Line Detection using Convolutional Neural Networks

Muneeb Aadil

National University of Computer and Emerging Sciences
imuneebaadil@gmail.com

Abstract

In this work, we present a novel approach of line detection through CNNs (convolutional neural networks) which can be used as a first stepping stone towards building an end-to-end neural network to detect lines. The CNN-based method would eliminate the limitations of standard hough transform including hyperparameter finetuning at test time. We empirically show that CNNs which incorporate scales of the image successfully detect lines while removing everything else. The promising results serves as a proof that line detection in natural images using end-to-end CNNs is a direction worth exploring.

I. Introduction

Standard HT (Hough Transform) is a popularly used method of estimating lines, given a binary image. However, standard HT requires an accumulator array whose size determines the level of incorporated detail. This introduces a tradeoff between precision and computational cost. Furthermore, the level of detail to be accounted in accumulator array differs from image to image which leads to hyper-parameter optimization *for each image*.

Given a binary edge-image, this project tries to detect lines using CNNs (Convolutional Neural Networks) by 'switching off' active pixels that does not belong to any line. We hypothesize that line detection requires a global structure of the problem due to which receptive field of view becomes an important parameter. Motivated by the idea of bigger receptive field of view, we used CNNs which incorporate scale space of the image. This provides consistently better results; thus reinforcing our hypothesis.

Results show that CNNs successfully detect lines while removing everything else. The promising results serves as a proof that line detection in natural images using end-to-end CNNs is a direction worth exploring.

II. Methodology

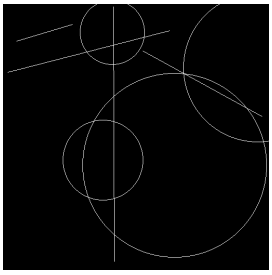
i. Overfitting Test

A 3-layer fully convolutional sequential neural network (explained in section ii) was used for initial overfitting test to determine the representational power and adequacy of the network.

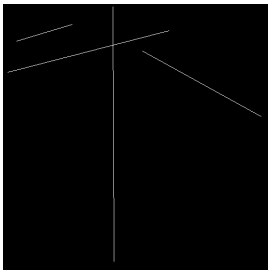
The results (as shown in figure 1) suggested that network having bigger filter sizes, thus having bigger effective receptive field size, tends to overfit the image better. Based on this, it was hypothesized that network needs to see the global picture instead of local spatial neighborhood to correctly detect a line. This makes sense because an arbitrary geometric shape (triangle, for instance) might look identical to line in local spatial neighborhood.

ii. Image-to-Image Sequential Network

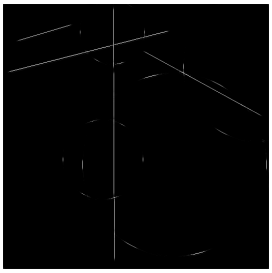
As the name suggests, it is a fully convolutional network which takes an images of arbitrary size (having a single depth channel) and outputs an image of same dimensions as input.



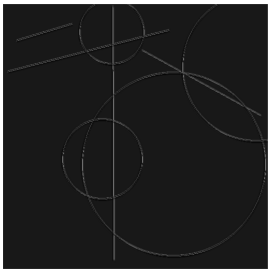
(a) *Input Image*



(b) *Target Image*



(c) 7×7 *Filters' Result*



(d) 3×3 *Filters' Result*

Figure 1: *Overfitting Test*

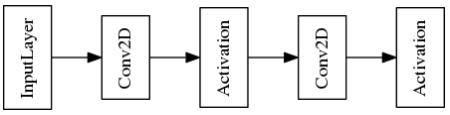
The architecture is shown in figure 2(a) and its specifications are listed in table 1.

Parameters	Value
Number of Filters	8 (1)
Conv Kernel Size	-
Conv Padding	Same
Conv Strides	1
Nonlinearity	ReLU (Sigmoid)
Network Depth	-

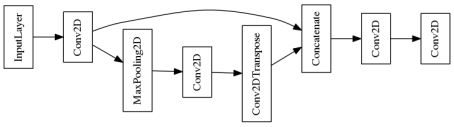
Table 1: *Image-to-Image Sequential Network specifications. Brackets indicate the value in last layer. Hyphen indicates the value was cross validated.*

iii. Image-to-Image UNet Network

One natural choice of dealing with scales in images is using image pyramids i.e set of repeatedly downsampled images. As downsampling also indirectly increases the effective receptive field size, U-Net (as shown in figure 2(b)) based image to image network seems an intuitive choice of network. The specifications



(a) *2-Layer Image to Image Sequential Network.*



(b) *1-Scale-Space Image to Image Unet. Activation layer is after every convolution layer but not shown for brevity.*

Figure 2: *Network Architectures*

of the model are listed in table 2.

