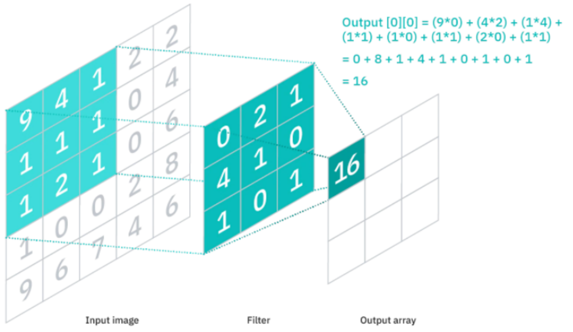
Output [0][0] = (9*0) + (4*2) + (1*4) + (1*1) + (1*0) + (1*1) + (2*0) + (1*1)
= 0 + 8 + 1 + 4 + 1 + 0 + 1 + 0 + 1
= 16
Input image
Filter
Output array

Fig. 3. Convolution Demo [4]

The feature detector is a two-dimensional *(2-D)* array of weights, which represents part of the image. While they can vary in size, In our case, we are using 12 filters of dimension *6x6*. The filter is then applied to an area of the image, and a dot product is calculated between the input pixels and the filter. This dot product is then fed into an output array. Afterwards, the filter shifts by a stride which is 1 in our case, repeating the process until the kernel has swept across the entire image. The final output from the series of dot products from the input and the filter is known as a feature map, activation map, or a convolved feature.

- Number of filters: 12
- Filter size: 6x6
- Stride: 1

Example Figure details Fig.4: 28x28 image is converted to 26x26 to 24x24, with a filter of dimension 3x3 and stride of 1.

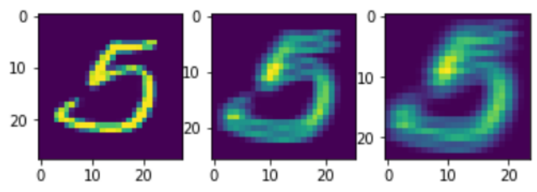

Fig. 4. Convolution [5]

B. Pooling Layer

Pooling layers are used for dimensionality reduction, to reduce number of parameters. It is similar to convolutional layer, except its output is determined by some aggregation function.

In our case, we used *5x5* MaxPooling, in which filter (5x5) moves across the input, and returns the pixel with largest value to output array.

Although, in pooling layers, lot of information is lost, but it reduces complexity and solves the chances of overfitting.

Example Figure details Fig.5: 28x28 is converted to 14x14 to 7x7, with 2x2 maxpooling.

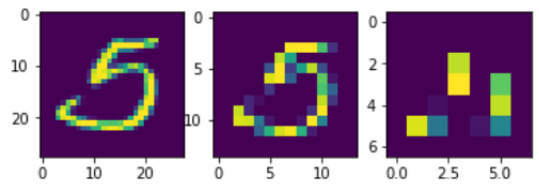

Fig. 5. Maxpooling [5]

C. Activation Layer

This layer aims to classify the images on basis of its features of previous layer. In this layer, all the previous layer nodes connect directly to output nodes. In our case we used *Softmax Layer* [6], which uses Softmax function for its classification.

We can see in Fig 6, first layer shows outputs from Maxpooling, and it is connected to second layer via a neural network connection, which has certain number of classes, number of classes depends on the number of outputs we want, and then begins the softmax transformation.

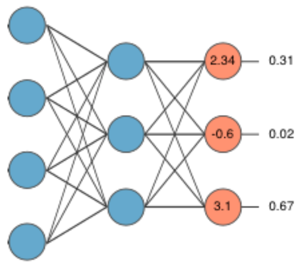

Fig. 6. Softmax [7]

Given a set of numbers, softmax function converts them into probabilities. Softmax performs the following transform on n numbers $x_1, \ldots, x_n$

$$S(x_n) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{1}$$

The reason for converting them into probability spectrum is to consider relative chances of being predicted correct, and how much our class' prediction is true than other classes, which helps in training our model. The output of these layers are always less than 1 and sums up to 1.

