Comic text/speech region detection and textless style transfer using CNN

Jiaao Guan
UC San Diego
jig186@eng.ucsd.edu

Kuan-Wei Chen
UC San Diego
kuc010@eng.ucsd.edu

Po-Jung Lai
UC San Diego
polai@eng.ucsd.edu

Hsiao-Chen Huang
UC San Diego
hsh030@eng.ucsd.edu

Abstract

A method of reconstructing a meaningful content image with a different style is called style transfer. Here we apply this on comic images and paste back the original text, which may be distorted during transfer, on our new-styled comic image. This work separates into two tasks: first, we implement A Neural Algorithm of Artistic Style[2] which proposed by Gatys *et al.* to synthesize an image with new style. The algorithm allow us to recombine a content image and style image by extracting the feature of content and style in layers of convolutional neural networks(CNN) for each target image. Besides, we also train our network[13] to achieve our goal. Second, we adapt the Efficient and Accurate Scene Text detector (EAST) by Zhou *et al.*[14], train it to recognize text and speech positions, and put the original texts back into their positions later after style transfer.

1 Introduction

Different authors of comic books get different drawing styles, and it is interesting to see comics replaced by a different style. Modern convolutional neural networks(CNN) based algorithms were already able to automatically synthesize the texture from a source image and keep this texture to perform style transfer to another content image and output reasonably good-looking image with new style. It is intuitive to adapt the state-of-the-art image style transfer technique and apply on comic style transfer tasks.

However, although Gatys *et al.* in the paper[2] showed an outstanding performance of style transfer, there are still some task specific problems never addressed by the paper. For instance, there are text in the comic images and transferring style of comic pages directly will distort texts on the image. We are trying to solve this text distortion problem by performing style transfer on images from comic books while keeping texts from the original images.

In order to keep the original texts, we need to know the positions of the texts on each image. To automate this process of finding text positions instead of manually selecting text locations, we plan to use a text detection algorithm. Our choice of text detection tool is EAST[14], which is a powerful CNN based word and text line detector especially good at detecting oriented texts in real world scenes.

Our project is focusing on how to properly train the style transfer network and the text detection network, and to combine the two networks and build up an algorithm to achieve end-to-end comic style transfer while keeping the original text and speech intact.

Preprint. Work in progress.

2 Motivation and Related Work

Our project is inspired by the recent successes of computer image style transfer[2], and the lack of their robustness on maintaining the readability of texts and speech lines while applying style transfer on text-rich images like comics. This task is composed of image style transfer and text line detection. Our work combines and modifies some recent powerful network designs.

The paper [2] provides one of the state-of-the-art approach to synthesize the texture information from a image and the content information from another image. Both style and context information were extracted using VGG19 [11] network.

The paper [14] demonstrates a recent text scene detection using PVANET[7] as convolutional neural network(CNN) architecture. The pipeline is designed to robustly predict text lines and words even with orientations and arbitrary quadrilateral shapes.

We aim to design an algorithm to use EAST[14] test detector to automatically detect and maintain the text information while doing style transfer using [2] to the remaining text-less comic images.

2.1 Adapt style transfer networks from object recognition

In the paper [2], the task was executed by loading the pretrained model to find image representations that independently model variations in the semantic image content and the style. Deep Convolutional Neural Networks [11] was trained to perform object recognition and the network was able to extract high-level image content in generic feature representations. With the powerful result from VGG19, researchers focused on speeding up convolutional neural networks and Iandola, Forrest N., *et al.* come up with a new architecture of convolutional neural networks for object recognition with 50x fewer parameters[4]. The accuracy of SqueezeNet is almost the same as AlexNet[8] but the speed of Sqeezenet is 50 times faster than AlexNet.

2.2 Text recognition and localization

The Efficient and Accurate Scene Text detector (EAST)[14] introduced a fast and accurate text detection pipeline with only two stages, which predicts word and text line positions and text content prediction directly from the input image. This approach of simplifying intermediate stages, while comparing to some of the previous CNN based text detectors [1, 3, 6], is more accurate and more efficient. The CNN network used by this EAST algorithm can be chosen from any regression CNN, but the paper suggested the use of PVANET[7], which is a deep but lightweight CNN object detector, as giving the best result.

