Can a Fully Connected Convolutional NeuralNet perceive an "Optimal" Hamilton Cycle of a Graph In Spatial Domain?

Peeyush Kumar
Boston University
peeyush@bu.edu

ABSTRACT

In this new wave of Neural Networks, deep learning is being used in various industries as a means of problem solving. The area of Optimizations is one of the main fields in which neural networks have started being deployed. In this paper one such problem of Optimization has been targeted, which is 'Traveling Salesman Problem'. The key part in optimization through neural nets is to convert the optimization problem to a representation suitable for deep learning. Given the fact that Convolutional Neural Nets are good at image processing, therefore the TSP problem has been mapped to a spatial domain so that CNNs perceive the problem better. A fully convolutional network (FCN) is used to solve graphs for an optimal Hamilton cycle. The training process is completely analyzed and has been interpreted through stages. The Synthetic Dataset has been generated for this CNN application. Through this application of CNN, the TSP problem can now be considered as a polynomial type problem, rather than NP hard Problem. At the end of the experiment the obtained results show acceptable performance in finding the optimal Hamilton cycle.

1. INTRODUCTION

In today's world, via back-propagation, the neural net algorithms can learn and extract knowledge from the given data. In other words, they can map the training example to a training label efficiently. For example, in order to develop a car detection algorithm, we only have to give neural nets the images of cars and tell them if it's a car or not, then through back propagation the network learns to map input images to if it's a car or not. The network does this via capturing spatial information available to the network in different image representations.

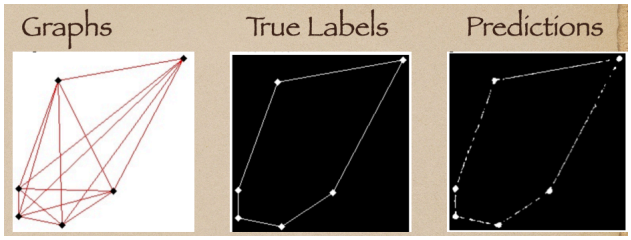


Figure 1. Instance of Input image, True Label and the output by the network.

In this project the advantage of the ability of CNN's to map input images to output labels has been taken. The network was given the images in which the graphs had been mapped and the label

images consisting of the optimal hamiltonian cycle. Through optimization algorithms the neural net learns to efficiently solve the given optimization problem. Large numbers of iterations were needed to obtain the optimal solution for the TSP problem.

The images segmentation application of the CNN's has been exploited in this project. In this application, we teach the network to do the pixel wise prediction for the image. FCN or Fully Connected Neural Network (VGG16 version of the FCN) has been chosen for this project. The network takes an image of size 224x224x3 and outputs the label map of size 224x224x2. The Figure 1 shows the input image (in which the graph has been mapped to), the true label map (showing the Optimal Hamiltonian Path) and the pixel wise prediction of the image done by the VGG16 inspired version of FCN.

2. APPROACH

This section describes the complete process followed in this entire project, along with the structure and process used to generate the dataset and the training process used in the project.

2.1 DATA GENERATION

In this project, the data fed to the FCN used was generated synthetically. A specified number of graphs were randomly generated and were mapped to the image using the below equation:

$$\left\{\begin{array}{c} x_{max} = x_{max} + s_f,\ x_{max} = x_{min} - s_f \\ y_{max} = y_{max} + s_f,\ y_{max} = y_{min} - s_f \\ \lambda_x = \frac{x_{max} - x_{min}}{w},\ \lambda_y = \frac{y_{max} - y_{min}}{w} \\ x_i^* = \frac{x_i - x_{min}}{\lambda_x + \epsilon},\ y_i^* = \frac{y_i - y_{min}}{\lambda_y + \epsilon} \\ i = 1,2,3,...,n \end{array}\right\} \quad \text{Eq 1}$$

Where, x_{max}, y_{max} = Maximum x and y coordinate among all graph nodes ; x_{min}, y_{min} = Minimum x and y coordinate among all graph nodes ; s_f = Special Factor ; $\lambda_x \; and \; \lambda_y$ = Scaling factor for x and y coordinates; $x_i^* \; and \; y_i^*$ = Scaled version of points x and y.

