Channel Decomposition into Painting Actions

Shih-Chieh Su*
Microsoft
San Diego, CA 92130
jessysu@gmail.com

Abstract

This work presents a method to decompose a convolution layer of the deep neural network into painting actions. The pre-trained knowledge in the appointed operation layer is used to guide the neural painter. To behave like the human painter, the actions are driven by the cost simulating the hand movement, the paint color change, the stroke shape and the stroking style. To help planning, the Mask R-CNN is applied to detect the object areas and decide the painting order. The proposed painting system introduces a variety of extensions in artistic styles, based on the chosen parameters. Further experiments are performed to evaluate the channel penetration and the channel sensitivity on the strokes.

1 Introduction

For years, the convolutional neural networks have been digesting the visual world and serving a wide range of applications. Neural style transfer is one of the most popular applications. Soon after the pioneering work by Gatys *et al.* [1], the feed-forward network incorporating convolutional layers has been introduced to perform near realtime style transfer [2, 3]. In recent work [4, 5], the stroke and the attention factors have been considered. Although fast, current style transfer work generates the whole transferred frame in one feed. This leaves the audience wondering that, in which stroke order the neural painter would paint the art that can lead to the style transfer output.

On the other hand, the stroke composure process has been studied in the literature. The steps to paint the image (or to write the letters) is typically referred to as stroke-based rendering (or inverse graphics). To learn the painting behavior without pairing stroke-wise training data, reinforcement learning is applied to help stroke planning, in recent work like SPIRAL[6], StrokeNet[7] and LearningToPaint[8]. While the strokes can typically be arranged in the coarse-to-fine order as the painting architecture of the designed, the stroke shape and stroke order may differ from that of the human painter.

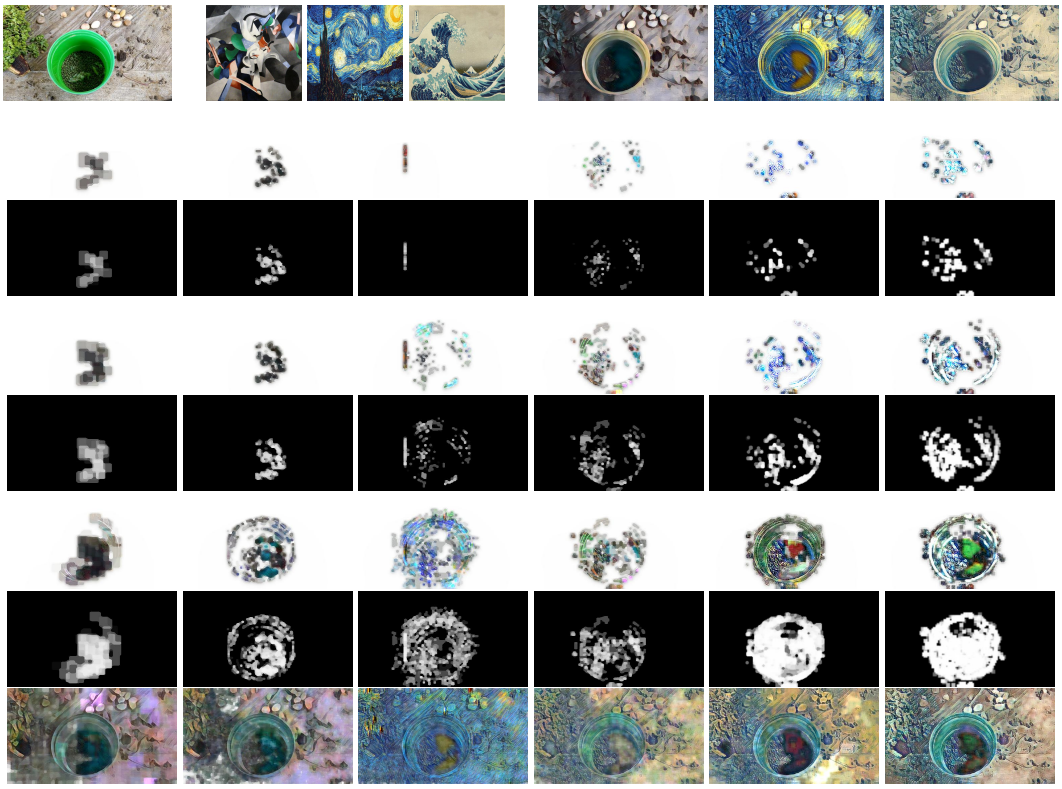
Decomposing the target into reasonable amount of stroking actions of reasonable amount of stroke shapes remains challenging. Which parts of the target needs to be painted, in which kind of artistic style? How does the human painter plan and compose the paint with strokes? For different painter to paint the same object or the same scene, how different would their approaches be? We try to find clues from the generator network.

This work presents a strategy to decompose the channel response of the generator networks into stroke actions, called the *channel stroke*. The channel stroke considers the burden of the human painter in changing paint brushes and changing colors. Leveraging the channel depth of the generator networks, the proposed strategy strokes through the same channels continuously over the regions with high receptive field response.

