ENGN8535 Project Report
An Improved Texture Prediction Network (ITPN)

Weiyue Wang
u5889098
The Australian National University, Acton, ACT 2601, Australia

***Abstract*—Texture prediction has been an important task in the field of computer vision. This task aims to isolate the textures from images so that other advanced applications such as image smoothing could be achieved. Previously, Convolutional Neural Networks (CNN) are applied to achieve such task; For example, a network named Texture and Structure Aware Filtering Network (TSAFN) proposed in 2018 has demonstrated better performance over traditional approaches [1]. In this article, we will have some analysis on the existing Texture Prediction Network(TPN), a texture isolation network for TSAFN, and suggest a network named Improved Texture Prediction Network (ITPN).**

I. Introduction

Image smoothing is becoming an increasingly important technique when dealing with Computer Vision (CV) tasks by removing trivial textures while retaining salient structures. Traditionally, we can achieve this by using kernel-based methods by averaging each pixel with its neighbouring pixels. Some common averaging techniques including rolling guidance filter[3], segment graph filter [2] are based on assumptions such as the gradients of structures are always large, and textures have small oscillations in colour intensities [1]. By using deep convolution networks, we can construct a network to ”learn” the pattern of removing irregular and complicated textures. TSAFN, on the other hand, is built to solve these problems.

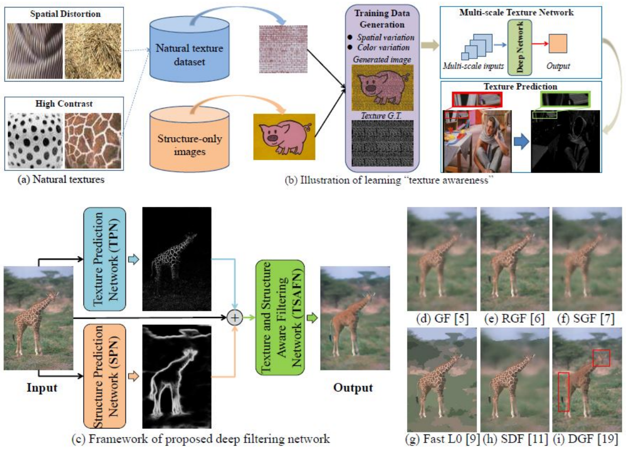
In this paper, however, we only analyze the texture predicting neural network of this system by proposing an improved network, Improved Texture Prediction Network (ITPN). The reason why we need to improve the existing TPN is because during the practical use of this network, we found some information would be lost due to the down-sampling and up-sampling design, leading to the inaccuracy in texture predictions. The basic idea of ITPN is by modifying the down-sampling and up-sampling process, we could mitigate the loss of texture features. In the following chapter II, we will introduce the background of this network, including related work and our motivation in details.

II. Background

In this section, we will present some background information, including the related work and our motivation in building the ITPN.

A. Related Work

Figure 1 shows the basic idea of the implementation of TSAFN network. The basic idea of this network is to train the data set separately on a TPN and a SPN respectively, then use the textural output and structural output from TPN and SPN as inputs for a TSAFN network to learn the mixed output pattern. The training data set, on the other hand, is generated by mixing textures of different variations with structural images in Figure2. Also, the texture ground-truth of the data set will also be generated. For this paper, we used the same method of generating data set as the original paper.



(a) Natural textures (b) Illustration of learning "texture awareness"

(c) Framework of proposed deep filtering network (d) GF [5] (e) RGF [6] (f) SGF [7] (g) Fast L0 [9] (h) SDF [11] (i) DGF [19]

Fig. 1: Overview of the design and implementation of TSAFN [1]. (a) Natural textures are difficult to detect due to their spatial distortion and high contrast. (b) To generate the training set, the author of the network blended the varied texture patterns with structure-only images (i.e. cartoon images). (c) The filtering network is composed by three sub-networks, TPN, Structure Prediction Network (SPN) and TSAFN. TSAFN uses the combined results of textures generated by TPN, structures generated by SPN. (d-i) Traditional approaches for implementing texture filtering

Figure 3 shows the structure of TPN. The input (Figure 4) of this network has been firstly down-sampled into $\frac{1}{2}$, $\frac{1}{4}$ and $\frac{1}{8}$ of the original size by three max pooling layers with $2*2$ kernels of $stride = 2$. Afterwards, these four channels are trained in three layers of CNN before the three down-sampled channels are up-sampled by deconvolution layers. After concatenating these 4 branches (each branch is 4-channeled) together, another T-convolution layer is used for reducing 16 channels to 1 channel for the output (Figure 5).

To train the network, an adam optimizer for Mean Squared Error (MSE) loss has been applied [1].

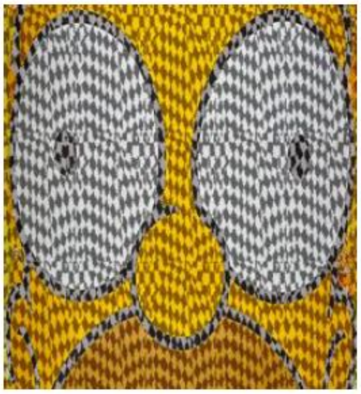


Fig. 2: An example of training data set

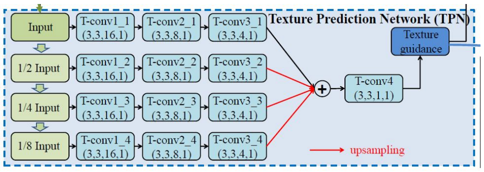


Fig. 3: The Structure of TPN [1].