Cross entropy loss: It is calculated by taking the logarithm of correct class' probability.

$$L = -\ln(p_c) \tag{2}$$

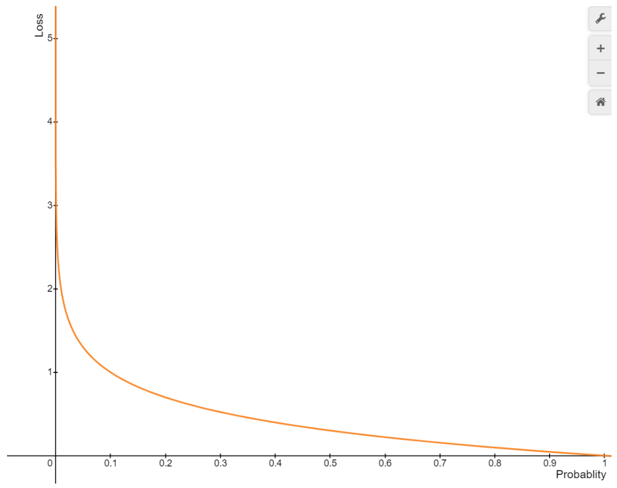

Fig. 7. Probability Loss graph [8]

If probability of correct class is close to one (means our prediction is better) then the loss is close to zero, but if the probability is close to 0 then penalty will be high and loss will approach infinity.

Backpropagation [9]: In forward phase, the inputs of one layer were passed to next layer and completely through the network, whereas in backward phase gradients are backpropagated through the network and weights are updated.

During forward phase, network will store the data like inputs, input size, intermediate values, etc, and Each layer will receive a gradient of Loss with respect to output. ($\frac{\partial L}{\partial out}$), and will give the gradient of Loss with respect to input ($\frac{\partial L}{\partial in}$). In our case, we are using loss as negative of logarithm of probability of correct class.

Next step was to update the values of weights and biases, we used the stochastic gradient descent method. In SGD method, all the variables are updated using following equation,

$$x \longleftarrow x - \alpha \frac{\partial L}{\partial x} \tag{3}$$

α is learning rate, It depends on this constant how fast or slow will our loss converge to zero. Overall, x will converge to that value which will result into minimum loss.

Figure 8 illustrates the transformations on our image, From 200x200x12 to 195x195x12 to 49x49x12 to 1x10.

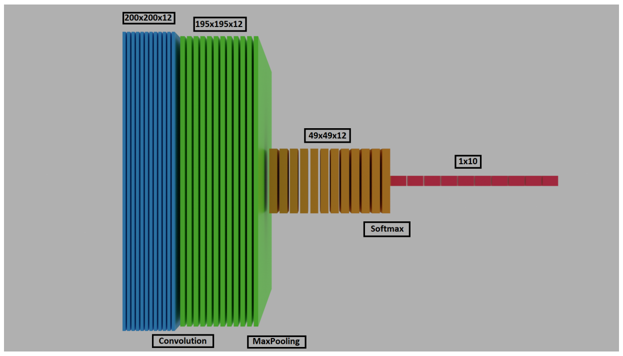

Fig. 8. Layers in our network [10]

IV. Results and Discussion

The purpose of this research was to teach a machine to recognize optics patterns that would otherwise be difficult to distinguish by the "human eye". With the help of Keras API, we were able to train and test a CNN by providing simulated optics data from Jefferson Lab, Hall C. Each of the six 2D optics correlation was trained with 31 optics tunes, and each were able to reach and plateau at an accuracy of 100 %, and a average loss of 0.3787 , in 100 epochs of training. We used 10 test images per each of the six 2D optics correlations, and network was able to correctly predict each of the patterns with average of 80% accuracy. The results of the training are shown in Fig.9 and Fig.10

- *Result: Accuracy vs Epochs*: In training graph 9, we can see that till 100 epochs all the correlations have achieved 100 % accuracy.
- *Result: Loss vs Epochs*: Similarly, graph 10 tells us that loss is approaching 0.

The results of the test images is summarized in Table II

2D optics correlation	Correct predicted patterns	No. patterns	Accuracy
x_{fp} vs. x'_{fp}	8	10	0.8
x_{fp} vs. y_{fp}	8	10	0.8
x_{fp} vs. y'_{fp}	10	10	1.0
x'_{fp} vs. y_{fp}	5	10	0.5
x'_{fp} vs. y'_{fp}	10	10	1.0
y'_{fp} vs. y_{fp}	7	10	0.7

TABLE II
Caption of table.