Experiments are performed over the generator in the GANs and the transformer network of the style transfer. The layer in any of these networks decides the stroke-able space (c, h, w) for the channel

*Demo and code are available at `https://github.com/jessysu/cpia`.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.



(a) big, $l = 9$ (b) small, $l = 9$ (c) small, $l = 9$ (d) small, $l = 9$ (e) small, $l = 8$ (f) small, $l = 7$

Figure 1: CPIA over the style transfer transformer of the MSG[10]: l indicates the operation layer and big/small means the stroke size (see $\mathcal{N}$ in Section 3.2). The top row consists of the content image, the style images (from left to right, *Udnie*, *Starry Night* and *the Great Wave Off Kanagawa*), and original style transfer outputs. The remaining rows are the intermediate outputs, the intermediate stroke maps, and the final outputs. (a) and (b) are styled by *Udnie*. (c) is styled by *Starry Night*. (d-f) are styled by *the Great Wave Off Kanagawa*. MSG has 13 layers where layers 2-9 are the operable bottleneck layers.

stroke. The cost in actions in the stroke-able space then quantifies the burden of the action and makes decision on either continuing stroke, change color, or stop painting. Depending on the learned knowledge in the pre-trained CNN layers, the channel stroking location and style varies. When applied toward the style transfer network, the stroke style varies on top of the neural style (Fig. 1).

Mask R-CNN[9] is used to help the neural painter plan via understanding what it is painting. With the knowledge of recognized objects, the neural painter then focus in painting the object regions one by one. The plan thus covers what to paint, whether the background is painted, and to what detail each region is painted, where the stroke detail is already parameterized in the channel stroke above. The tune-able planning helps to put appropriate focus over different regions. The whole system is called *channel painter in action*, CPIA (Fig. 4).

The paper details the method on channel decomposition and rendering over limited amount of channels. Qualitative results are presented with quantified channel coverage at the operation layer. Based on the existing pre-trained networks, the proposed CPIA provides: 1. stroke composure actions, 2. additional tune-able artistic outcome, 3. controllable brush shape and movement. It is unsupervised and without additional training data.

2 Related Work

Generative neural networks provides the tensor space for channel decomposition. The back-propagation concept of the autoencoders has been introduced in 1980's [11, 12], making the neurons

learn on the errors between the generated target and the real target. The infrastructural improvement on parallel computing later leads to two popular generative approaches: the variational autoencoders (VAEs[13]) and the generative adversarial networks (GANs[14]). The VAEs uses a framework of probabilistic graphical models to generate the output by maximizing the lower bound of the likelihood of the data. While the GANs leverages a discriminative network to judge and improve the output of the generative network. After the adoption of the deep convolutional nets (DCGAN[15]), the task-oriented GANs have been applied to image-to-image translation (Pix2Pix[16], CycleGAN[17], GDWCT[18]), concept-to-image translation (GAWWN[19], PG^2[20], StyleGAN[21]) and text-to-image translation (StackGAN[22], BigGAN[23]), among other domain-specific GANs[24, 25, 26]. In this paper, the BigGAN is used to generate the CPIA painting targets from keywords, as in Fig. 1.

Neural style transfer is one major domain that CPIA can be applied. In the work by Gatys *et al.* [1], the authors formulate the style transfer cost as a combination of the content loss and the style loss. The loss is measured over the pre-trained VGGnet[27], from the generated image to both the content image and the style image. The transformer networks with deep convolutional layers are introduced in [2, 3] to speed up the style transfer - the whole transformer is trained on a particular style. Then comes the transformer attempting to learn multiple styles in one single network, such as [28, 10]. In the following sections, the transformer of MSG[10] is decomposed into the CPIA actions.

Stroke-based rendering, or inverse graphic, without the training stroke sequence is challenging. To deal without the training stroke sequence, a discriminative network guides the distributed reinforcement learners to make meaningful progress in SPIRAL[6]. The computation cost is high for the deep reinforcement learners with large and continuous action space. That can be mitigated by creating a differentiable environment, like the ones in WorldModels[29], PlaNet[30] and StrokeNet[7]. The ongoing research has delved into various stroking agents that generate very different output styles. For cartoon-like stroking, LearningToPaint[8] efficiently generates the simple strokes to compose the complex image. On the other hand, the NeuralPainter[31] abstracts and recreates the image into a sketch-like output.

3 Methods

Let $\Lambda^{(l)}(Y)$ denote the layer operation of the l^{th} layer on its input Y. The generator network of L layers can be expressed as

$$Y^{(l)} = \Lambda^{(l)}\Big(Y^{(l-1)}\Big), \quad \forall l \in \{1, 2, \cdots, L\}, \tag{1}$$

where $Y^{(0)}$ is the input of the network and $Y^{(L)}$ is the output. The operations and corresponding weights in $\Lambda^{(l)}$ were trained with or without the input $Y^{(0)}$.