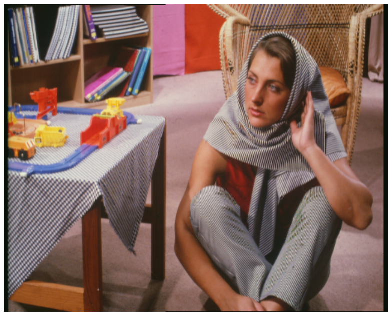


Fig. 4: An example of input image

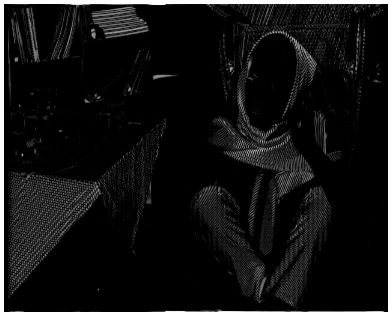


Fig. 5: An example of texture guidance [1]

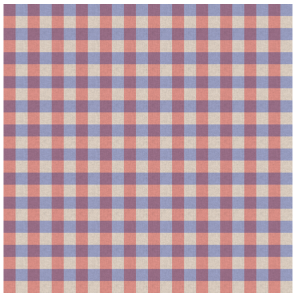


Fig. 6: Red checkerboard fabric on a flat surface

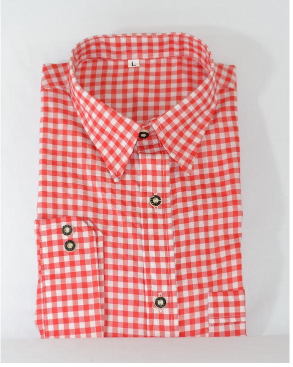


Fig. 7: A red checkerboard shirt

B. Motivation

In this section, we will introduce the motivation from a perspective of texture detection tasks, as well as from a point of view of improving existing TPN.

1) What is Texture?: Although the textures could be defined as regular or irregular patterns distributed on object surface, it is hard to mathematically determine the feature of textures. Also, even though some textures may follow some certain patterns (such as the red checkerboard fabric on a flat surface in Figure 6), it is still hard to identify natural textures since they could either be distorted or rotated given the surface may not always be ideally flat (such as the red checkerboard shirt in Figure 7). If we could find a robust approach to isolate textures from their surface, we could thus smooth any surface given an image. The TPN, on the other hand, is able to implement the isolation of texture from surface. However, there are some issues with the existing TPN. We will discuss these issues in the next section.

2) Why the existing TPN is not perfect?: As was introduced in Chapter II-A, the TPN was using pooling layers for down-sampling and deconvolution layers for up-sampling. This means the kernels of the pooling layers have a size of $2 * 2$, with $stride = 2$. In terms of the deconvolution layer for up-sampling from $\frac{1}{8}$, $\frac{1}{4}$ and $\frac{1}{2}$ of input to the original size, the kernel has to be $4 * 4$ with $stride = 8$, $stride = 4$ and $stride = 2$ respectively. For the down-sampling process, since the maxpooling layer only picks the maximum value

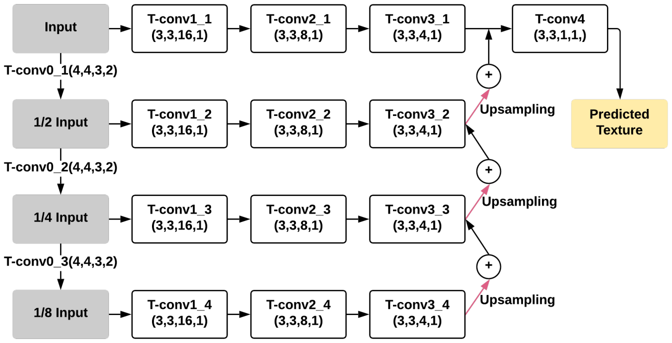


Fig. 8: The proposed ITPN structure

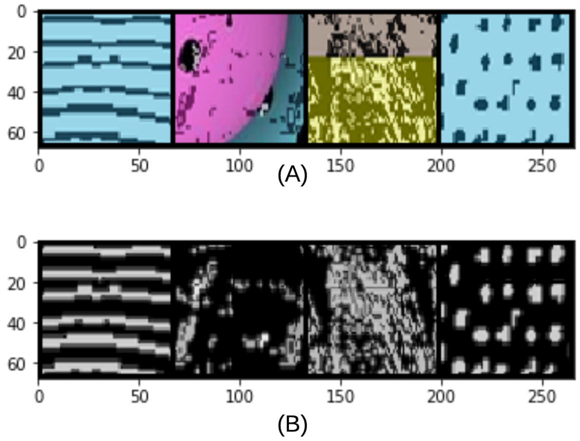


Fig. 9: (A) An example of a data batch and (B) ground-truth batch

from each kernel, the network is unable to ”learn” the down-sampling process. On the other hand, large strides in the up-sampling process will also result in a loss of information. Thus, we determined to design and implement a network that could overcome such issues.