3 Description of Method

In this section, we are going to provide how we solve the problem by dividing the problem into two parts: style transfer and text detection/recovery.

3.1 Architecture

3.1.1 Style Transfer

For the adaptive style transfer task, the problem is addressed by minimizing the difference in style feature domain and the difference in content feature domain between input and output image. Therefore, we use a set of filtered images at each stage in CNN to extract the features. By computing the gradient of loss, we can implement the back propagation to iteratively update the output image. The architecture of style transfer is shown in Figure1.

Besides, we also train our own feed-forward Multi-style Generative Network(MSG-Net)[13] which is proposed by zhang *et al.*. It improve the previous network as a Fully Convolutional Network(FCN) that can accept arbitrary input size to generate the output transfer image. First, we take a bunch of style and content images as our training set. Pass through a Siamese network and transformation network respectively, then match second order contents based on the given styles with CoMatch Layers.

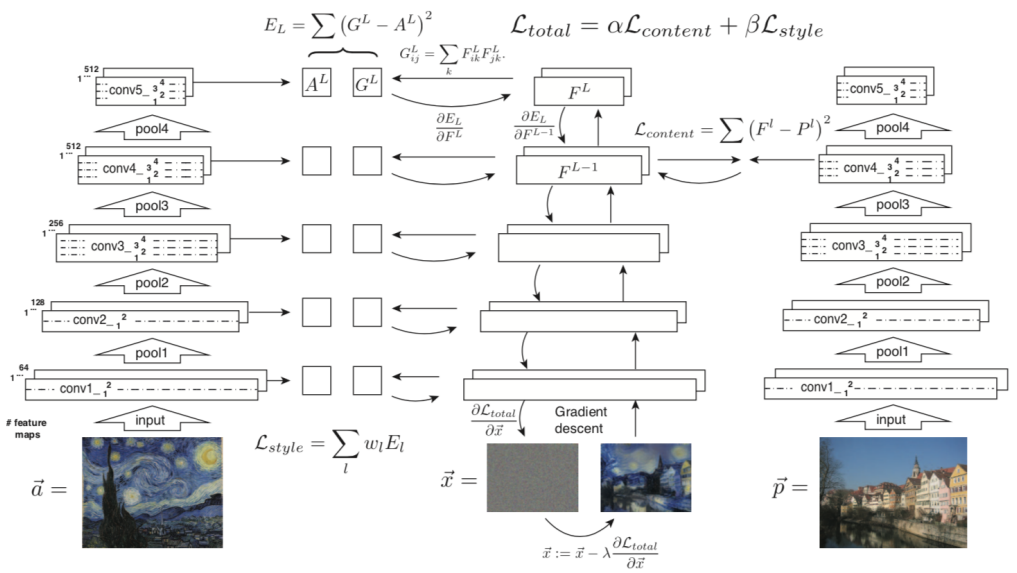


Figure 1: The architecture of style transfer[2]

3.1.2 EAST Text Detection

In the East text detection, an input image is fed into a multi-channel fully convolutional neural network (FCN), which then outputs labels including the coordinates of an axis-aligned bounding box (AABB) and its angle, which together form a rotated box (RBOX). The FCN can be divided into 3 stages. A feature extractor, a feature merging branch, and an output layer.

Since texts come in various sizes, to accurately detect and localize texts, low level features in early layers and high level features in later layers are both needed. The low level features are accountable for smaller sized texts and high level features for the texts with larger sizes. The different level of features are extracted during the first stage using a deep convolutional neural network. In the paper[14], Zhou *et al.* implemented the FCN using the PVANet, while ResNet-50 is used in our project since it is a deeper and more powerful network.

In stage 2, different level features are then merged together to generate the desired outputs. Ideally, these features could be combined into a long array, which is then fed through a text detection network. This, however, would result in an extremely high dimensional feature map as there are many channels in each layer of ResNet. Therefore, Zhou *et al.* used the idea of gradually merging from U-Net [9] by starting with the highest level features, unpool the features, merge with lower level features, 1x1 convolution to shrink dimension, 3x3 convolution to fuse information, unpool the merged features, merge with even lower level features, 1x1 and 3x3 convolutions, and continue the process until the very first feature is reached. This method significantly reduces the memory required to train the text detection network without compromising accuracy, as different level of features are still being incorporated.