Extending the forward path of the neural network to allow additional layer operations $\Phi^{(l)}$ leads to

$$Y^{(l)} = \Phi^{(l)}\left(\Lambda^{(l)}\Big(Y^{(l-1)}\Big)\right), \quad \forall l \in \{1, 2, \cdots, L\}, \tag{2}$$

3.1 Channel Flush

Next, we provide implementations of the layer operations $\Phi^{(l)}$. One key finding in our experiments is that, for the intermediate layer images $Y^{(1)} \cdots Y^{(L-1)}$, the decomposed representation preserves the spatial information of the output image $Y^{(L)}$, while the detail at each location can be truncated and represented by the high response channel(s). By only showing τ channels out of the total C channels at layer L, we force the later layers to respond only on the top τ channels at each location of L. This observation shed light on the following operation

$$\begin{aligned}
&\Phi^{(l)}\Big(Y\Big) = M\Big(Y\Big) \odot Y \\
&and \\
&M_{c,h,w}(Y, \tau) = \begin{cases} 1 & \text{if } \Big\| Y_{(c,h,w)} < Y_{(c',h,w)} \Big\| \leq \tau, \forall c' \in \{1, \cdots, C\} \\ 0 & \text{otherwise,} \end{cases}
\end{aligned} \tag{3}$$

(a) raw image generated from [23] (b) channel flush $l = 3, \tau = 512$ (c) channel flush $l = 6, \tau = 512$

(d) channel stroke $l = 4, \tau = 256$ (e) channel stroke $l = 3, \tau = 512$ (f) channel stroke $l = 6, \tau = 512$

Figure 2: Channel decomposition of the BigGAN [23]: (a) is generated using the keyword "seashore". From (a), we use the channel flush in Section 3.1 to generate (b) and (c), and the channel stroke in Section 3.2 to generate (d-f). BigGAN has 15 layers and 14 of them are generative bottleneck blocks.

where $\odot$ is the Hadamard product, $\| \cdot \|$ is the cardinality of a set, and τ is the channel limit out of the C channels at layer l per location h, w. The tensor M masks the original layer output Y and picks only the top τ channels of Y at each location h, w for rendering into later layers.

The channel flush provides a way to focus the image render on high response channels. On any spatial location of the operating layer, the lower response channels are muted. The number of channels to choose from also matters. In CNNs [32, 33], the depth (channels) is traded with the breadth (spatial size). As a consequence, the operation layer of the channel flush is better in the middle of the network - avoiding the last layers which have low channel variety, and the initial layers which have low spatial resolution.

The result of the channel decomposition is shown in Fig. 2. We start with an image generated from the BigGAN[23], then apply the channel flush and the channel stroke, which is described next.

3.2 Channel Stroke

Sometimes the channel flush incurs unnecessary discontinuities over the output image. To deal with this issue, we extend the $\Phi^{(l)}(\,\cdot\,)$ in Eq. 3 into an operation set of channel strokes at layer l. Let (C, H, W) denote the channel depth, height and width of its output $Y^{(l)}$. We define $\mathcal{N}(c, h, w; m) \in \mathbb{N}^3$ as the set of neighborhood pixels (c', h', w') near pixel (c, h, w), where $Y_{c',h',w'} \geq m \cdot Y_{c,h,w}$ for all (c', h', w') in $\mathcal{N}(c, h, w; m)$. The quantifier m is a real number from $(0, 1)$, which means the sensitivity of the stroke. When m is close to one, the stroke sensitivity is high. Thus the channel stroke can only turn on the neighboring pixels with highly similar response as the stroke pixel. The simplest case of $\mathcal{N}(c, h, w; m)$ is a square box centered at (c, h, w) with each side of $2z + 1$ pixels on channel c. In this case, the parameter z is the stroke size.

The channel stroke algorithm (Alg.1) updates the mask tensor M in $\mathbb{Z}_2^{C\times H\times W}$ and the cost tensor G in $\mathbb{R}^{C\times H\times W}$, on each of its iteration. The mask M then filters the layer l to response Y in $\mathbb{R}^{C\times H\times W}$.

The stopping criterion $\mathcal{S}$ terminates the procedure when current response $Y_{c,h,w}$ is lower than a fraction of Y_{max}. Other possible choices for $\mathcal{S}$ include number of strokes and the fraction of painted locations.

