P.AI.NT

Julius Fan
University of California - Santa Cruz
juliusfan98@gmail.com
https://github.com/concealedtea

Zike Peng
University of California - Santa Cruz
zpeng4@ucsc.edu

***Abstract*—This document builds upon prior technologies given by Gatys and Ulyanov, who ”characterized the style of an image by the statistics of convolutional neural networks”. Using the base model of Gatys’ texture generation and image stylization, we maximize output quality by testing various techniques and loss functions. In this work, we combat the inaccuracy of visual quality and stylization matching with a different training model. First, we use Ulyanov’s Fast-Style Transfer, but rather than using the perceptual loss function normally attributed to the topic, we use a method developed by Google. Google’s Follow the Regulated Leader loss method allows for the gap in stylization steps to be adjusted between steps, allowing for a more accurate final result. Our primary goals for this project were to develop a reliable image style transfer technology that was comparable to Ulyanov’s texture generation program.**

***Index Terms*—style, large-scale learning, instance normalization, styling**

I. Introduction

The recent developments made by Ulyanov, Vedaldi, and Lempitsky [2], which used feed-forward convolutional neural networks to replace the original optimization process by Gatys *et al.* [9,10,11], has created a resurgence of creativity in the field. The innovations made by Ulyanov, Vedaldi, and Lempitsky [2] suggest that using the feed-forward generator networks, we can eliminate the issue of run-time optimization that was prevalent in Gatys *et al.* [9,10,11] non-linear convolutional neural network filteration model.

Our primary contribution is the implementation of Google's paper on ”Follow the Regulated Leader” to determine our Log Loss with respect to our model's parameters of color and style. We use the

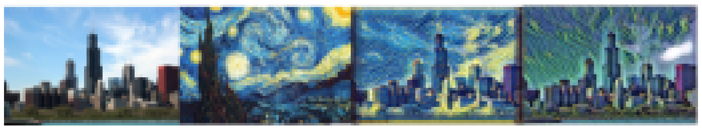


Fig. 1. Example results for style transfer. From left to right, Original Content, Simulated Style, P.AI.NT, Ulyanov *et al.* [2]. In terms of style transfer, we achieve much better color accuracy than Ulyanov *et al.* [2]. P.AI.NT’s stylization combines the color accuracy of Johnson *et al.* [3] and the style transfer properties of Ulyanov *et al.* [2] to produce a more accurate final stylization.

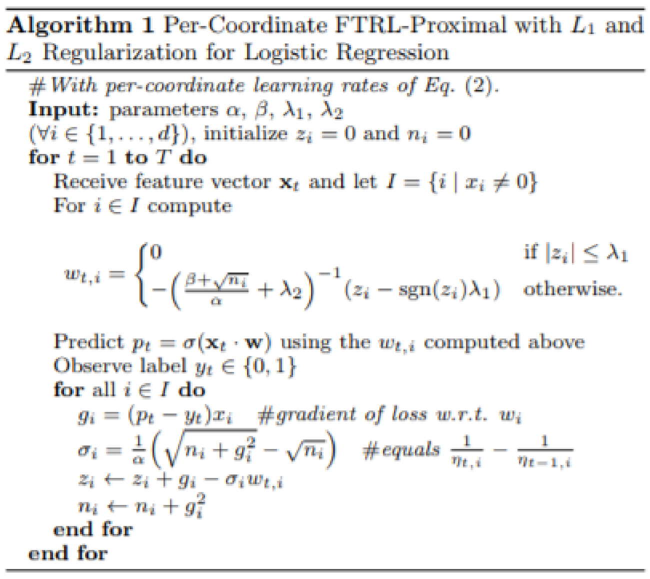
ROC_AUC_Score to determine how well the model was trained. This allows for iterative step training rather than uniform stylization across the board. Instead of using the original generator equation where a network g is learned to minimize the sum of texture loss and content loss:

$$g^\star = \underset{g}{\operatorname{argmin}} \underset{p_{x0},p_z}{\mathrm{E}} [\mathcal{L}(g(x_0, z)) + \alpha\mathcal{L}_{cont.}(g(x_0, z), x_0)]. \tag{1}$$