The last stage is simply the output layer. In this stage outputs such as score map, QUAD vertices, and RBOX coordinates and angles are generated. In our project, we only used the RBOX and score map.

3.2 Algorithm

3.2.1 Style Transfer

Given a content image $\vec{p}$, and a style image $\vec{a}$, we need to synthesis an output style transfer image $\vec{x}$. To solve the optimization problem by using Convolutional Neural Network, we first define the loss function

$$\mathcal{L}_{\text{total}}(\vec{p}, \vec{a}, \vec{x}) = \alpha\mathcal{L}_{\text{content}}(\vec{p}, \vec{x}) + \beta\mathcal{L}_{\text{style}}(\vec{a}, \vec{x}) \tag{1}$$

where α and β are the weighting for content and style representation, respectively. We jointly minimize the content feature distance between $\vec{p}$ and $\vec{x}$, and the style feature distance between $\vec{a}$ and $\vec{x}$.

The content feature is extracted by a specific layer l of CNN which is the l^{th} filter response to the input content image $\vec{p}$. So, the feature map can be denoted by P^l, where P^l_{ij} is the feature value of the i^{th} filter at position j in layer l. Similarly, F^l is the content feature representation of our target output image $\vec{x}$ generate in layer l. We define the squared-error of the two features as our content loss

$$\mathcal{L}_{\text{content}}\left(\vec{p}, \vec{x}, l\right) = \frac{1}{2} \sum_{i,j} \left(F^l_{ij} - P^l_{ij}\right)^2 \tag{2}$$

The style feature of target image $\vec{x}$ is obtain by using a feature space which is built by the correlation of the filter response in each layer. The correlation is given by the Gram matrix G^l, where G^l_{ij} is the inner product of the feature map F^l in layer l

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk} \tag{3}$$

Similarly, A^l is the style correlation feature of our style input image $\vec{a}$ generate in layer l. The mean-squared error in each layer is

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G^l_{ij} - A^l_{ij}\right)^2 \tag{4}$$

where N_l is the number of filters, and M_l is the filter size in layer l. We then define the sum of the mean-squared error in each layer as our style loss

$$\mathcal{L}_{\text{style}}\left(\vec{p}, \vec{a}\right) = \sum_{l=0}^{L} w_l E_l \tag{5}$$

where w_l are weighting of the loss contribution of each layer.

Using standard error back-propagation with gradient descent algorithm, we need to compute the gradient of the loss function. The derivative of the content loss is

$$\frac{\partial \mathcal{L}_{\text{content}}}{\partial F^l_{ij}} = \begin{cases} \left(F^l - P^l\right)_{ij}, & \text{if } F^l_{ij} > 0 \\ 0, & \text{if } F^l_{ij} < 0 \end{cases} \tag{6}$$

The derivative of E_l is

$$\frac{\partial E_l}{\partial F^l_{ij}} = \begin{cases} \frac{1}{N_l^2 M_l^2} \left(\left(F^l\right)^T \left(G^l - A^l\right)\right)_j i, & \text{if} F^l_i j > 0 \\ 0, & \text{if } F^l_{ij} < 0 \end{cases} \tag{7}$$

Finally, we can iteratively update our initially random output image $\vec{x}$

$$\vec{x}^{(n+1)} = \vec{x}^{(n)} - \mu \frac{\partial \mathcal{L}_{total}}{\partial \vec{x}^{(n)}} \tag{8}$$

3.2.2 EAST Text Detection

The total loss needed to be minimized in training of the EAST text detector is

$$\mathcal{L} = \mathcal{L}_s + \lambda_g \mathcal{L}_g \tag{9}$$

where $\mathcal{L}_s$ and $\mathcal{L}_g$ are the losses for score map and geometry respectively. λ_g is the weight used to scale the loss of geometry against the loss of score map.