The equations 1 was used to generate n graph images of size 224x224x3 and also their respective true labels maps of size 224x224. In this project the value of $n = 16{,}000$. (But the code was written to generate infinite number of graphs)

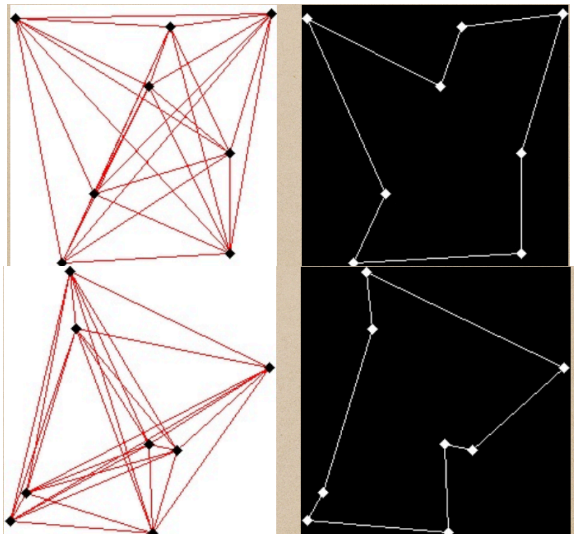


Figure 2. Example of Mapped representation of graphs and their respective label maps. The nodes in the figure re circles of radius 6.

The figure 2. shows the example of Mapped representation of graphs (on left side) and there respective label maps (on right side). The solution of the graphs were obtained using brute force technique.

2.2 ARCHITECTURE OF FCN

receptive field, deconvolution of the feature map alone will result in insufficient detail. In contrast, the feature maps of the previous pooling layers have more detailed information; but generalized features are rarely found in the feature maps. Therefore, these feature maps can complement each other. To this end, the feature maps output by

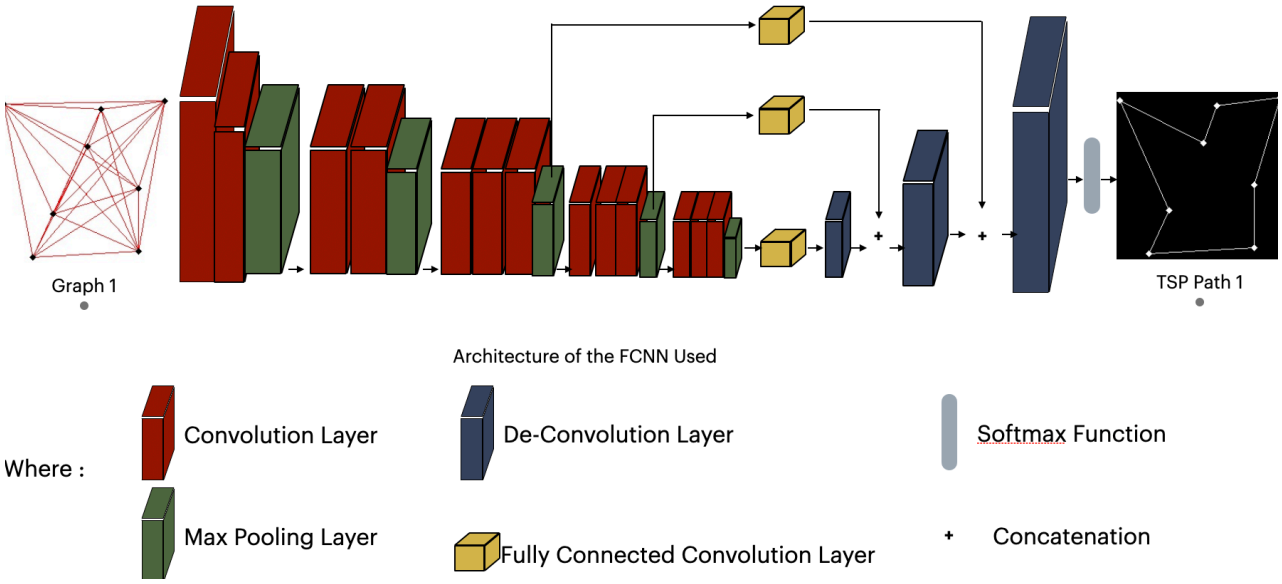


Figure 3. The Architecture of the FCN

The figure 3. Shows the complete Architecture of the network used. A FCN retains the first five convolutions and pooling operations of the VGG structure. The 7 size of the input image will be reduced by 2, 4, 8, and16 times in the five convolution and pooling operations; it eventually becomes a feature map with the size of 7×7×1024. In contrast to a VGG structure, the FCN structure uses a deconvolution layer to make the output image of the network the same size as the input image, thus the feature map is required to be up sampled by 32 times. Since the pixels in the 7×7×1024 feature map each have a large

the fourth and third pooling layers are deconvolve simultaneously with 16 times and 8 times up-sampling, respectively.