Algorithm 1 Channel Stroke

1: **procedure** CHANNELSTROKE($\mathcal{S}, \mathcal{N}, m, \tau$)
2: mask $M \leftarrow 0$
3: cost $G \leftarrow 0$
4: **while** true **do**
5: choose pixel $(c, h, w) \leftarrow \underset{c,h,w}{\operatorname{argmax}}(1 - G) \odot Y$ conditioned on

$$M_{c,h,w} = 0 \text{ and } \sum_{c'=1}^{C} M_{c',h,w} < \tau$$

6: **if** stopping criterion $\mathcal{S}$ met **then**
7: break
8: extend stroke $M_{c',h',w'} \leftarrow 1$ for all (c', h', w') in $\mathcal{N}(c, h, w; m)$
9: update G according to the chosen pixel (c, h, w)

The intuition of channel stroke is that, for a human painter to paint, one needs to select a color of paint, the painting brush, and the pattern to paint. Once these items are selected, the painter can stroke on the canvas and then extend the stroke over a certain region. In Alg.1, the color of paint and pattern to paint are controlled by channel c, which is chosen in Step 5 according the current most responsive pixel (c, h, w). The neighborhood $\mathcal{N}(c, h, w; m)$ decides the stroke shape, which can be related to the painting brush of the human painter.

At the end of each stroke, the human painter can either continue to use the same color to paint other area on the canvas, or switch to other color. To be effective, it is desirable to keep using the brush of current color as much as possible on the same level of painting detail. We factor this behavior into the cost G at the stroke pixel (c, h, w). The channel stroke will continue the stroke into nearby stroke pixel at the same channel. Note that the neighborhood extension in Step 8 is about the shape of the stroke, while the stroke continuation in Step 9 models the behavior of the human painter in switching color.

3.3 Action Cost

We first consider the channel cost, which reflects the cost in changing the brush color for human. Let g_c be the constant cost accounting to changing channel. The channel cost tensor J is

$$J_{c',h',w'}(c) = \begin{cases} g_c & \text{if } c' = c, \text{ where } c \text{ is the current stroking channel} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The next cost factor is the stroke movement. Naturally, the stroke continues into its nearby region. Therefore, the movement cost increases as the distance from the current stroke location increases. The movement cost K is

$$K_{c',h',w'}(h, w; \sigma) = 1 - e^{-\frac{(h'-h)^2+(w'-w)^2}{2\sigma^2}}, \tag{5}$$

where σ is the standard deviation of the Gaussian kernel centered at the current stroking location h, w.

The overall cost is then the Hadamard product of the individual cost components, $G = J \odot K$. This cost is being updated in every channel stroke iteration at Alg.1 Step 9. Further stroke behavior modeling may incorporate location cost for top-to-bottom and left-to-right handwriting behavior. It is also possible to add the directional cost to make the stroke continue in the same direction and attain certain artistic feel. Here we focus on quantifying the burden in between continuing with the current brush or changing color. The cost G provides the next stroking location and channel.