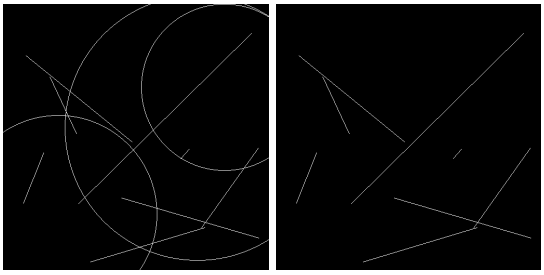
Parameters	Value
Number of Filters	8 (1)
Conv Kernel Size	-
Conv Padding	Same
Conv Strides	1
Nonlinearity	ReLU (Sigmoid)
Scale Space	-
Pool Kernel Size	2
Pool Strides	2
Pool Padding	Valid

Table 2: *Image-to-Image U-Net specifications. Brackets indicate the value in last layer. Hyphen indicates the value was cross validated.*

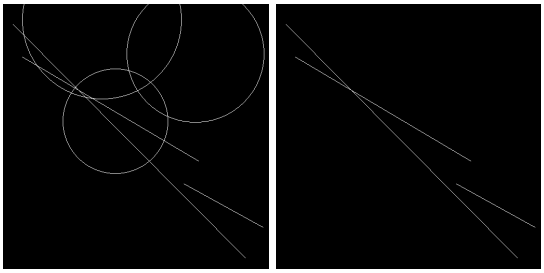
III. Experiments

i. Dataset

A dataset of binary images (spatial dimensions of 512×512) was generated synthetically. Input images contains lines and circles (drawn by randomly selecting parameters from uniform distribution) random number of times (maximally 10 lines and 5 circles are drawn.) Some samples of dataset are shown in figure 3 and dataset split is shown in table 3.



(a) *Input Image* **(b)** *Target Image*



(c) *Input Image* **(d)** *Target Image*

Figure 3: *Dataset Samples. (b) and (d) are ground truth images of (a) and (c) respectively.*

Set	Number of Images
Training	10,000
Validation	100
Test	100

Table 3: *Dataset Split Distribution.*

ii. Hyper-parameters Optimization

Hyper-parameters to cross validate in this problem includes choice of objective function, learning rate, sequential model's properties, and UNet's properties which are briefly described below:

Objective Functions: We used pixel-wise objective functions. Since white pixels are sparse, it's a highly imbalanced class problem for which weighted loss is a popular choice. We explored two following choices:

- Weighted Binary Cross Entropy Loss
- Weighted L2 Loss

where a weight for each pixel is simply the inverse of probability of label of that particular pixel.

We empirically found that weighted L2 Loss performs better because a) it is numerically stable (*log* probabilities in binary cross entropy sometimes generate NaNs which causes training instability), b) It gives visually better results, c) model converges faster.

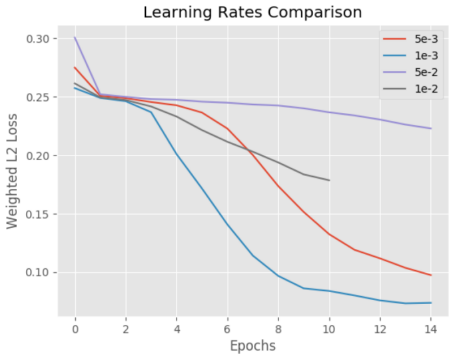


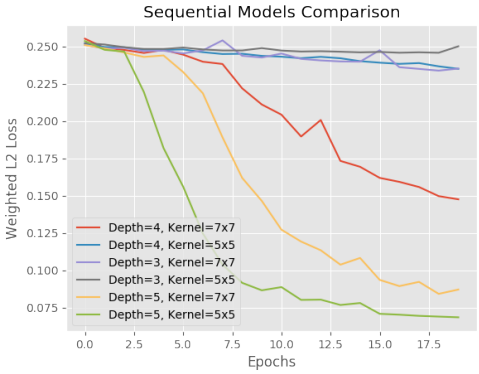
Figure 4: *Learning rates comparison (on training set) on sequential model with Kernel Size=3, network Depth=3*

Learning Rates: Comparison of learning rates is shown in figure 4(b). We empirically found $1e$-3 to be the best learning rate as it was converging faster.