III. Approach

In this chapter, we will introduce the implementation of our design as well as the experiment setup. Section III-A will show the design of the network structure, while section III-B will present the training of the ITPN. The last section, on the other hand, will demonstrate the experimental design and analysis.

A. Network Design

As was mentioned in section II-B2, the ITPN will focus on improving the up-sampling and down-sampling structure of the existing TPN. Figure 8 demonstrates the design of ITPN. An explanation of the design is as included in the following sections.

1) Convolutional down-sampling: To replace the pooling down-sample layer, a 3-channeled $4 * 4$ T-convolutional kernel with $stride = 2$ has been applied. The output of each convolutional down-sampling could thus remain the same dimensions as the previous pooling down-sampling. Recall that for the maxpooling down-sampling technique, only the pixel with the largest value for every $2 * 2$ kernel will be kept, while the remaining 3 pixels will be disposed. This will lead to a loss of some important information, due to the pooling technique only maintains the regional maximum pixel. By replacing the pooling layers with the convolutional layers, the network is able to ”learn” the pattern for down-sampling, reducing the loss of information.

2) Step-wise up-sampling: In the section II-B2, we have mentioned that the up-sampling for the TPN may potentially ”skip” some features due to large strides. For example, when up-sampling the $\frac{1}{8}$ of the input to the original size, a $4 * 4$ de-convolutional kernel with $stride = 8$ is applied. This means the de-convolutional kernel will have to ”recover” more information than other two de-convolutional kernels that has less strides, leading to unbalanced capabilities for recovering features from different scales. By using a step-wise approach, each up-sampling only increases the output to the double of the current tensor before concatenates with the tensor for the next-level. As a result, the strides of up-sampling de-convolutional layers is always fixed to 2, reducing the probabilities for ”skipping” some features.

B. Network Training

1) Loss Function: The loss function of the network is defined as the mean squared error (MSE) between the predicted texture guidance map and the ground-truth:

$$L_\theta = \frac{1}{N}\sum_{i}^{N} ||T_i - T_i^*||_2^2$$

, where N represents the total number of pixels in the image, while $*$ indicates the ground-truth. The optimizer we applied is Adam Optimizer. The tuning of hyper-parameters of Adam Optimizer is included in section III-C.

2) Training Data Setup: As was introduced in the section II, we trained this network based on the same data generation method as the original paper and we got 2000 images. The size of the original data is $512 * 512$. However, to further increase the variety of the data set, we randomly flipped and cropped $64 * 64$ image patches from the original images. For network training and validation, we used these image patches with a batch size of 4. An example of the data batch could be seen as Figure 9.

C. Experimental Design and Analysis

In this part, we will look into the design of the experiment to evaluate the performance of our network. Also, an analysis for the purpose of each design will be provided.

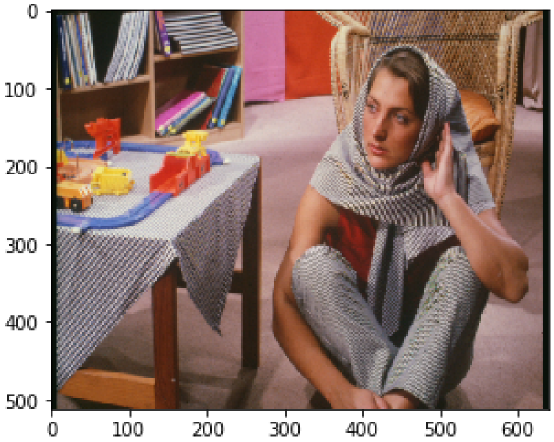


Fig. 10: A sample scaled test image

1) Training device: We implemented this network in PyTorch and trained and tested the data on a NVIDIA GTX1070 graphics card.

2) Data set: Since there is a total of 19,505 data sets to fully train TPN [1] and that will take a tremendous training time for a GTX1070 graphics card (compared with the NVIDIA Titan X used in the TPN paper), we only obtained a small amount of data set (1,500 images) for training and 500 images (25% of the entire data set) for validation. Another reason why we used such a small number of training set is because we want to compare the performance of the ITPN against other network structures.

3) Training: The specific experimental design as well as the purposes for each design could be seen as follows.