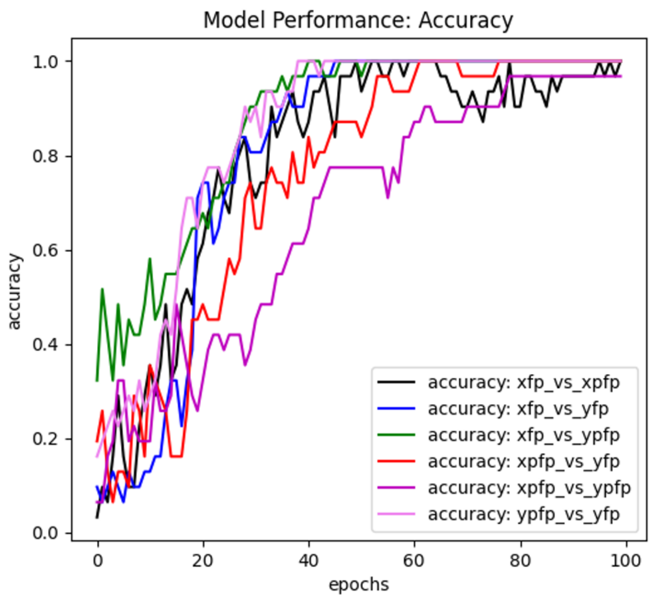

Fig. 9. Accuracy vs epochs plot

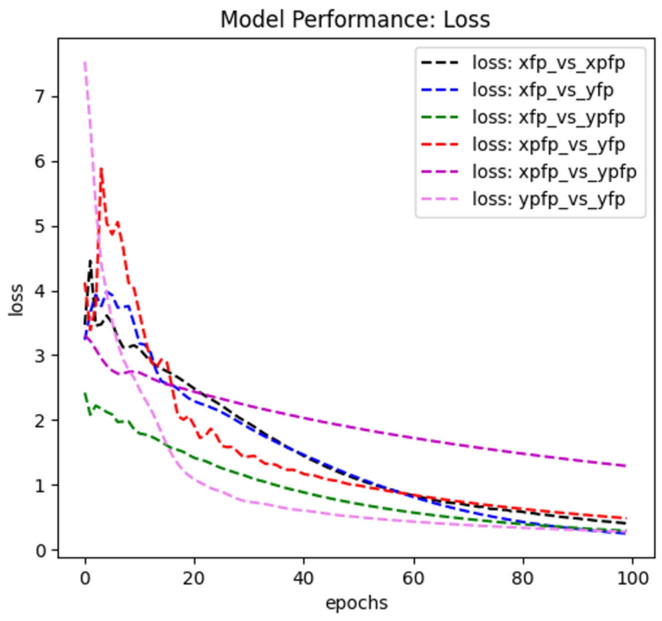

Fig. 10. Loss vs epochs plot

References

[1] T. Bhattacharya. (2020) An Introduction to Neural Networks with Implementation from Scratch in Python. (Accessed on 2021-09-15). [Online]. Available: https://towardsdatascience.com/an-introduction-to-neural-networks-with-implementation-from-scratch-using-python-da4b6a45c05b

[2] C. Yero, Private communication, August 2021.

[3] V. Zhou. (2019) CNNs, Part 1: An Introduction to Convolutional Neural Networks. (Accessed on 2021-08-20). [Online]. Available: https://victorzhou.com/blog/intro-to-cnns-part-1/

[4] I. C. Educaion. (2020) Convolutional Neural Networks. (Accessed on 2021-09-12). [Online]. Available: https://www.ibm.com/cloud/learn/convolutional-neural-networks

[5] Y. LeCun. MNIST Images. (Accessed on 2021-09-25). [Online]. Available: http://yann.lecun.com/exdb/mnist/

[6] V. Zhou. (2019) A Simple Explanation of the Softmax Function. (Accessed on 2021-08-27). [Online]. Available: https://victorzhou.com/blog/softmax/

[7] Z. Singer. (2019) Softmax and Uncertainty. (Accessed on 2021-09-12). [Online]. Available: https://towardsdatascience.com/softmax-and-uncertainty-c8450ea7e064

[8] A. Mehra, Used Desmos graphing Calculator, October 2021. [Online]. Available: https://www.desmos.com/calculator

[9] V. Zhou. (2020) CNNs, Part 2: Training a Convolutional Neural Network. (Accessed on 2021-08-27). [Online]. Available: https://victorzhou.com/blog/intro-to-cnns-part-2/