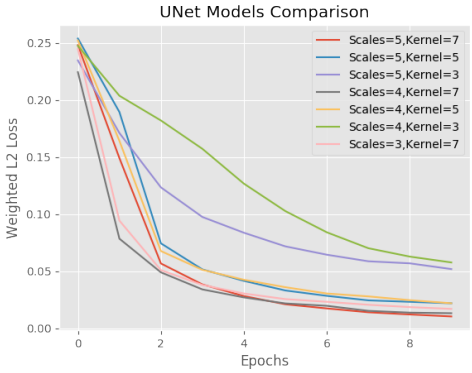
Sequential Models As convolution's filter size and network depth plays role in conjunction to determine effective receptive field of the network, both parameters were cross validated and results are shown in figure 5(a). We refer to sequential models having d depth and k kernel size as Seq-d-k in the text.

UNet Models Similarly, in UNet-style networks, convolutional's filter size and number of scale spaces collectively determine effective receptive field of the network. The results of cross validation are shown in figure5(b). We refer to UNets having s scale space and k kernel size as UNet-s-k in the text.

The results are consistent with the hypothesis

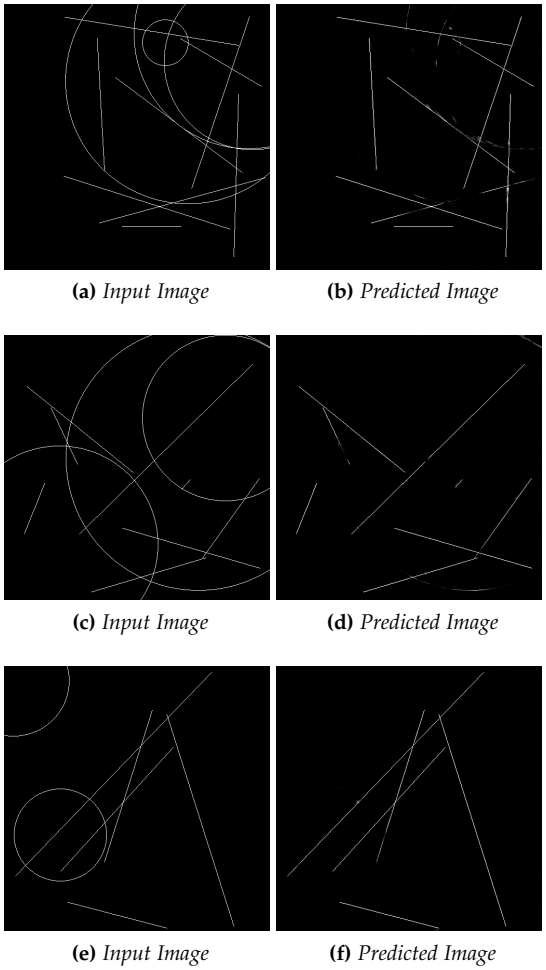


(a) *Sequential Models*



(b) *UNet Models*

Figure 5: *Learning Curves (on Validation Set) of Models*



(a) *Input Image* **(b)** *Predicted Image*

(c) *Input Image* **(d)** *Predicted Image*

(e) *Input Image* **(f)** *Predicted Image*

Figure 6: *Samples of Predictions of UNet-7-7 on Test Set.*

in section i: The deeper network with bigger filter sizes, thus having bigger receptive field, performs better because of global structure of the problem.

iii. Training

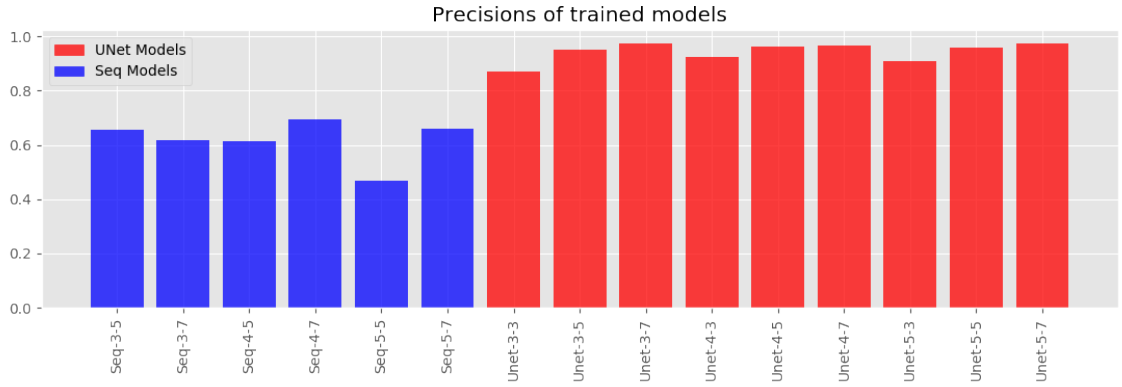
We used batch size of 64 and trained using ADAM optimizer ($\beta_1 = 0.9$, β_2=0.999) with learning rate of 1e-3.