Experiment 1. Train the original network and the convolution down-sampled network on 1,500 images of the entire data set, calculate the validation loss for each epoch. The network is trained for 100 epochs, with a learning rate of 0.01.
The purpose of this experiment is by using small data set and large learning rate, we can validate the down sampling modification in both qualitative approach (analyze how effective it is by looking into the output for realistic images) and quantitative approach (by numerically analyze the training loss and validation loss).
Experiment 2. Train the ITPN on 1,500 images of the entire data set, calculate the validation loss for each epoch. The network is trained for 100 epochs, with a learning rate of 0.001.
The purpose of this experiment is by using small data set and large learning rate, we can compare the performance of the ITPN and the two previous networks in both qualitative approach (analyze how effective it is by looking into the output for realistic images) and quantitative approach (by numerically analyze the training loss and validation loss).
Experiment 3. Decrease the learning rate to 0.0001 and train three networks on 1500 images of the entire data set, calculate the validation loss for each epoch.
The purpose of this experiment is by using small data set and small learning rate, we can smooth the validation curve and compare the performance of the proposed ITPN and the two previous networks in both qualitative approach (analyze how effective it is by looking into the output for realistic images) and quantitative approach (by numerically analyze the training loss and validation loss).

IV. Result and Discussion

In this chapter, we will demonstrate the results according to the experiments shown in the previous chapter. Also, some discussion will be provided to analyze the results qualitatively and quantitatively.

*1) Results for **Experiment 1**:* Figure 11 and Figure 12 shows the training and validation curves for the original TPN and convolution down-sampled TPN respectively. It could be observed that the training loss of the original TPN converged to 0.023, which is similar to 0.022 of that for the convolution down-sampled one. On the other hand, the validation loss for the two networks could not be observed clearly due to the fluctuation of the curve. The analysis of the validation loss will be implemented in Experiment 3, where the learning rate has been decreased to 0.0001.

Figure 13 and Figure 14 shows the output of these two networks respectively on the same test sample. A comparison of details between these outputs could be seen as Figure 15. It could be observed that compared with the original TPN, the TPN with convolution layers as up-sampling layers has better performance in keeping the texture details.

*2) Results for **Experiment 2**:* Once we validated that the convolutional down-sampling is more effective in keeping the details compared with pooling layers, we can thus further modify the up-sampling layer in this experiment. This fully-proposed network is thus our proposed ITPN.

The training and validation loss is as shown in Figure 16. It could be observed that the training loss converged to a slightly smaller value of 0.021 compared with previous result. Figure 17, on the other hand, shows the output of the proposed ITPN given a sample test image. A comparison between the details for the test image, convolution up-sampled TPN and ITPN outputs could be seen as Figure 18. It could be observed that the ITPN output is overall darker compared with the previous design; however, unlike the output of original TPN, the texture of the darkened part is still visible and clear. In comparison with the input image, the ITPN is better at retaining the texture considering the highlights and shadows, while its counterpart makes the textures in shadows brighter.

To numerically explain this, we could express the original up-sampling as the following equation in a linear form:

$$\hat{y} = X_1 + A_2^T X_2 A_2 + A_3^T X_3 A_3 + A_4^T X_4 A_4$$

, where $\hat{y}$ is the output of the up-sampling and X_1, X_2, X_3 and X_4 are the input from the four branches of the network. Denote that the dimensions of these four matrices are 64×64, 32×32, 16×16 and 8×8 respectively, while the dimensions

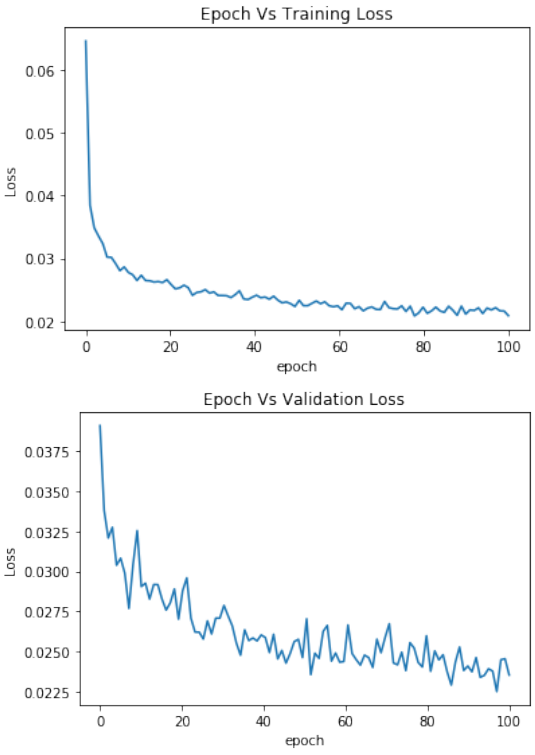


Fig. 11: (a) The training loss of TPN for **Experiment 1** (b) The validation loss of TPN for **Experiment 1**

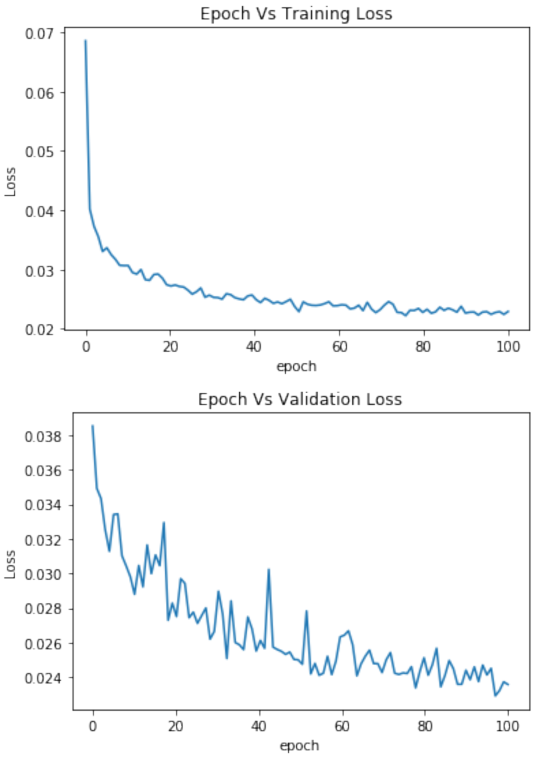


Fig. 12: (a) The training loss of convolution down-sampled TPN for **Experiment 1** (b) The validation loss of convolution down-sampled TPN for **Experiment 1**