As the network outputs an image of size 224x224x2, in order to perform pixel wise prediction, so the true label maps were one hot encoded before training.

2.3 Training Process

Due to pixel wise prediction of the image map, the training process took a long time to train. The network was trained for 2000 epochs with batch

size of 100 on Nvidia SMI GPU. The GPU information is given in Figure 4.

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 495.44       Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   34C    P0    26W / 250W |      0MiB / 16280MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
```

Figure 4. The Specifications of the GPU Used for Training.

Instead of using normal training process, as Truncated Training was used. The network was trained for 4 times for 500 epochs each and the first training for 1500 epochs was done using ADAM optimization (with learning rate 0.001) and last training for 500 epoch was done using SDG optimizer (with learning rate 0.001). For both Scenarios the Learning was Decayed by the factor of 0 after every 100the epoch. The minimization of the loss function 'Categorical cross entry ' given in the equation below is show with the help of Loss vs Epoch Trade foo Graph in Figure 5 :

$$\sum_{i}^{w}\sum_{j}^{h}\sum_{k}^{2} y_{ijk}^{'} log(y_{ijk})/(2wh) \qquad \text{Eq 2}$$

Where y is the probability predicted by the network as output and y' is the true probability of the label.

The RELU Activation function was used for every Convolutional layer and Fully Connected layer (which are just convolutponals layer with 1x1

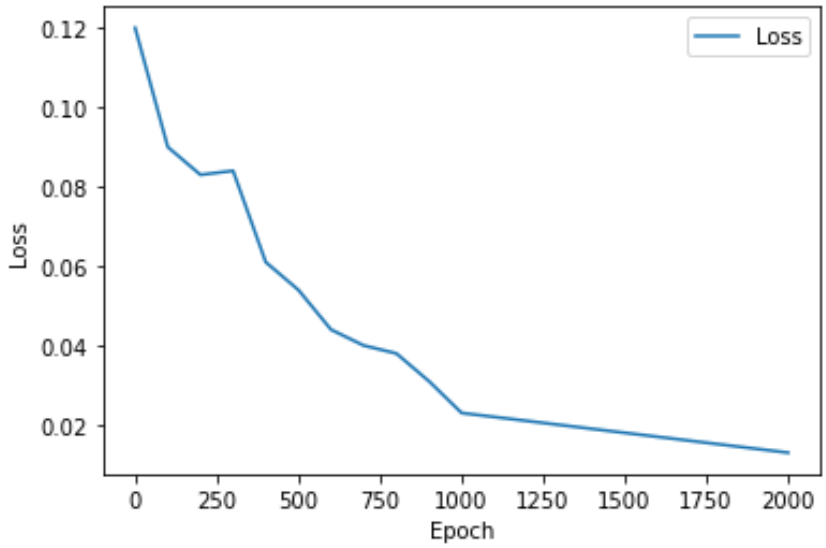


Figure 5. The Loss vs Epoch Trade Off

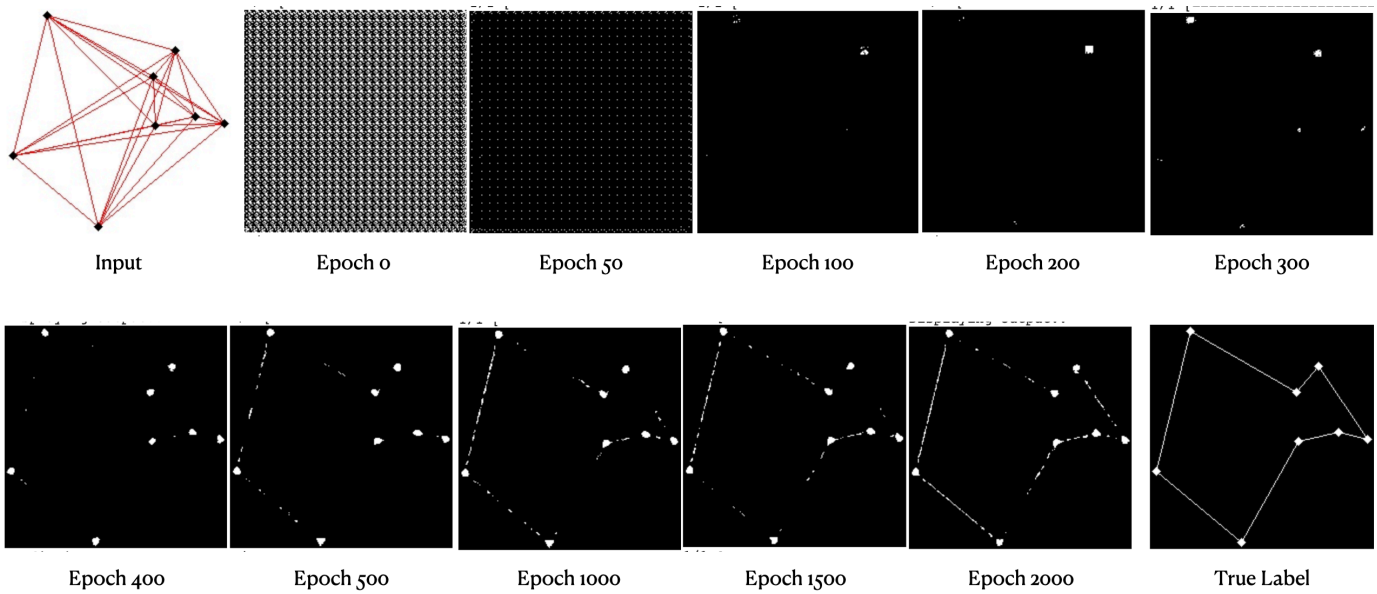


Figure 6. Epoch Wise Prediction of the FCN.

filter). For the FCN layers the Weights were initialized using Truncated Normal Distribution with Standard Deviation 0.01 and the L2 Regularization was also applied with $\lambda = 0.001$. For the De-convolutional layers no activation function was used and weights were also initialized using Truncated Normal Distribution with Standard Deviation 0.01 and the L2 Regularization was also applied with $\lambda = 0.001$.

The whole learning process of the network is lucidly depicted in Figure 6. In Figure 6 it can be clearly seen that the FCN network is appropriately capturing the the spatial information of the graphs nodes in order to find the optimal path.

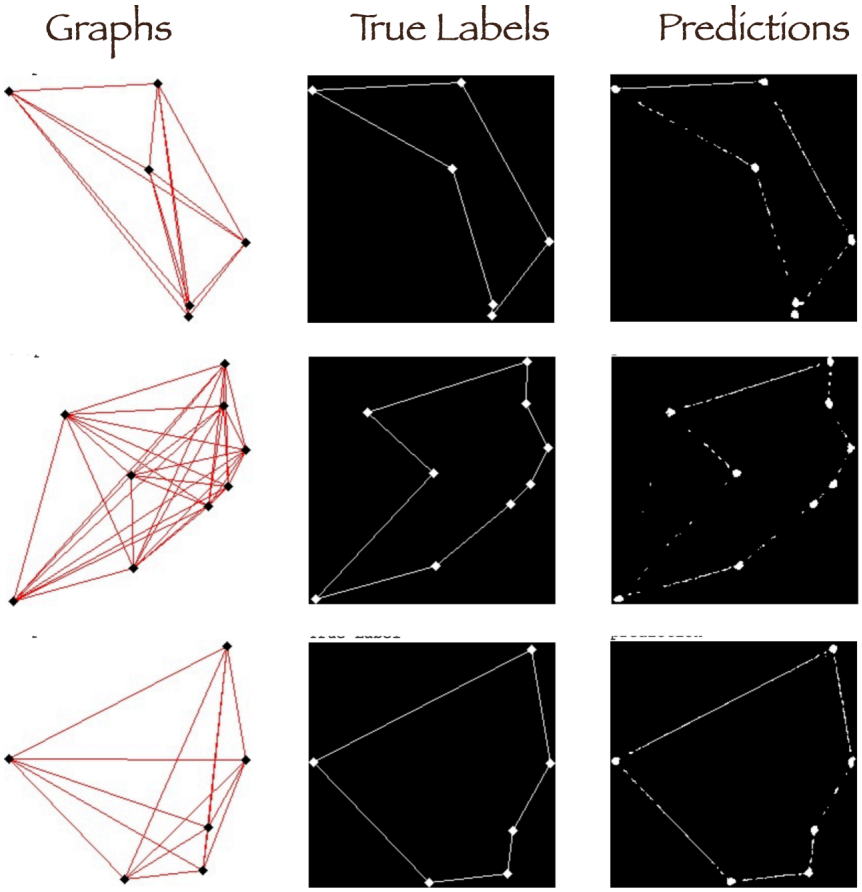
3. RESULTS

The trained network was able to solve majority number of graphs appropriately. The different result scenarios are show in Figure 7 and 8 .

From the results it can be seen that the network is able to map the input graphs to their respective Optimal Hamiltonian Cycles. One would wonder that the obtained results are a bit noisy, but it can be further improved via training for more epochs and with huge amount of data.