Instead, we use the equation where we define a learning rate for our loss function, using the above function but defining a variable σ where the FTRL process produces a similar sequence of stylization vectors but reduces sparsity between certain points of unidentifiable color palettes:

$$w_{t+1} = \underset{w}{\operatorname{argmin}}(g_{1:t} \cdot w + 0.5 \sum_{s=1}^{t} \sigma_s ||w - w_s|| + ||w||) \tag{2}$$

Originally, the gradient descent pattern that the feed-forward convolutional neural network used a method of storing multiple variables at a time in order to measure stylization accuracy. Using the FTRL method, we can lower local data storage to only one variable by iterating on the same variable



Algorithm 1 Per-Coordinate FTRL-Proximal with L_1 and L_2 Regularization for Logistic Regression

```
# With per-coordinate learning rates of Eq. (2).
Input: parameters $\alpha$, $\beta$, $\lambda_1$, $\lambda_2$
($\forall i \in \{1, \dots, d\}$), initialize $z_i = 0$ and $n_i = 0$
for $t = 1$ to $T$ do
  Receive feature vector $\mathbf{x}_t$ and let $I = \{i \mid x_i \neq 0\}$
  For $i \in I$ compute
    $w_{t,i} = \begin{cases} 0 & \text{if } |z_i| \leq \lambda_1 \\ -\left(\frac{\beta+\sqrt{n_i}}{\alpha} + \lambda_2\right)^{-1}(z_i - \operatorname{sgn}(z_i)\lambda_1) & \text{otherwise.} \end{cases}$
  Predict $p_t = \sigma(\mathbf{x}_t \cdot \mathbf{w})$ using the $w_{t,i}$ computed above
  Observe label $y_t \in \{0, 1\}$
  for all $i \in I$ do
    $g_i = (p_t - y_t)x_i$   #gradient of loss w.r.t. $w_i$
    $\sigma_i = \frac{1}{\alpha}\left(\sqrt{n_i + g_i^2} - \sqrt{n_i}\right)$   #equals $\frac{1}{\eta_{t,i}} - \frac{1}{\eta_{t-1,i}}$
    $z_i \leftarrow z_i + g_i - \sigma_i w_{t,i}$
    $n_i \leftarrow n_i + g_i^2$
  end for
end for
```

Fig. 2. Algorithm psuedocode model from Google's paper [4].

and changing it between such iterations.

We validate our contributions by means of extensive quantitative and qualitative experiments, including comparing the stylization results with optimization based ones and prior stylization techniques developed by Gatys *et al.*[9,10,11] and Ulyanov *et al.*[2]. We show that combining the FTRL logarithm loss function with feed-forward convolutional neural networks, we can achieve more accurate texture synthesis and image stylization results.

II. Background and Related Work

A. Convolutional Neural Networks

In artificial intelligence, a convolutional neural network (CNN) is a particular class of neural networks that is commonly used to analyze and identify visual patterns. CNN are neural networks that use convolutional filters as neurons. Convolutional filters are matrices, and the convolution represents the calculation that multiplies the data and filter matrix to get the inner product.

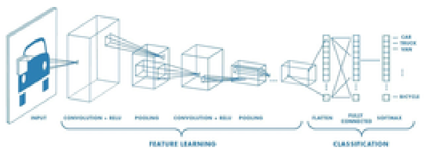


Fig. 3. CNN basic architecture, from Sumit Saha [12]

The main purpose of CNNs is to train those convolvutional filters to make them recognize some specific feature of data (like the color, shape and style of a picture). In order to strengthen the feature and reduce the computation, many have implemented Rectified Linear Units layers and Pooling layers. Compared to other machine learning algorithms, CNNs can help reduce the strict requirements of various parameters, and thus perform better in image and video recognition.