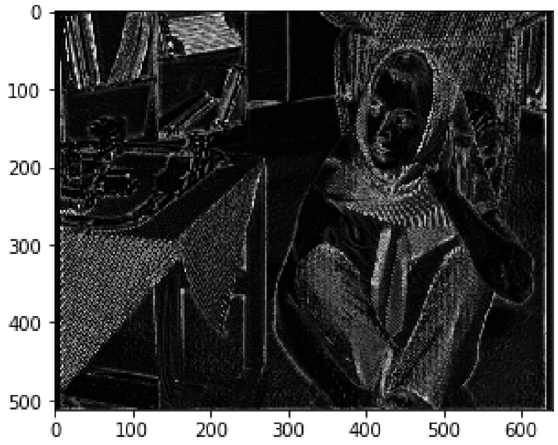


Fig. 13: The output of TPN for **Experiment 1**

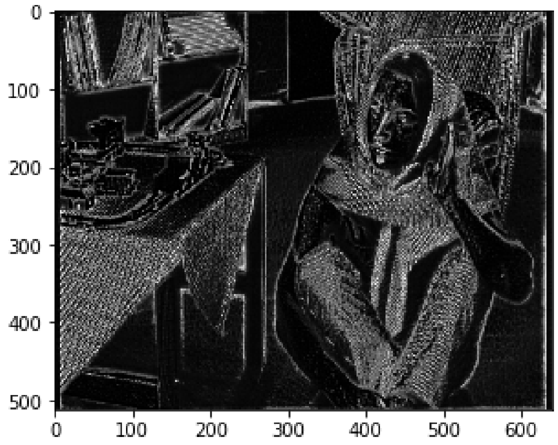


Fig. 14: The output of convolution down-sampled TPN for **Experiment 1**

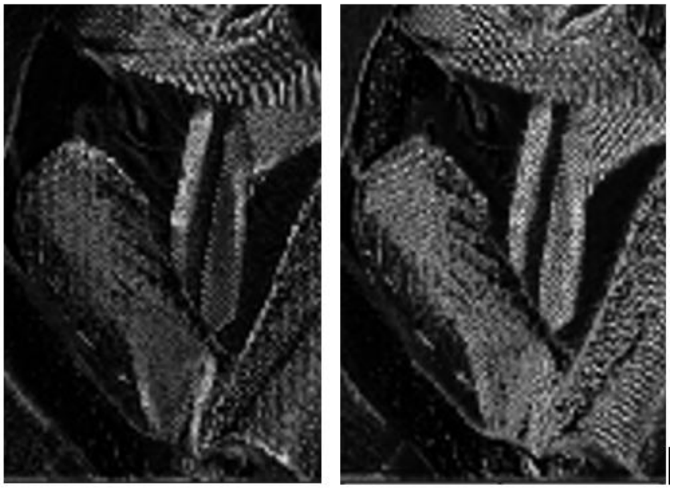


Fig. 15: A comparison of details for the testing for original TPN (left) and convolution down-sampled TPN (right)

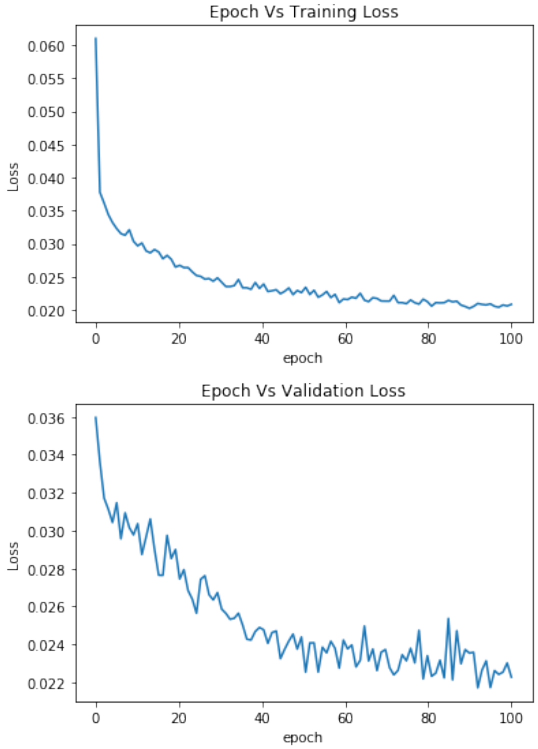


Fig. 16: (a) The training loss of ITPN for **Experiment 2** (b) The validation loss of ITPN for **Experiment 2**

of the weights A_2,A_3 and A_4 are 32×64, 16×64 and 8×64 respectively.