The source code for dataset generation, training networks, and generating test-time results is written in Python using Keras deep learning framework and is publicly available online[1]. Our proposed UNet model roughly takes two hours to train on NVIDIA Tesla K80.

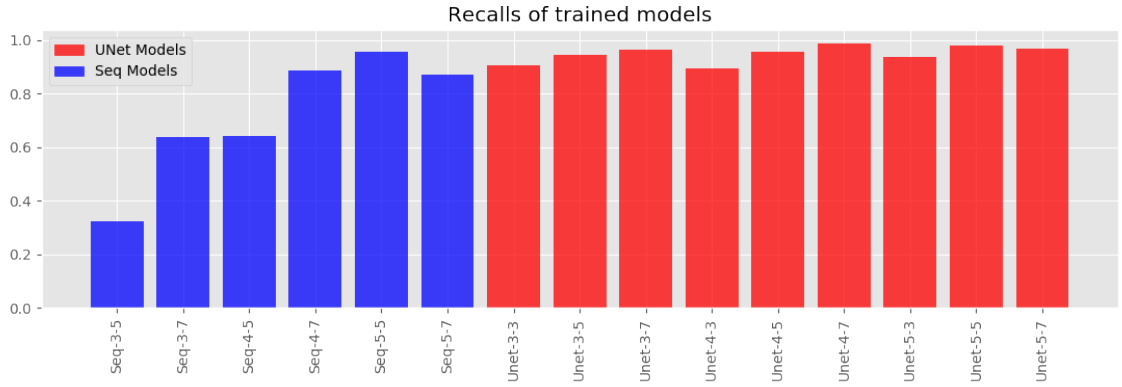
[1] `https://github.com/muneebaadil/Hough-Transform-using-CNNs`

iv. Results

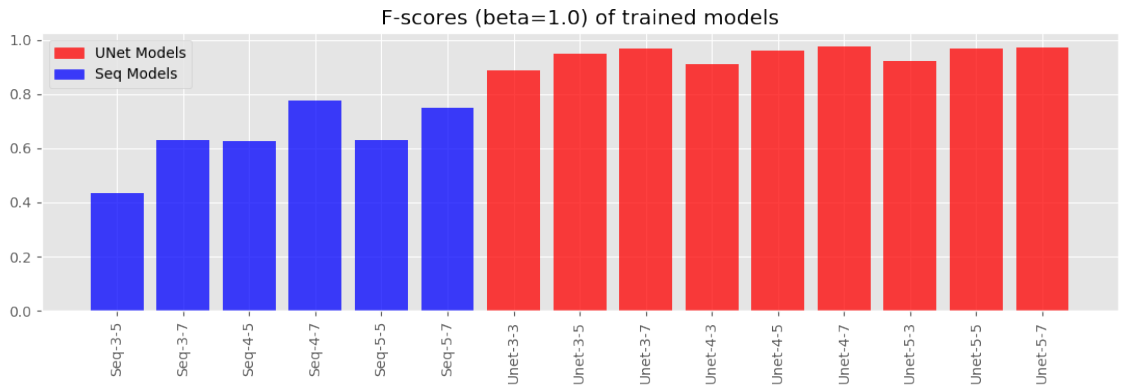
We use popularly-used metrics precision, recall, and fscore as an evaluation metric. UNet models give significantly better results than sequential models because UNet, as explained above, incorporate scales in architecture through repetitive down-sampling. The qualitative results of best performing model (UNet-7-7) is shown in figure 6 and quantitative results including comparison of all the models trained is given in figure 7.



(a) *Precision Comparisons*



(b) *Recall Comparisons*



(c) *F-score comparisons*

Figure 7: *Evaluations on test set for all trained models. Cutoff threshold is 0.5 for all experiments.*

IV. Conclusion and Future Work

In this work, we tried to tackle line detection problem using CNNs which will eliminate the need of fine-tuning hyper-parameters during test-time.

We hypothesized that global structure of image is important for this problem and showed that empirical results are consistent with our hypothesis i.e models that incorporate scale-space of the image (such as UNet) consistently perform better than those which don't.

Lastly, we show visually that CNNs are successfully able to detect lines while filtering out other objects. The promising results serves as a proof of concept that end-to-end line detection in natural images through CNNs is a direction worth exploring.