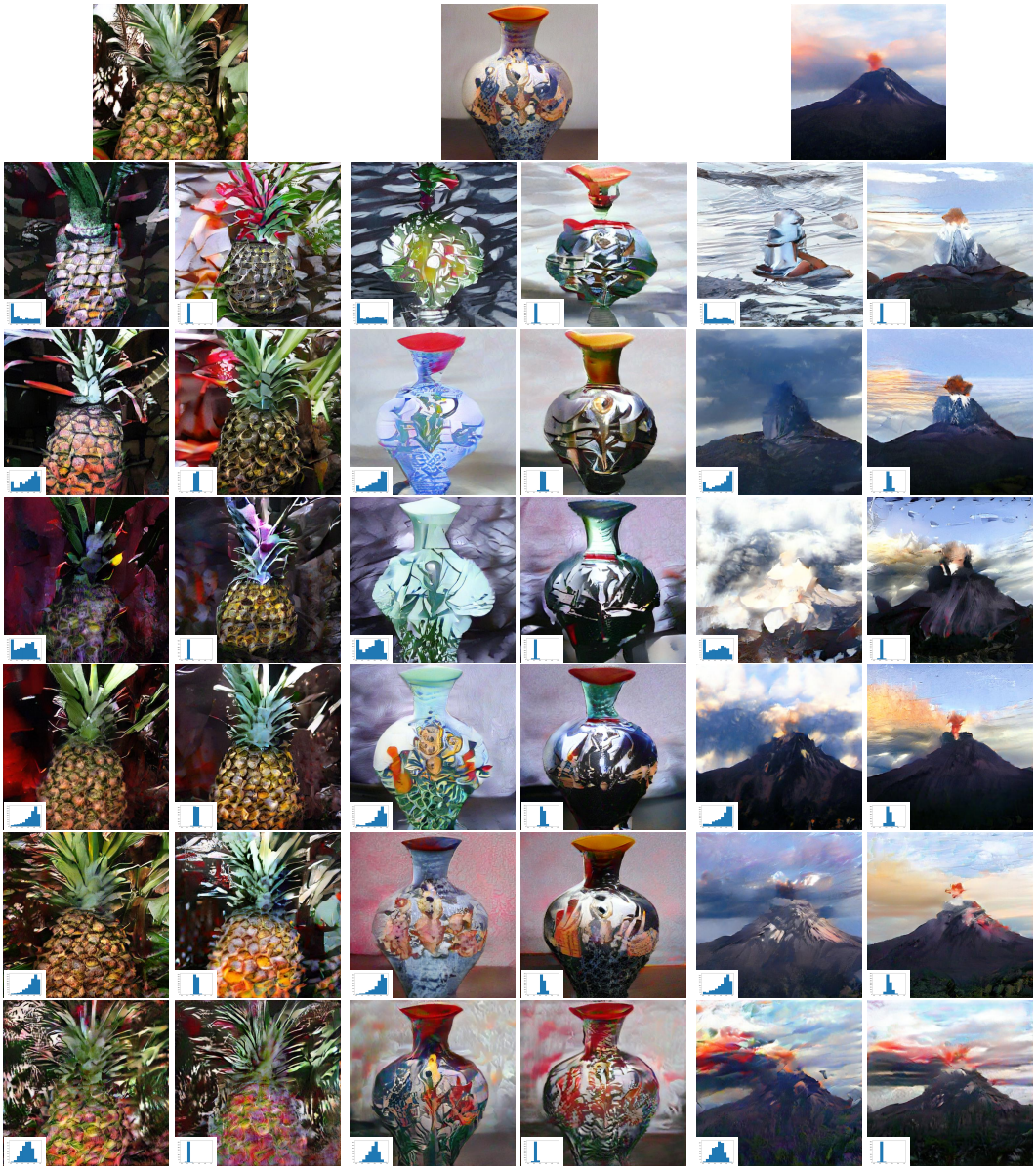


Figure 3: Comparison between the channel stroke (left of pair) and channel flush (right of pair). Histogram of channel coverage floats over each image. From top to bottom, the results in each row are (1) BigGAN [23] from keywords "pineapple", "vase" and "volcano" respectively (2) $l = 4, \tau = 256$ (3) $l = 4, \tau = 512$ (4) $l = 5, \tau = 256$ (5) $l = 5, \tau = 512$ (6) $l = 6, \tau = 512$ (7) $l = 7, \tau = 128$

In Fig.3, we compare the results from channel flush and channel stroke over different operation layers l and different channel limits τ. We also provide a histogram of channel coverage for each outcome image. For an arbitrary channel, the coverage means the fraction of locations having their mask M turned on. The results from the channel flush have more concentrated coverage compared to those from the channel stroke. Because channel stroke extends the stroke into its neighborhood and continues onto the nearby stroke-able region, some channels tend to have higher coverage then others. That causes in the diverged channel coverage.

With neighborhood extension and stroke continuation, the channel stroke overcomes the occasional discontinuity issue in the channel flush, while keeping the artistic look from blocking out the low response channels at each h, w location. When the channel limit τ becomes closer to the number

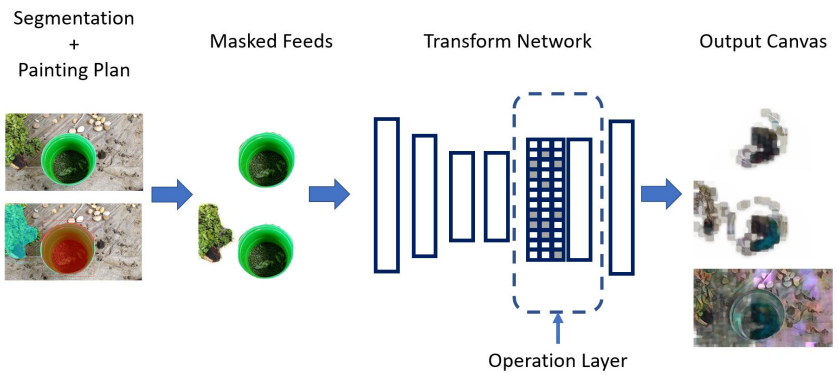
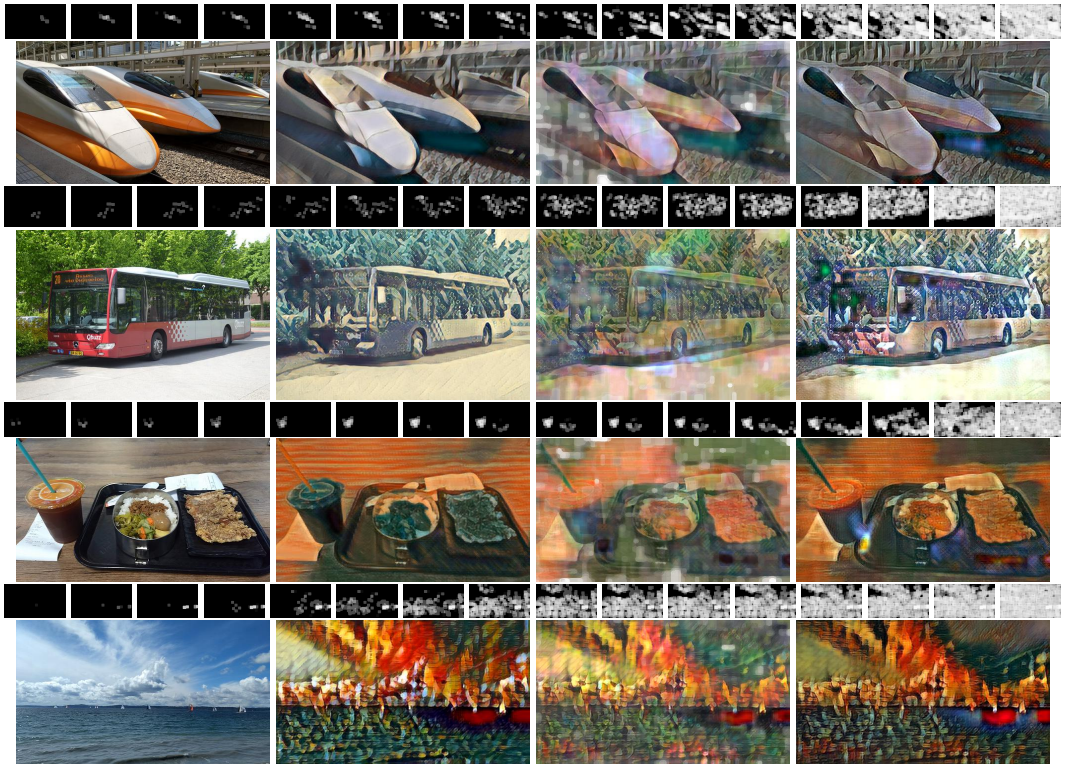