In contrast, the proposed up-sampling could expressed as:

$$\hat{y} = X_1 + B_2^T(B_3^T(B_4^T(X_4)B_4 + X_3)B_3 + X_2)B_2$$

$$= X_1 + B_2^T X_2 B_2 + B_3^T B_2^T X_3 B_2 B_3 + B_4^T B_3^T B_2^T X_4 B_2 B_3 B_4$$

, where the dimensions of the weights B_2,B_3 and B_4 are 32×64, 16×32 and 8×16 respectively. For the matrix X_4 which extracts the features with large scales, it is heavily weighted with B_2,B_3 and B_4. This could explain the observations that the predicted textures for shadowed parts could be less obvious. In the meantime, since the weight for each scale is also correlated with other scales, the proposed ITPN could be more consistent in terms of different scale levels.

*3) Results for **Experiment 3**:* In the previous experiment, we plotted the curve of validation loss. However, it is hard to observe the convergence due to large learning rate. In this experiment, we reduce the learning rate to 0.0001 and combine the previous two experiments so that we can implement both qualitative and quantitative analysis. Figure 19, Figure 20 and Figure 21 shows the curve of the loss for training of each network. It could be observed that when we take learning rate as 0.0001, the converged validation loss for these three networks are roughly at the same level at around 0.028, although that for the proposed ITPN is slightly smaller. In terms of the qualitative analysis on this experiment, we compared some details against each other in Figure 25. It could be observed that the texture details have been better kept by the proposed ITPN in the selected three areas marked in Figure 25 and Figure 26.

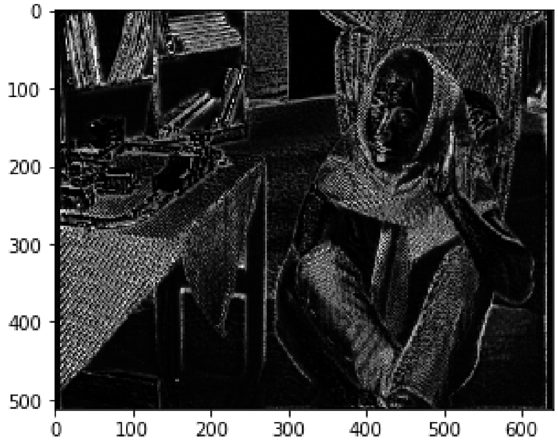


Fig. 17: The output of convolution down-sampled TPN for **Experiment 2**

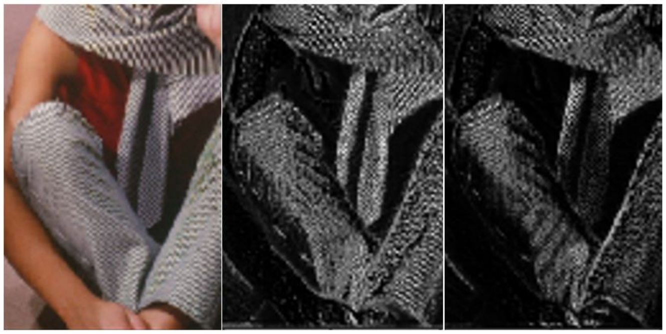


Fig. 18: A comparison of details for the input image (left) ,convolution down-sampled TPN (middle) and ITPN output (right)

V. Conclusion

In this paper, we firstly proposed a modified TPN by changing its up-sampling and down-sampling structure. After that, we setup three experiments to validate the design. To analyze each modification step-wise, we firstly looked into the result for the original TPN and the TPN with convolutional down-sampling. Afterwards, we compared the convolutional up-sampling in the full ITPN network against the previous result. After validating that the design to be effective in first two experiments, we reduced the learning rate to smooth the curve for the validation loss. To summarize, compared with the previous TPN design our ITPN is able to ”learn” the down-sampling process so that the features could be kept, rather than some being wasted by pooling layers. In terms of up-sampling process, our ITPN is better at ”detecting” the features between different scales by making the weights for each scale relevant to features in other scales. Also, the number of latent variables in the up-sampling process is less than that in the TPN, making

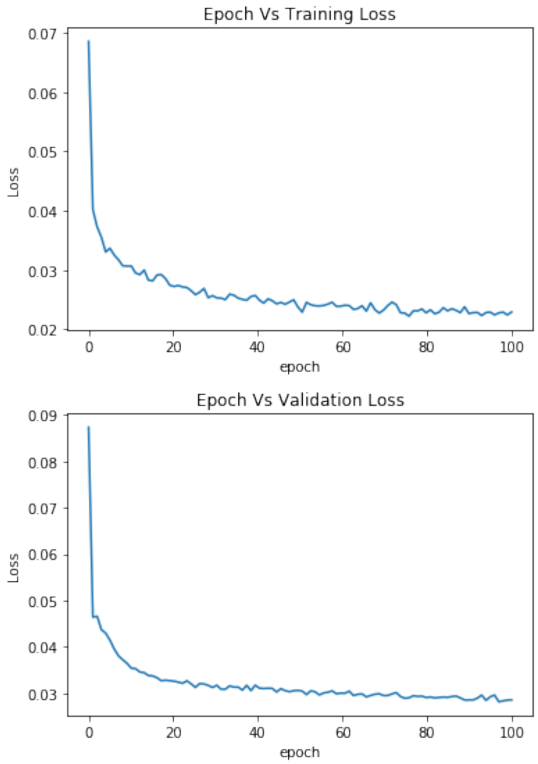


Fig. 19: (a) The training loss of TPN for **Experiment 3** (b) The validation loss of TPN for **Experiment 3**