B. Feed-forward image transformation

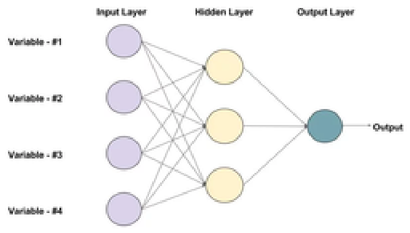


Fig. 4. Example of an Feed-Forward Neural Network with 3 neurons (From OpenCV)

A Feed-Forward neural network is an artificial neural network, however, the nodes cannot form a cycle. Thus, it is different from recurrent neural networks where nodes may be revisted. The Feed-Forward Image Transformation model developed by Ulyanov *et al.*[2] proposes to *learn generator neural networks g(z)* that can map random noise samples z $\sim$ p$_z$ = N(0,I). As such, by learning the neural network *g*, we can minimize the objective using the equation given in the Ulyanov [2] paper:

$$g^\star = \underset{g}{\operatorname{argmin}} \underset{p_z}{\mathrm{E}} \mathcal{L}(g(z)). \tag{3}$$

However, this method has the distinct limitation in the cases where the samples provided by g* were biased and not a good representation of the image style as a whole. As such, Ulyanov[2] developed the generalization method of sum minimization between the style loss and the content loss:

$$g^\star = \underset{g}{\operatorname{argmin}} \underset{p_{x0}, p_z}{\mathrm{E}} [\mathcal{L}(g(x_0, z)) + \alpha \mathcal{L}_{cont.}(g(x_0, z), x_0)]. \tag{4}$$

We continue to use this combination of style loss and content loss in our methods by taking the area expected and the area returned for each individual loss curve. Although this may be considered overfitting to a particular style in the traditional use of Area Under the Curve (AUC) loss methods, it is fine to use here because our sample data is constructed of a singular image rather than multiple different images.

C. Style Transfer

Style transfer has been around since generation-by-minimization stylization prior to Gatys *et al.*[9,10,11]. Our primary style transfer is taken from Ulyanov's CNN's filters using loss values combined from the content image and the style image. By minimizing the combined loss compared to an expected Area Under the Curve $\mathcal{L}$ provided from Ulyanov's paper, we are able to reach a very good estimate of a style from the original image.

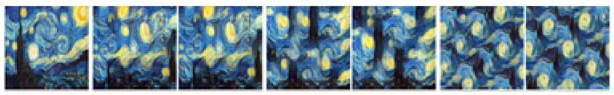


Fig. 5. From Left to Right: Original, TextureNet2 $\lambda = 0$, TextureNet2 $\lambda = 0$, TextureNet2 $\lambda > 0$, TextureNet2 $\lambda > 0$, TextureNet1 $\lambda = 0$, TextureNet1 $\lambda = 0$

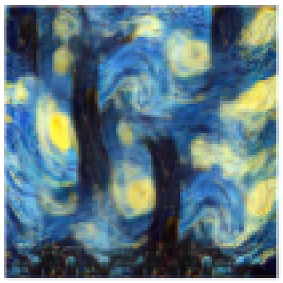
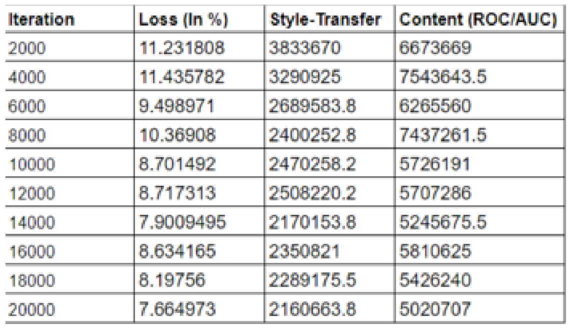


Fig. 6. Our identified image style (20,000 iterations and 7 epochs)

D. Follow the Regulated Leader (FTRL)

The Follow the Regulated Leader (FTRL) technology was originally developed to detect fraudulent advertising click bots. However, since then, many have been able to adjust the algorithm's use to various artificial intelligence purposes. FTRL is based upon a basic logarithmic loss function that has been adjusted by differentiating the step sizes between each iteration in order to find the best step size between items rather than a uniform one across the whole model. By adjusting footstep length in logistic regression, supposedly it becomes both faster and better than logistic regression.