The loss of the score map, $\mathcal{L}_s$ is the balanced cross-entropy between the predicted score map and the ground truth of the score map. It is defined as

$$\mathcal{L}_s = -\beta \mathbf{Y}^* log \hat{\mathbf{Y}} - (1-\beta)(1-\mathbf{Y}^*) log(1-\hat{\mathbf{Y}}) \tag{10}$$

where β is the balancing factor used to balance the weights between positive and negative samples. β is defined as

$$\beta = 1 - \frac{\sum_{y^* \in \mathbf{Y}^*} y^*}{|\mathbf{Y}^*|} \tag{11}$$

which was first used by Yao *et al.* in [12].

The output used for this project of the EAST text detection is the coordinates of the vertices of a rotated axis-algined bounding box (AABB), RBOX. Therefore, $\mathcal{L}_g$, the loss function for geometry used is $\mathcal{L}_g$, which is defined as

$$\mathcal{L}_{\text{g}} = \mathcal{L}_{\text{AABB}} + \lambda_\theta \mathcal{L}_\theta \tag{12}$$

where λ_θ is the weight to scale the importance of angle loss, $\mathcal{L}_\theta$ against the AABB loss, $\mathcal{L}_{\text{AABB}}$.

The loss for AABB is defined as

$$\mathcal{L}_{\text{AABB}} = -\log IoU(\hat{\mathbf{R}}, \mathbf{R}^*) = -\log \frac{\hat{\mathbf{R}} \cap \mathbf{R}^*}{\hat{\mathbf{R}} \cup \mathbf{R}^*} \tag{13}$$

where $\hat{\mathbf{R}}$ is the predicted geometry of the AABB box while $\mathbf{R}^*$ is the ground truth.

The loss for angle is

$$\lambda_\theta(\hat{\theta}, \theta^*) = 1 - \cos(\hat{\theta} - \theta^*) \tag{14}$$

where $\hat{\theta}$ is the predicted angle and θ^* is the ground truth angle.

4 Implementation Details

4.1 Style Transfer

We adopted the algorithms above and build a feed-forward network using PyTorch. Fine-tune the training parameters to get a proper performance. After updating in 300 iterations, we can obtain our desired style transfer image by given a content image and a style image.

4.2 Text Detection

We have adapted the EAST implementation from a PyTorch reappearance of EAST[1]. We only did minor modifications on parameters. In terms of the usage of output, for each input image we are getting a text file output containing multiple lines of eight numbers indicating the x and y coordinates for the four vertices of the bonding quadrilaterals. In order to easily recover the original texts in the final output image, we implemented a transformation function that calculates the corresponding mask using the vertices information.

4.3 Combined Algorithm

We simultaneously feed our comic content image into the two networks. In text detection, we will generate a mask image that just remain the detected text bounding boxes. In style transfer, we also input a style image which we desire to transfer on the comic image. Then, combining the mask image with text and the transferred comic image as our result.

5 Experimental Setting

5.1 Dataset

The VGG19 was original trained for ILSVRC[10] competition. For this problem, the VGG net was fine tuned by using Common Objects in Context Dataset[2] which contains 118287 images as training dataset. The COCO dataset serves as training data for getting general shape of items.

[1] `https://github.com/Kathrine94/EAST`

[2] `http://cocodataset.org/`

For text detection, the network was trained using the train set of the ICDAR2015 Robust Reading Competition Incidental Scene Text data set for text localization. The dataset contains 1000 labeled images. The labels are the x and y coordinates of each of the four vertices which form the qudrangle box that contains text. The first 500 images were used to train the EAST text detector.

The comic images are subtracted from two dataset. The kaggle dataset: Comic Books Images-Resized Comic Books Images with Train and Test Set [3] The paper, The Amazing Mysteries of the Gutter: Drawing Inferences Between Panels in Comic Book Narratives[5][4], performed a large scale of experiment with comic images. The whole dataset consists of 120G image raw data. For the test image, the images were get from GetComics[5]

5.2 Training Parameters

5.2.1 Style Transfer

The optimizer used to train the network for style transfer is Adam with learning rate of 1e-3.

The weighting of content loss and style loss (α, b) is (1, 1e6).

5.2.2 EAST Text Detection

The optimizer used to train the EAST text detector is Adam with learning rate of 1e-4.

