Convolutional Neural Networks for Super High Momentum Spectrometer Optics Pattern Recognition

Zain Mothupi
Department of Physics
University of Warwick
Coventry, England
mothupizain@gmail.com

***Abstract*—Our research focused on developing algorithms for the recognition and prediction of specific tune patterns in spectrometer optics data from Experimantal Hall C at Jefferson Laboratory. The main goal was to create a machine learning model capable of recognizing optics patterns that would otherwise be tedious for humans to classify, in a mere matter of seconds with reasonably high accuracy. Specifically, we utilised the Keras deep learning Application Programming Interface(API) to build a Convolutional Neural Network(CNN). The CNN was trained on a dataset of 186 simulated optics patterns and a Cross-Entropy function was implemented to assess the model's accuracy throughout the training process. The model was able to reach an average accuracy close to 100% and an average loss of approximately 0.3. The machine learning model's ability to predict output was then tested against a set of 60 new images and achieved an average accuracy of 83%.**

I. Introduction

Since the inception of Artificial Intelligence in the mid 20th century, there has been rapid progress towards the creation of systems capable of matching or even surpassing human ability. As a result, Machine Learning (ML), a subset of artificial intelligence, emerged. In ML computer systems learn to perform tasks without explicit instructions as they learn purely from a set of data. The concept of computers learning directly from data has led to many algorithms modelled after the way humans learn. In particular, Neural Networks were inspired by the wiring of the human brain, imitating the interconnectedness of the neurons. See Fig.1

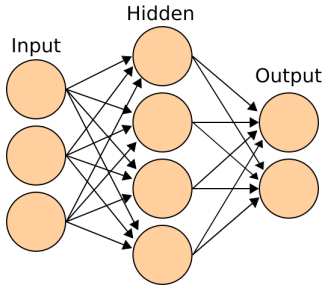

Fig. 1. Example of an Artificial Neural Network.Reprinted from Ref. [1].

In our research, we used a specific type of Neural Network known as a Convolutional Neural Network to interpret and analyse images. All CNNs follow a basic list of steps in order to learn patterns from data:

- receive input data
- make a prediction
- assess the prediction by comparing it to the desired output
- adjust its internal structure to give more accurate predictions the next time

A simple illustration of a neural network architecture is shown in Fig.2

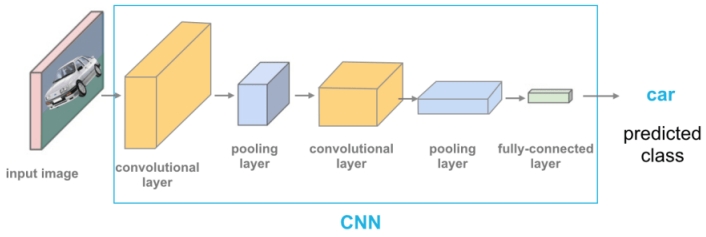

Fig. 2. Illustration of the structure of a Convolutional Neural Network. Reprinted from Ref. [2].

II. Methodology

In this research, we analyze a total of 186 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_{pfp} vs. y_{pfp}, x_{fp} vs. y_{pfp}, x_{pfp} vs. y_{fp}, x_{fp} vs. x_{pfp} and y_{pfp} vs. y_{fp} . Here the "fp" refers to the focal plane, which is an imaginary plane in the spectrometer where the transported particles are focused to and the first "p" refers to the prime symbol which is a derivative of the focal plane variable with respect to the optical z-axis along which particles are transported. Each pattern had 31 optics images with varying optics tunes [Q1, Q2, Q3], corresponding to the spectrometer quadrupole magnets, summarized in Table I:

Quadrupole Magnet	Range	Stepsize
$Q1$	[0.90, 1.10]	0.02
$Q2$	[0.95, 1.05]	0.01
$Q3$	[0.90, 1.10]	0.02

TABLE I: Summary of the variance of Optics tunes during the training process.

Each of the six 2D SHMS optics pattern correlations were trained separately, using 31 different optics tunes per correlation plot for a total of 186 images. The optics patterns 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.
Figure 3 shows an example of a 2D Optics Correlation Plot.