Iteration	Loss (In %)	Style-Transfer	Content (ROC/AUC)
2000	11.231808	3833670	6673669
4000	11.435782	3290925	7543643.5
6000	9.498971	2689583.8	6265560
8000	10.36908	2400252.8	7437261.5
10000	8.701492	2470258.2	5726191
12000	8.717313	2508220.2	5707286
14000	7.9009495	2170153.8	5245675.5
16000	8.634165	2350821	5810625
18000	8.19756	2289175.5	5426240
20000	7.664973	2160663.8	5020707

Fig. 7. Our iteration values from a single epoch

III. Experiments and Results

Through our experiments, we tested using multiple different models and styles. We used various Van Gogh paintings to test the results. The primary ones being Starry Night, Irises, and Sunflowers.

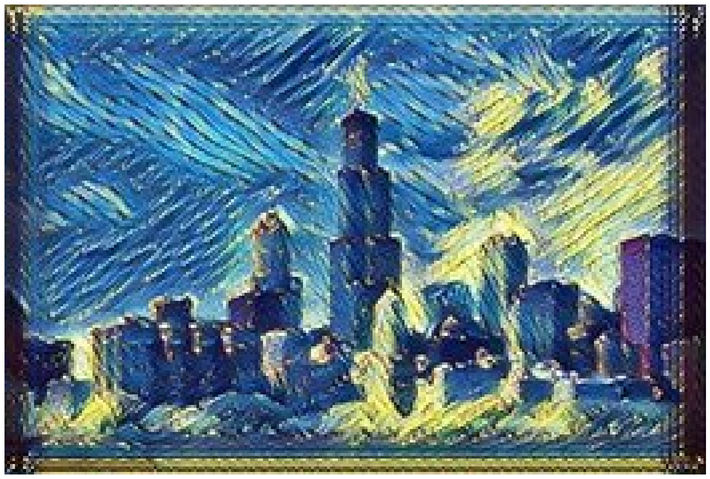


Fig. 8. Chicago skyline with Starry Night style imposed

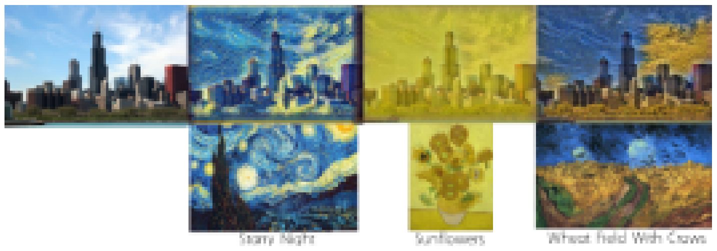


Fig. 9. Various models and results given content image (Chicago Skyline) and style image (From Left to Right: Starry Night, Sunflowers, Wheat Field With Crows)

A. *Code and Technologies we Used*

We used a base code from a code repo (https://github.com/lengstrom/fast-style-transfer) in order to have a base model and built upon that using an additional filter mechanism between the loss functions that rerouted to my personal github's FTRL predictive model (https://github.com/concealedtea/Loss-Models-for-P.AI.NT). We also tested using logarithmic loss functions.

Base Python Packages we Used: Tensorflow, Numpy, SkLearn, DeepCopy, ConfigParser, Scipy, Scipy.Optimize, ArgumentParser.

B. *Qualitative and Quantitative Results*

By using Ulyanov's method as our base comparison method, we recognize that our images may be overfitted in certain cases. This can be easily recognized by the fuzzy logic shown in the edges of prominent visual cues such as the edges of the Willis Tower in the centerpiece of the 3 test styles. We also use our loss function as a measure against Ulyanov's loss function and find that our loss percentages are much higher given the content and styles we tested.

We ran our model on each training data for 10 hours using the baseline images provided from various online sources, using Ulyanov's model (which required 4-6 hours per image) as the base style image we were striving towards. Our loss function and style-transfer functions both show similar curves to Ulyanov *et al.*[2] predicted values given 5000 iterations.

Although originally we assumed that by implementing an alternative loss function of ROC_AUC function rather than the traditional logistic regression method, we would achieve more accurate results.