λ_g used in loss function is 1.

$\mathcal{L}_\theta$ used in our project is 20.

The network was trained for 220 epochs. This was picked since it gives the best validation results. Under 220 the network seemed to underfit and over 220 it seemed to overfit.

5.3 Hardware Used

For training convolutional neural networks, the hardware configuration was provided by ITS/Educational Technology Services from University of California, San Diego(UCSD).[6] The configuration as following: GPU: NVida GTX 1080Ti, CPU: 2xE5-2630 v4, RAM: 16GB. For optimization problem, we can simply use laptop which contains intel i5 processor without using GPU.

6 Results

An example of style transfer together with text detection is presented below.

6.1 Text Detection

First, the image is fed through the EAST text detector, the text regions are highlighted with red boxes in Figure 2 (b).

Then, the highlighted areas, i.e. text regions, are being extracted and saved, the regions without text are discarded, resulting in masked image as shown in Figure 2 (c).

6.2 Style Transfer

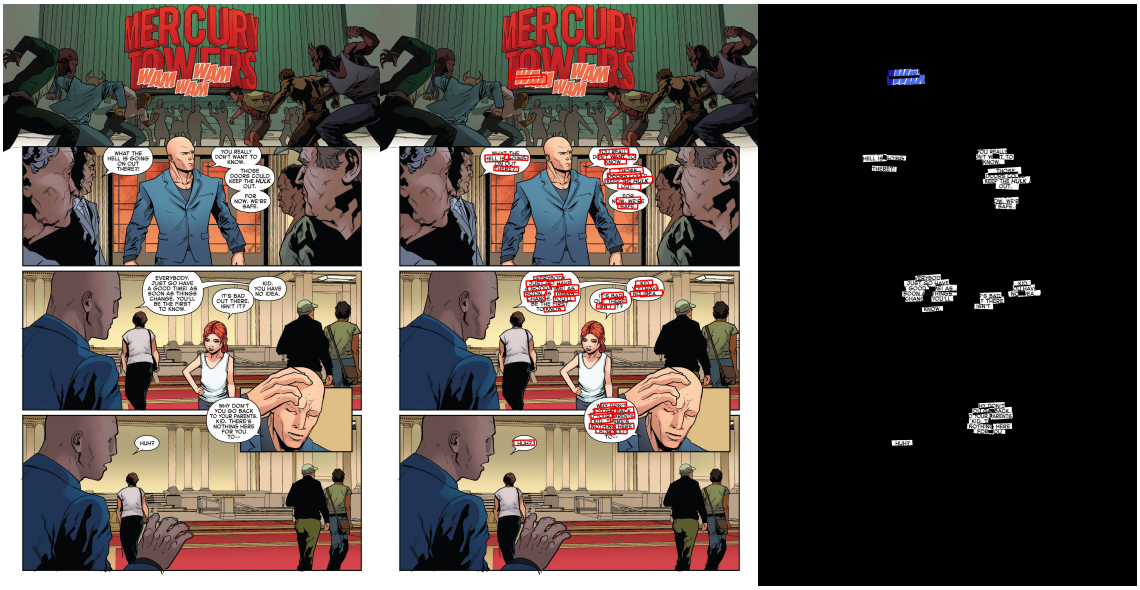
The original image, Figure 2 (a), is then style transferred, with style images Figures 3 (a) and 4 (a) which generates transferred images as shown in Figures 3 (b) and 4 (b) for 2 different styles. As shown in the transferred images, Figures 3 (b) and 4 (b), the styles have transferred successfully but it also causes distorted texts, which make reading difficult. To address this problem, the original texts of Figure 2 (c) are put on top of the transferred images, which are shown in Figures 3 (c) and 4 (c).

[3] https://www.kaggle.com/cenkbircanoglu/comic-books-classification?fbclid=IwAR2RDe_B5py9tuGlPOlWs0ogCbonr_MGcO-g4dPDadI2s1IgzFJALXo9MB0

[4] https://obj.umiacs.umd.edu/comics/index.html

[5] https://getcomics.info/

[6] http://go.ucsd.edu/2CZladZ.