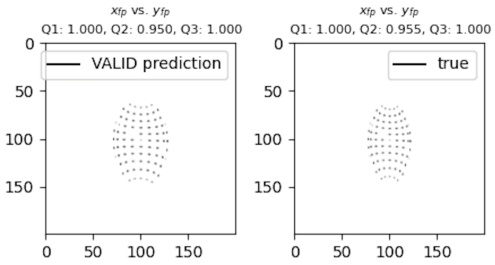

Fig. 3. Example of a 2D Optics Correlation with the Predicted(left) and True(right) Optics Patterns.

The data with specific [Q1,Q2,Q3] tunes were simulated using the standard Hall C simulation program (`mc-single-arm`) and the raw data output was written to a ROOTfile. A separate ROOT C++ script (`make_2Doptics.C`) was used to form each of the six abovementioned 2D focal plane correlations correlations which were stored in a separate ROOTfile as histogram objects. The 2D histograms were then converted to a 2D pixelated array and stored in binary format (.h5) via a Python code (`save2binary.py`) array to be read by the Neural Network using Python Keras. Each optics image used was 200x200 pixels and was passed through each of the hidden layers of the network described in Section III of this article.

III. Data Analysis Procedure

Our Neural Network utilised 5 layers in total(See Fig.4). It consisted of input and output layers which represent the raw input image and output prediction and intermediate hidden layers each with a specific image analysis task. The intermediate layers are the *Convolutional layer, Pooling layer* and *Activation layer* described in the subsections below.

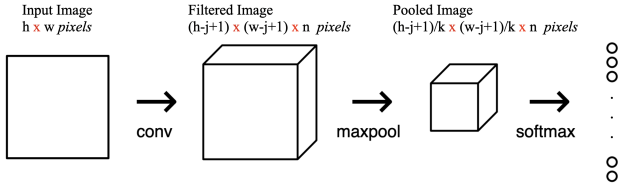

Fig. 4. Structure of the Convolutional Neural Network used in this analysis.

A. Convolutional Layer

The convolutional layer contains a set of 12 filters with dimension 6x6, which are convolved with the input image and outputted by the layer. The process of convolving the image with the filters consists of overlaying the filter over a group of pixels (with the same dimensions as the filter), carrying out element-wise multiplication, calculating the sum of the results and using this as a pixel value in the output image. We then move on to the next possible grouping of pixels and continue the process until the entire image has been convolved with the filter. Convolutional layers help detect specific localized features in images which aids the model in making more accurate predictions (see Ref. [3]).

B. Pooling Layer

We used a pooling size of 6 which decreases the size of our input images from 200x200 to 32x32 pixels. This allows for the model to reduces the number of computations it has to perform while still preserving the important elements of the image. This is achieved by "pooling" or grouping pixel values together and selecting the maximum of the values which acts as a summary of the features present in the region (see Ref. [3]).

C. Activation Layer

The activation layer uses a function known as an activation function to transform an unbounded input to a bounded output. In this research we used the Softmax function as our activation function. The Softmax function is given by $\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$ and turns the neural network's output values into probabilities which are more useful when evaluating our models predictions as it allows us to assess the certainty with which the NN makes a prediction. We assess the performance of our NN using a loss function. The loss function compares the prediction of the NN to the actual value and calculates how far off the model is. This is then feed back to NN so that it can adjust its weights and biases to achieve a lower loss or higher accuracy (See Ref. [4]).

D. Backpropagation

The input image is passed through all the layers and the final output of the Softmax layer is a prediction. The prediction is then compared to the known result and the loss is calculated. A backpropagation method is used to minimize the loss by updating the parameters of the NN so as to decrease the loss. Each combination of a forward runthrough of the NN and updating of parameters via backproagation is known as an

epoch. The updated parameters are then analysed through the same process for a certain number of epochs so that the loss is minimized.