[10] A. Mehra, Used Blender Software, October 2021. [Online]. Available: https://blender.org

[11] A. Mehra. Result images. [Online]. Available: https://github.com/AJRocks321/Temp/tree/main/Result

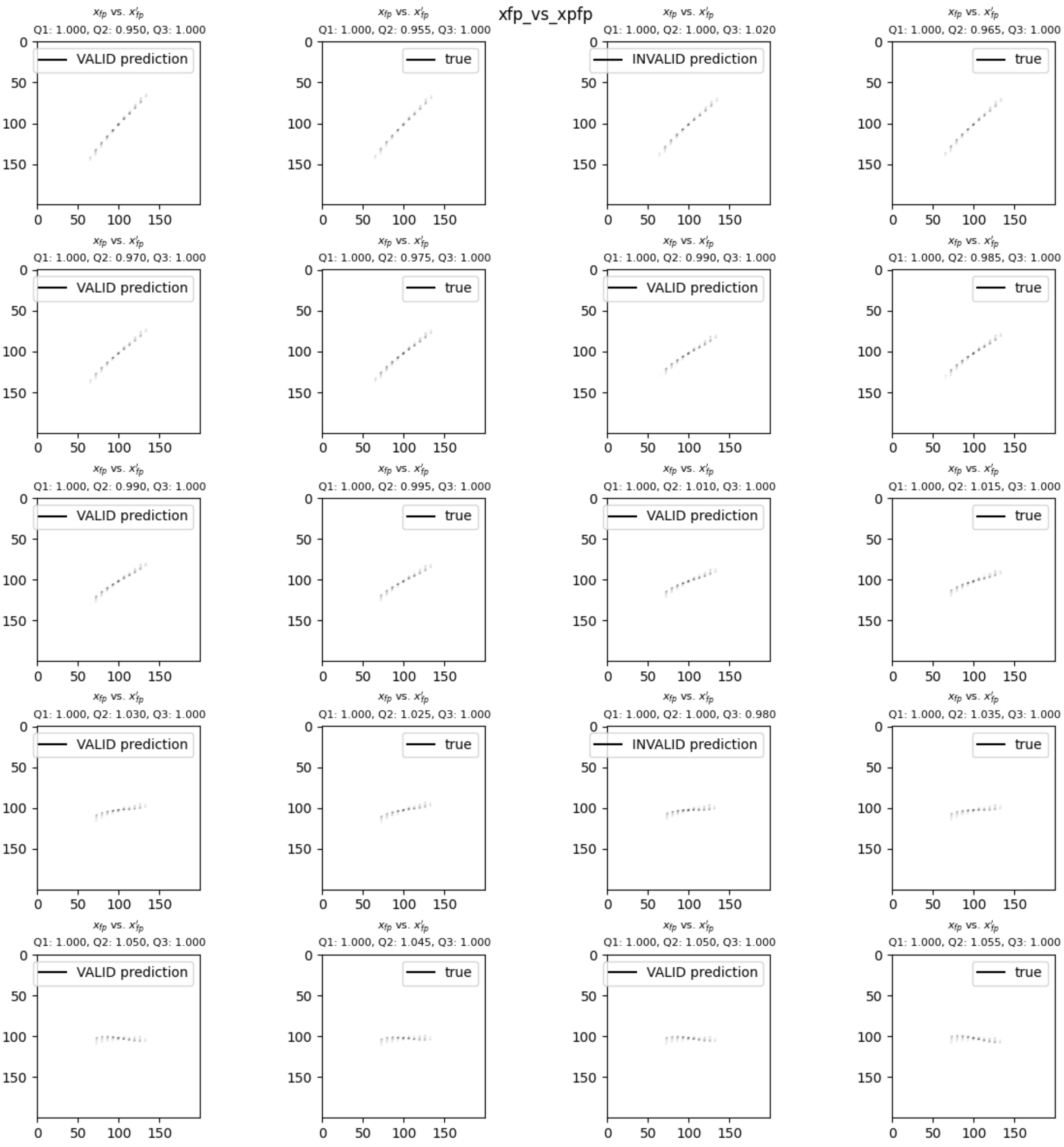

Fig. 11. xfp vs xpfp: A sample result: clearly we can see that it cannot be distinguished by naked eye. More images can be found in [11]