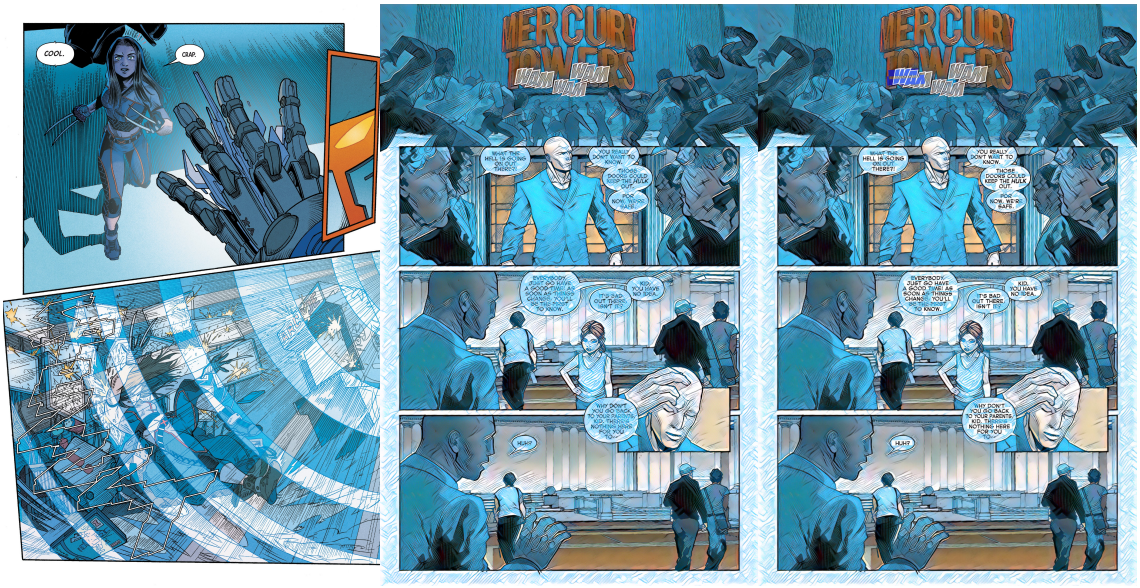
(a) Original comic image (b) Output of EAST text detector (c) Mask of detected text

Figure 2: Original image, text detection output, and corresponding mask

This makes the texts more readable. Now the styles of the comic have changed while the original texts are preserved.

(a) Style image 1 (b) Transferred image 1 (c) Transferred image 1 with original text

Figure 3: Style image and the transferred image for style 1



(a) Style image 2 (b) Transferred image 2 (c) Transferred image 2 with original text

Figure 4: Style image and the transferred image for style 2

(a) Style image 3 (b) Transferred image 3 (c) Transferred image 3 with original text

Figure 5: Style image and the transferred image for style 3 in MSG-Net

6.3 Comparisons

Due to the nature of style transfer, it is difficult to numerically evaluate and compare our algorithm with others. We only manually evaluate the results intuitively by our human observation. We can say that our results indeed outperforms the direct style transfer algorithm in terms of text readability.

7 Discussion

7.1 Inference

We can now can successfully transfer our comic style using the pre-trained model with the method, A Neural Algorithm of Artistic Style. However, it still does not get a perfect texture performance. To improve our result, we try to train our own MSG-Net model, but we only use 9 different drawing styles in our training set. Therefore, if we predict the result using other style images, the performance will be really poor.

7.2 Conclusion

We present that our algorithm of combining the two CNNs can successfully transfer comic images to another style while maintaining the original texts. Although, there are still some defects on the accuracy of text detection as well as the problem of lacking variety of valid comic styles due to the limited time and data we used to train our networks. The experiments show that the proposed way of retaining the original text can indeed improve the text readability in the new styled comic images.

7.3 Learning and Challenges

For this project, we learning how to apply two different neural networks to perform a more complicated task. We also learned how to build up our own docker images and share with team member. Therefore, all the team member can share the same development environment. For the challenging part, the COCO dataset is too big to train the MSG-Net and it took 6 hours to train 4 epochs. Moreover, the text detection problem is hard as EAST algorithm is design to detect general text and not special for text in books.