Figure 4: Block diagram for applying the CPIA over generator network



(a) raw (b) styled (c) painted @ $l = 9$ (d) painted @ $l = 7$

Figure 5: CPIA result over the style transfer network in [10]. Images are painted with $\tau = 20$ in (b) and with $\tau = 128$ in (c). Each row of the stroke maps illustrates the painting actions of the (c) image in the following row.

of channels at the operation layer, the output image becomes closer to the original output without channel operations.

3.4 Channel Painting in Action

Based on the channel stroke, we propose the channel painting-in-action (CPIA) framework. This framework first analyzes the input image into painting regions, and come up with a painting plan. The painting plan contains a list of step images, which are composed of certain masked regions of the input image. The step images are then sequentially fed into the pre-trained generator network. The operation layer carries out the stroke actions and paint on the output canvas. The framework is

presented in Fig. 4. In the implementation of this work, we use the Mask R-CNN to mark out the objects as regions of interest (ROIs). The step images are prepared according to these ROIs.

The step images mask out the regions to ensure channel continuity on those regions. When the first step image is fed, the channel stroke algorithm is applied to paint on the masked regions of the current step image, until the stopping criterion $\mathcal{S}$ is met. The output canvas contains only strokes on the regions that have been exposed to channel stroke so far. Then the next step image is fed. The painting process continues until the list of step images is exhausted.

Because of the convolution kernels, the stroked response will propagate into its nearby regions in the next convolutional layer of the CNNs. The cascaded propagation tends to tint the whole canvas at the last layer. Therefore, the stroke maps are used as the post-generator masks to wipe out the response of the non-stroking regions at the last layer.

Note that although there can be as many as τ channels being stroked for each h, w location, the painting process does not guarantee that every location has exactly τ channels stroked. In fact, at the operation layer, it is very likely that the locations with more high response channels have reached the maximum τ channels being stroked. While the less response locations have less than τ channels being stroked.

Fig. 5 presents the CPIA result over the style transfer network in [10]. The original style transfer offers the sharper result in general. On the other hand, the CPIA introduces additional artistic styling components on top of the existing style transfer. Such components are defined by the parameters $\mathcal{S}$, $\mathcal{N}$, m, l and τ. At the same time, the step images in the painting plan can also be seen as a controlling factor that decides the painting priority of each masked region.

4 Discussion and Implementation Details

Working with the stroke penetration parameter, the visually plausible stroke outcome requires the proper stroke mask after the last layer. With low penetration in channels, the response can be very dimmed and need to be masked in low opacity on the background. The mask opacity increases as the location collects more stroked channels.

In current implementations, the painting area ordering is based on the object and its detection score of each ROIs from the Mask R-CNN. We can optionally add side information about the semantics to the planning - such as paint the persons at the end, or paint the large objects first.

To stop the painting at each step image feed of the CPIA, we leverage the stopping criterion $\mathcal{S}$ in Alg.1. $\mathcal{S}$ can be a global threshold cutting of the response ratio at each ROI, or can be the stopping condition tailored toward different object types or sizes. The former is adopted in this work.

5 Conclusion

The proposed channel stroke strategy utilizes the knowledge learned and stored within the deep convolutional generator network. On top of that, the CPIA leverages the learned object and segmentation knowledge in frameworks like Mask R-CNN to plan the painting regions. The CPIA works with the existing generator networks and the existing image segmentation tools, without additional training data on stroking order.