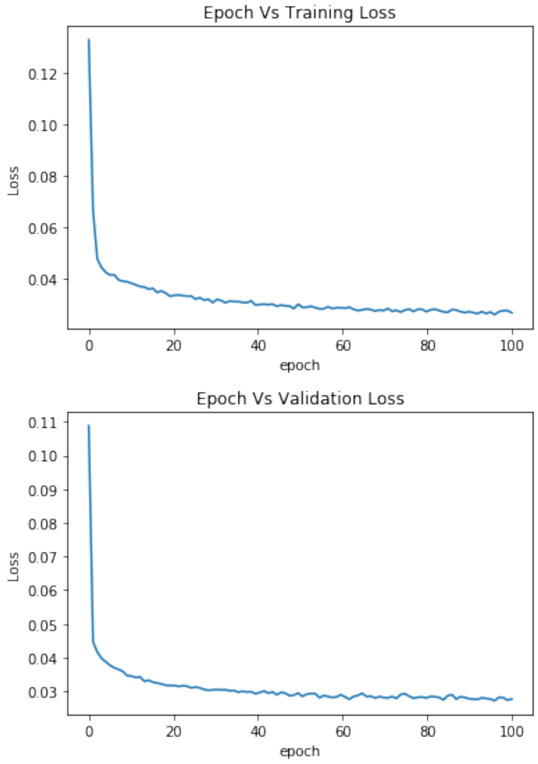


Fig. 20: (a) The training loss of convolution down-sampled TPN for **Experiment 3** (b) The validation loss of convolution down-sampled TPN for **Experiment 3**

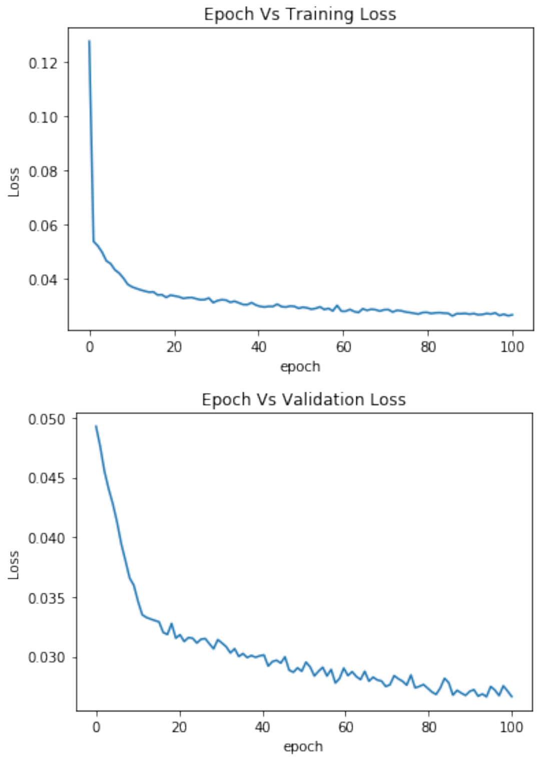


Fig. 21: (a) The training loss of proposed ITPN for **Experiment 3** (b) The validation loss of proposed ITPN for **Experiment 3**