C. *Our Measurement Fields*

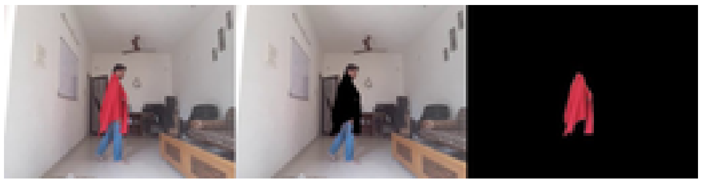


Fig. 10. Example of Python Color Detection Algorithm, by Learn OpenCV (https://www.learnopencv.com/invisibility-cloak-using-color-detection-and-segmentation-with-opencv/)

In order to determine the style's numerical values, we split up each image into a 1920 by 1080 pixel wide matrix and determined each matrix value by using a color identification palette provided by built in packages in Python. For example, when two adjacent pixels share a similar value within 10 RGB values, we don't differentiate between them, however when they are separated by a large value, we can easily tell that there is a stylistic change. Looking back, we could've used a higher definition image without altering the run time too much. This may have resulted in a better final result.

IV. Conclusion

In this paper we have attempted to combine two technologies in order to better the stylization attempts brought forward by Ulyanov *et al.*, however, our models seemed to have significantly worse results than the results given by using Ulyanov's Feed-Foward technologies. Ultimately, we learned how to use Tensorflow to process digital images and gained valuable insights about CNNs, then we tried using FTRL in order to minimize step variance between perceptual loss functions of both the style and content functions. In future work we hope to explore the reasons behind our attempt's underperformance and rectify it in order to better advance the field of stylization.

Acknowledgment

Acknowledgements to Professor Yang Liu, and Teaching Assistants Jiaqi Wu and Can (Molly) Zhang for helping with testing data sets and algorithm clarification. We also want to thank all the great online resources available that made it possible for us to complete this project.

date	**task**	**progress**
01/23/19	Submit a project proposal. (1 page)	Done
02/06/19	Collect the dataset, begin to implement the baseline models.	Done (100%)
02/15/19	Complete the baselines & submit a progress report (3 - 4 pages).	Done (100%)
02/20/19	Implement the stretch models.	Done (100%)
02/27/19	Complete the stretch models & begin the evaluation.	Done (100%)
03/06/19	Finish the evaluation & work on the poster and the final report.	Done (100%)
03/14/19	Poster presentation.	Done (100%)
03/24/19	Submit final project report (5 - 7 pages).	Done (100%)

TABLE I
THE UPDATED MILESTONES.

REFERENCES

[1] Gross, S., Wilber, M.: Training and investigating residual nets. http://torch.ch/blog/2016/02/04/resnets.html (2016)

[2] Ulyanov, D., Vedaldi, A., Lempitsky, V.: Improved Texture Networks: Maximizing Quality and Diversity in Feed-forward Stylization and Texture Synthesis. https://arxiv.org/pdf/1701.02096.pdf (2017)

[3] Johnson, J., Alahi, A., Fei-Fei, L.:Perceptual Losses for Real-Time Style Transfer and Super-Resolution. (2016)

[4] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, Jeremy Kubica.: Ad Click Prediction: a View from the Trenches.(2013)

[5] A. Gretton, K. M. Borgwardt, M. Rasch, B. Scholkopf, and A. J. Smola.: A kernel method for the two-sample-problem.In Advances in neural information processing systems,NIPS,pages 513–520 (2006)

[6] C. Cortes, M. Mohri, M. Riley, and A. Rostamizadeh.: Sample selection bias correction theory. (2008)

[7] Timofte, R., De Smet, V., Van Gool, L.: A+: Adjusted anchored neighborhood regression for fast super-resolution.(2014)

[8] Bevilacqua, M., Roumy, A., Guillemot, C., Alberi-Morel, M.L.: Low-complexity single-image super-resolution based on non-negative neighbor embedding.(2012)

[9] Gatys, L.A., Ecker, A.S., Bethge, M.: Texture synthesis using convolutional neural networks.(2015)

[10] Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576 (2015)

[11] Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: CVPR. (2016)

[12] Saha, S., A Comprehensive Guide to Convolutional Neural Networks, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (2018)