Looking ahead, we seek to drive the stroking factors in a more adaptive way. The stroke size (bundled in the neighborhood $\mathcal{N}$) can vary based on the stroking location. Depending on the stroking channel c, the stroke penetration can possibly change in a responsive fashion. In this work, one single layer of the CNN is chosen as the operation layer. A further expansion is to investigate the multi-layer coordination of the channel stroke. The channel decomposition for the generator networks still has much to explore and the application may be beyond the artistic rendering and the step-wise painting.

References

[1] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 2414–2423, 2016.

[2] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of the European conference on computer vision (ECCV)*, pages 694–711, 2016.

[3] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *International Conference on Machine Learning (ICML)*, volume 1, page 4, 2016.

[4] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *Advances in neural information processing systems (NIPS)*, pages 386–396, 2017.

[5] Yongcheng Jing, Yang Liu, Yezhou Yang, Zunlei Feng, Yizhou Yu, Dacheng Tao, and Mingli Song. Stroke controllable fast style transfer with adaptive receptive fields. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 238–254, 2018.

[6] Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, SM Eslami, and Oriol Vinyals. Synthesizing programs for images using reinforced adversarial learning. *International Conference on Learning Representations (ICLR)*, 2019.

[7] Ningyuan Zheng, Yifan Jiang, and Dingjiang Huang. Strokenet: A neural painting environment. In *International Conference on Learning Representations (ICLR)*, 2019.

[8] Zhewei Huang, Wen Heng, and Shuchang Zhou. Learning to paint with model-based deep reinforcement learning. *arXiv preprint arXiv:1903.04411*, 2019.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 2961–2969, 2017.

[10] Hang Zhang and Kristin Dana. Multi-style generative network for real-time transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[11] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[12] Dana H Ballard. Modular learning in neural networks. In *Proceedings of the sixth National conference on Artificial intelligence-Volume 1*, pages 279–284. AAAI Press, 1987.

[13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems (NIPS)*, pages 2672–2680, 2014.

[15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2016.

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1125–1134, 2017.

[17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 2223–2232, 2017.

[18] Wonwoong Cho, Sungha Choi, David Keetae Park, Inkyu Shin, and Jaegul Choo. Image-to-image translation via group-wise deep whitening-and-coloring transformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10639–10647, 2019.

[19] Scott E Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee. Learning what and where to draw. In *Advances in Neural Information Processing Systems (NIPS)*, pages 217–225, 2016.

[20] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. Pose guided person image generation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 406–416, 2017.

[21] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410, 2019.

[22] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, pages 5907–5915, 2017.

[23] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019.

[24] Yanghua Jin, Jiakai Zhang, Minjun Li, Yingtao Tian, Huachun Zhu, and Zhihao Fang. Towards the automatic anime characters creation with generative adversarial networks. In *Advances in neural information processing systems (NIPS)*, 2017.

[25] Yang Chen, Yu-Kun Lai, and Yong-Jin Liu. Cartoongan: Generative adversarial networks for photo cartoonization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9465–9474, 2018.

[26] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 8798–8807, 2018.

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[28] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *International Conference on Learning Representations (ICLR)*, 2017.

[29] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2450–2462, 2018.

[30] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning (ICML)*, pages 2555–2565, 2019.

[31] Reiichiro Nakano. Neural painters: A learned differentiable constraint for generating brushstroke paintings. *arXiv preprint arXiv:1904.08410*, 2019.

[32] Yann LeCun, Fu Jie Huang, Leon Bottou, et al. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 97–104, 2004.

[33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*, pages 1097–1105, 2012.

A Appendix

In this section, we discuss additional parameters that can futher fine-tune the channel stroke and the CPIA result.

A.1 Stroke Sensitivity

In the channel stroke (Alg. 1), the parameter m is used to qualify which of the neighboring pixels can be turned on during the current channel stroke. Therefore, both the scope of neighborhood $\mathcal{N}$ and the sensitivity m decides the shape of the current stroke. In Fig. 6, different m's are compared over the generated images from the BigGAN [23] network.

Picking a smaller m can further extend the current stroke into its neighboring pixels and create a wider stroke shape. For any pixel at the operation layer, being turned on at one channel reduces the chance of being turned on at other channels. The channel coverage histograms also confirms that smaller m's cause more channels not being turned on. The channels not being turned on are the less responsive channels across the operation layer. Thus the output image from the smaller m's focuses more on the globally responsive channels.

A.2 Stroke Penetration

A stroke typically affects multiple channels. For example, the last layer in the generator consists RGB channels. A stroke in yellow color at least impacts the red and the green channels. Reconsidering the channel strokes, another option is to enable the stroke to penetrate through several channels. The penetration happens when there exists several channels having similarly high response as the stroking channel c at location h, w.