The main goal of the project was to prove that the Traveling Salesman Problem can be solved in Polynomial time, and which has been proved as the computational complexity of CNNs is polynomial.

4. CONCLUSION AND FUTURE WORKS



By training the model for longer time and on huge data set (for eg. 200K High Definition images) can significantly improve the quality of solution generated by the network.

One of the demerit of this application is that it requires huge amount of computational power to train the network. And when the number of graph nodes increase a 3D mapping of the graph nodes on the network will be a better choice.

Figure 7. The results obtained after training the FCN

So, the Fully Convolutional Network was successfully able to capture spatial information of graph nodes and thus was able to identify optimal path.

Given the Sensitivity of the Application the margin of error is very low as with even 1% error in a label maps of 224x224x2 means the network is predicting 500 pixels wrong (and that's the reason the network is predicting such noisy images). This problem can be avoided by choosing a label map where the path connecting the nodes is thick.

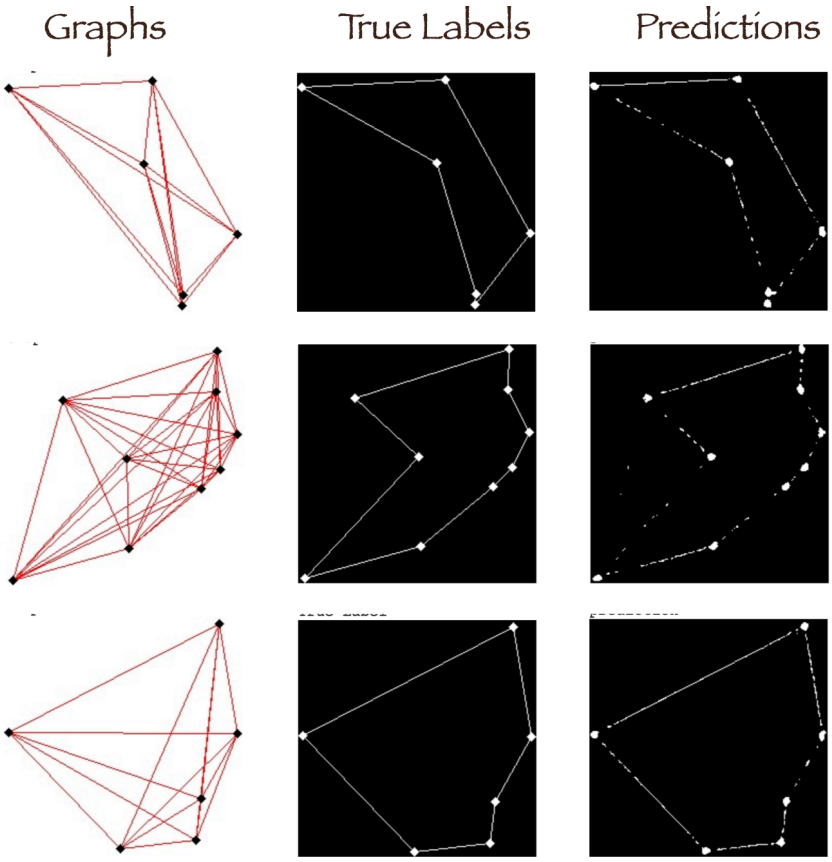


Figure 8. Some More results obtained after training the FCN