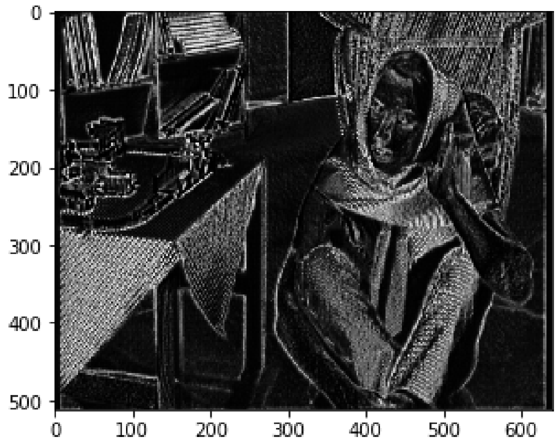


Fig. 22: The output of TPN for **Experiment 3**

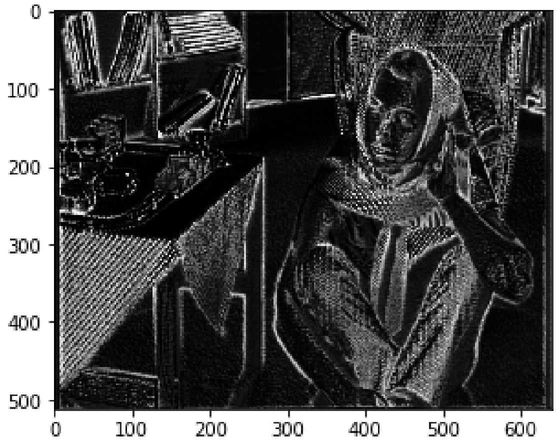


Fig. 23: The output of convolution down-sampled TPN for **Experiment 3**

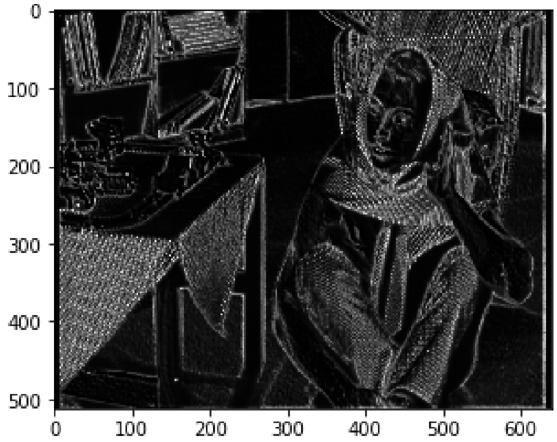


Fig. 24: The output of proposed ITPN for **Experiment 3**

ITPN more advantageous when the data set is small.
Due to the time and GPU constraints, we only trained the network on 2,000 data set and implemented some experiments for comparison with different network structures. For the future work, it is worthwhile to fully train and analyze the performance of the ITPN on more data set.

References

[1] Kaiyue Lu1,2, Shaodi You2,1, Nick Barnes2,1, *Deep Texture and Structure Aware Filtering Network for Image Smoothing*, ECCV2018,Canberra, Australia, 2018

[2] He, K., Sun, J., Tang, X.: Guided image ltering. IEEE transactions on pattern analysis and machine intelligence 35(6), 13971409 (2013)

[3] Qi Zhang1 Xiaoyong Shen1 Li Xu2 Jiaya Jia1: Rolling Guidance Filter ECCV2014,Hong Kong, China, 2014

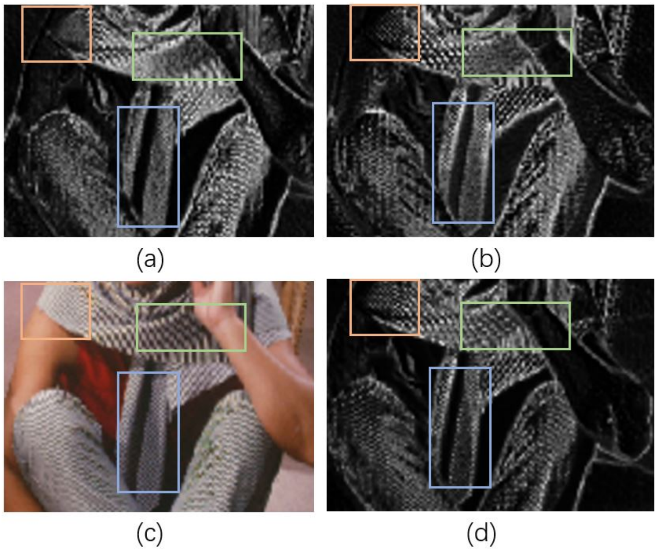


Fig. 25: A comparison of details for the testing for (a) original TPN, (b) convolution down-sampled TPN, (c) original image, and (d) proposed ITPN

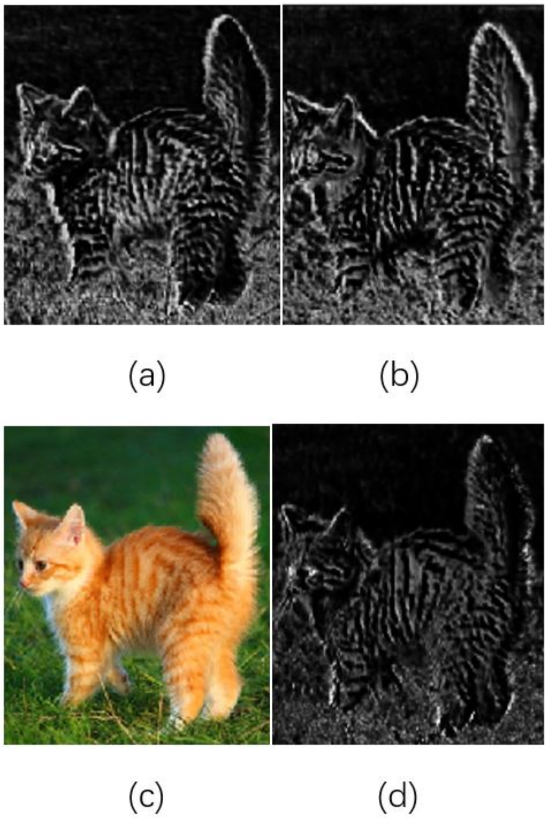


Fig. 26: Another comparison of details for the testing for (a) original TPN, (b) convolution down-sampled TPN, (c) original image, and (d) proposed ITPN