IV. Results and Discussion

In this research our aim was to teach a computer to recognize optics patterns that would otherwise be tedious or difficult to distinguish for humans. Using the Keras API we trained and tested a CNN by providing simulated optics data from Jefferson Laboratory, Hall C. Each of the six 2D optics correlations were trained with 31 optics tunes and were able to reach and plateau at an accuracy of 98% and a loss of 0.3 in 100 epochs. We used 10 test images per optics correlation and the network was able to correctly predict the patterns with at least 83% accuracy. The results of the training are shown in Fig.5 and Fig.6.

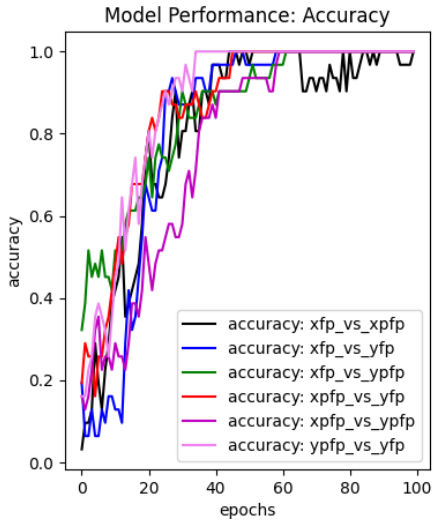

Fig. 5. Plot of Accuracy vs Number of Epochs

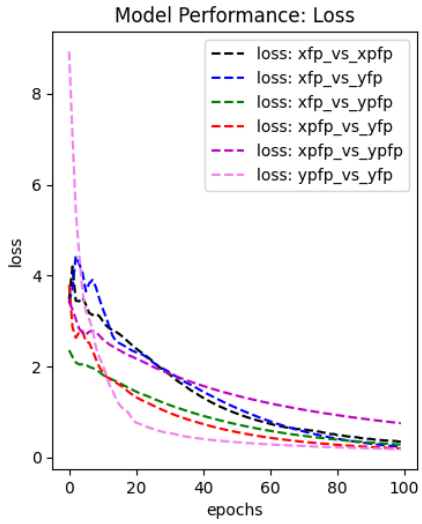

Fig. 6. Plot of Loss vs number of Epochs

The results of the test images is summarized in Table II where N_c represents the number of correctly predicted patterns and N_t represents the total number of patterns.

2D Optics Correlation	N_c	N_t	Accuracy
x_{fp} vs. y_{fp}	9	10	90%
x_{pfp} vs. y_{pfp}	10	10	100%
x_{fp} vs. y_{pfp}	10	10	100%
x_{pfp} vs. y_{fp}	5	10	50%
x_{fp} vs. x_{pfp}	9	10	90%
y_{pfp} vs. y_{fp}	7	10	70%

TABLE II: Performance of the trained CNN for each of the 2D optics correlation patterns.

These results show that the computer can achieve a reasonably high accuracy and low loss during training. The test data shows us that when given new data the model still outputs reliable predictions for the majority of optics correlations. However, from these results we can see that the model had trouble predicting the correct labels for the x_{pfp} vs. y_{fp} and y_{pfp} vs. y_{fp} optics correlations. To improve the predictions we could decrease the step size between the optics tunes so that the neural network can learn to distinguish more subtle features in the data which will allow it to make more accurate predictions. Altogether we have demonstrated that the model can recognize these optics patterns fairly well and with a few adjustments could be used to support humans in greatly reducing the time taken to classify these optics tune patterns.

References

[1] C. M. Burnett. (2006) Illustration of the topology of a generic Artificial Neural Network (ANN). (Accessed on 2021-09-20). [Online]. Available: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg

[2] C. Camacho. (2018) Convolutional Neural Networks. (Accessed on 2021-09-20). [Online]. Available: https://cezannec.github.io/Convolutional_Neural_Networks/

[3] V. Zhou. (2019) CNNs, Part 1: An Introduction to Convolutional Neural Networks. (Accessed on 2012-03-20). [Online]. Available: https://victorzhou.com/blog/intro-to-cnns-part-1/

[4] V. Zhou. (2019) A Simple Explanation of the Softmax Function. (Accessed on 2012-03-20). [Online]. Available: https://victorzhou.com/blog/softmax/