7.4 Future Work

For future research, we can try to enlarge the data sets, increase the training time and further optimize the training parameters and some optimization choices in order to improve the accuracy of text detection and flexibility of style transfer. On the other hand, due to the text detection method only detects a quadrilateral bonding box for words or lines of texts, there are still some background regions around the texts that are retained in the original style. We would like to solve this and retain only the text parts of the image and smoothly merge it into the new styled images, instead of a hard replacement like we currently do.

8 Appendix

8.1 Github Repo Link

`https://github.com/fraserlai/ECE_285_styletransfer`

8.2 Github Repo Readme

The project focus on how to keep the original text on comic images and perform style transfer on comic images.

8.2.1 Getting Started

The following instrustion is to deploy the system on UCSD ITS computing cluster. The docker image is: `fraserlai/285_project:v1` or `ucsdets/instructional:ets-pytorch-py3-latest`

```
prep ee285f
launch-pytorch-gpu.sh -i fraserlai/285_project:v1  or
launch-py3torch-gpu-cuda9.sh
```

8.2.2 Installing all requirements

Run the following commend in terminal `pip install -r requirements.txt --user`

8.2.3 Running the evaluation code:

First, download the model[7] and put it under `scr/checkpoints_total`. Use the juypyter notebook: `eval_notebook.ipynb` in `src`.

The following are parameters for style image, content image, model weight and final output.

```
style_img_path, content_img_path, style_transfer_output_path,
test_data_path, checkpoint_path, output_dir_box output_dir_txt,
output_dir_pic, mask_path, final_output_path
```

8.2.4 Training the style transfer network code:

Train your own VGG net by using COCO dataset: `Train_StyleTransfer_v2.ipynb`. The dataset should put under `src/dataset/train2017`

8.2.5 Training the EAST text detection code:

Use the juypyter notebook: `text_localization_train.ipynb` in `workspaces/kuanweichen_workspace/285_project`

The trained network parameters will be saved every 20 epochs under `workspaces/kuanweichen_workspace/checkpoints_total`, with name `model_XX` where `XX` being the epoch number.

ACKNOWLEDGMENT

We thank the assist from the instructors in ECE 285 for setting up the GPU cluster and providing the suggestion for the final project.

References

[1] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. J. Wu, and A. Y. Ng. Text detection and character recognition in scene images with unsupervised feature learning. In *2011 International Conference on Document Analysis and Recognition*, pages 440–445, Sept 2011. doi: 10.1109/ICDAR.2011.95.

[2] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[3] W. Huang, Y. Qiao, and X. Tang. Robust scene text detection with convolution neural network induced mser trees. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 497–511, Cham, 2014. Springer International Publishing.

[4] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[5] M. Iyyer, V. Manjunatha, A. Guha, Y. Vyas, J. Boyd-Graber, H. Daumé III, and L. Davis. The amazing mysteries of the gutter: Drawing inferences between panels in comic book narratives. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[6] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *CoRR*, abs/1412.1842, 2014. URL `http://arxiv.org/abs/1412.1842`.

[7] K. Kim, Y. Cheon, S. Hong, B. Roh, and M. Park. PVANET: deep but lightweight neural networks for real-time object detection. *CoRR*, abs/1608.08021, 2016. URL `http://arxiv.org/abs/1608.08021`.

[7] `<https://drive.google.com/file/d/1y0KwerE26j9SnYYLSqqN5ydIgq-_Ln_u/view?usp=sharing>`

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL `http://arxiv.org/abs/1505.04597`.

[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[11] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[12] C. Yao, X. Bai, N. Sang, X. Zhou, S. Zhou, and Z. Cao. Scene text detection via holistic, multi-channel prediction. *CoRR*, abs/1606.09002, 2016. URL `http://arxiv.org/abs/1606.09002`.

[13] H. Zhang and K. Dana. Multi-style generative network for real-time transfer. *arXiv preprint arXiv:1703.06953*, 2017.

[14] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. EAST: an efficient and accurate scene text detector. *CoRR*, abs/1704.03155, 2017. URL `http://arxiv.org/abs/1704.03155`.