When stroking at the pixel (c, h, w), other $p - 1$ top response channels at the same location are also turned on, conditioned on they are previously muffled. The penetrating stroke then follow Alg.1 to complete the stroking process. The set of the penetrating channels decides the color and the pattern of the stroke. Depending on the

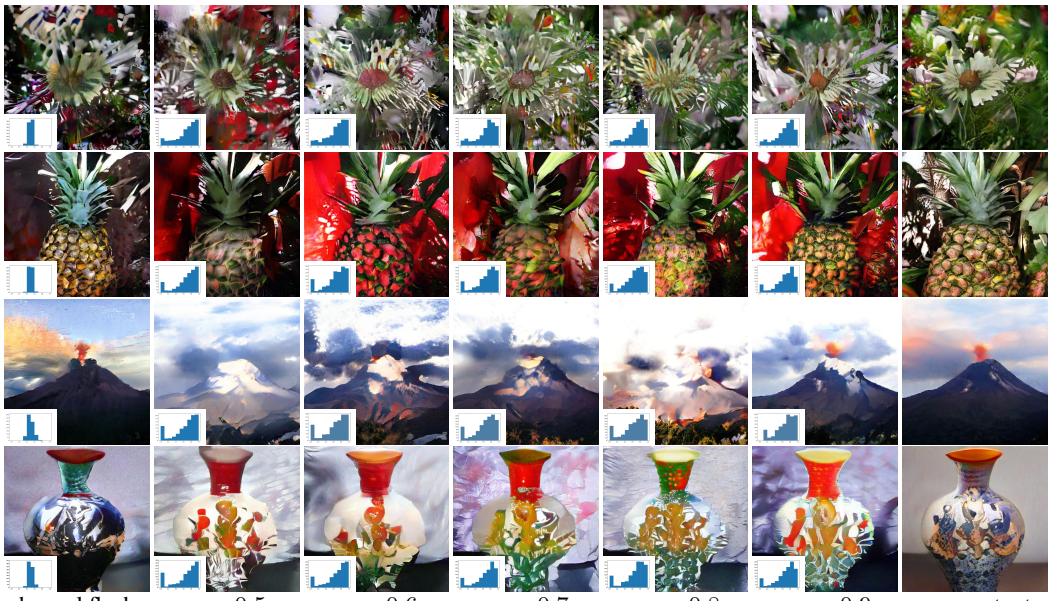


Figure 6: Comparison on various stroke sensitivities m over the BigGAN[23] outputs. The channel operations are on layer $l = 5$ with $\tau = 512$ (out of 1024 available channels).

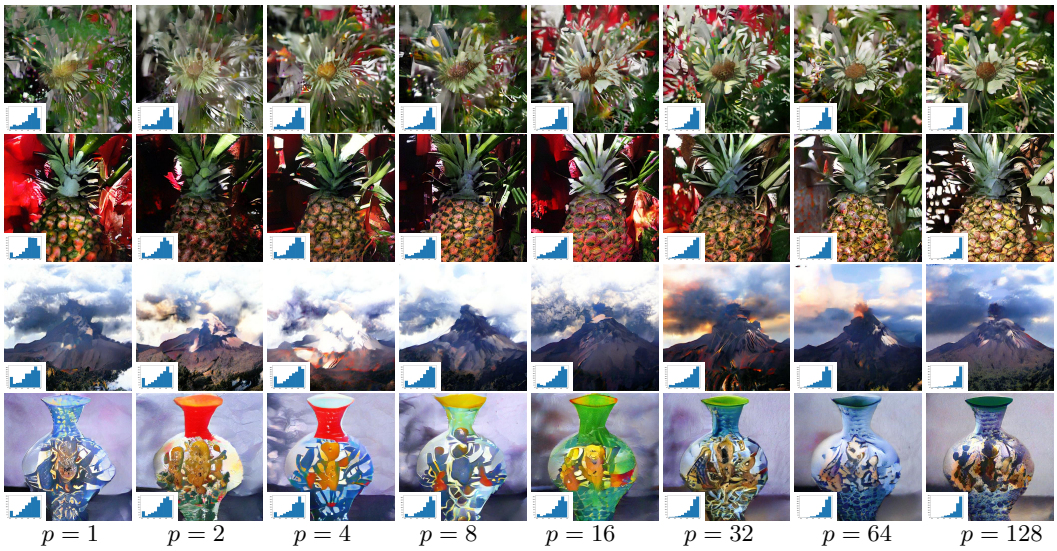


Figure 7: Comparison on various stroke penetration p over the BigGAN[23] outputs. The channel operations are on layer $l = 5$ with $\tau = 512$ (out of 1024 available channels).

operation layer, there can be more than a thousand paint-able channels. The depth of the layer gives room for the stroke penetration. We evaluate the stroke penetration in Fig. 7, where the operation layer has 1024 channels.

The passing channels are less coupled on lower stroke penetration. This induces the variety in channel coverage, and generates high contrast bold results. On the other hand, the higher stroke penetration brings the output closer to the original full channel output. This can be verified with the histograms of channel coverage. While stroke sensitivity m maintains the spatial continuity, the channel continuity is controlled by the stroke penetration p. Note that in Fig. 6 the penetration is fixed at $p = 2$, and in Fig. 7 the sensitivity is fixed at $m = 0.95$.