Quick Snapshots of Cambridge in Artist-Eye

Wenzhao Li[*]
Department of Computer Science
University of Cambridge
wl301@cam.ac.uk

Abstract

Neural style transfer is a computer vision technology that recomposes an image's content in another's style using a neural network. This technology makes it possible that a photograph might appear as if it were created by a famous artist. Nowadays, neural style transfer is extensively used in real-time photo and video editing software. However, this is not achieved overnight. Numerous scholars have explored how to improve the speed and size of the model. In this project, we want to have a taste of how these improvements happen in practice. We select two breakthrough methods in this field to implement - 1. perceptual losses feed-forward transformation network for neural style transfer speed-up and 2. model pruning for compression. We create and train our stylization model - Artist Eye - and explore how the ordinary view of Cambridge would be transformed into various artist styles. When compared with the baseline, our trained and pruned model achieves 118x faster in speed and 53x smaller in size, without obvious visual performance degradation. Besides qualitative comparison, we conduct quantitative analysis by using the state-of-the-art metric ArtFid. This project is a record of our discovery and hands-on practice process in this field. We also include our observations and findings in the Experiments and Evaluation Section.

Project link: click here

1 Introduction

Artists see the world as a place full of possibilities rather than as a set, unchanging paradigm for how to live. They are able to perceive much of what is invisible to the majority of people because they have less limitations on their thinking. Their unique way of interpreting objects makes most ordinary people feel miraculous and amazed. Most of artists live in the same environment and see the same scenery as us, but they seem to have magic - somehow inject their ideas into the artworks and create masterpieces by using various art style.

The neural style transfer technique [1] unravels the mysteries of the artist. With the aid of this technology, a photograph may look to have been produced by a well-known artist. The application of neural style transfer in real-time photo editing software has become widespread in recent years. To simulate the practical use of neural style transfer in daily photo editing, we collect a dataset which contains photos of everyday life in Cambridge and also prepare our target style images. In this application scenario, we select two breakthrough and well-known techniques - model pruning [2] and perceptual losses feed-forward transformation network [3] - to make our model smaller and faster. Our goal is to gain an understanding of how neural style transfer speed-up and model compression work in real practice.

.

[*]King's College, University of Cambridge, CB2 1ST

2 Background

2.1 Style Transfer

The early 2000s saw the beginning of style transfer and texture transfer research. Hertzmann et al. [4] presented image analogies based on a straightforward multi-scale autoregression, whereas Efros and Freeman [5] proposed texture synthesis and transfer via image quilting. An iterative optimization technique for style transfer was put out by Gatys et al. [1] and makes use of the representations learned by convolutional neural networks. A pre-trained convolution neural network (CNN) is used in neural style transfer to transfer styles from one image to another. A loss function is constructed to reduce the discrepancies between generated images, style reference images, and content images. Since each step of the optimization problem necessitates a forward and backward trip over the pre-trained network, their method is computationally expensive. Johnson et al. [3] used a feed-forward network via perceptual loss and were able to speed up the image transform process in a single feed-forward pass. Although this network could produce results instantly, it is limited to the one style that it was trained on.

2.2 Model Pruning

Pruning a model often involves removing from the network any components that give little to no information during inference. Usually, a big initial network is trained to be accurate, and the aim is to create a smaller network with a similar level of accuracy [6].

To increase training speed, several algorithms prune the network during training [7, 8] or even at initialization [9]. Nevertheless, the majority of pruning algorithms adhere to the conventional method suggested by [2, 10] and perform pruning after the usual training procedure.

The literature review work [11] comes to the conclusion that intelligently allocating parameters to distinct layers [12, 2, 13, 14, 15] or global pruning [9, 16] likely to perform better than uniformly pruning all layers. Moreover, when keeping the number of fine-tuning iterations constant, many approaches create pruned models that perform better than initiating with the identical sparsity pattern [17, 18, 19, 16].

2.3 Evaluation of Neural Style Transfer

In the field of neural style transfer, there is always no single correct output. Currently, the most popular approach for evaluating neural style transfer models is a qualitative comparison of style and content images. A qualitative comparison is generally helpful for a first impression, however, the performance of style transfer models varies significantly across various styles and content images. A few works [20, 21] also employ classical perceptual metrics such as PSNR or SSIM [22], however, these metrics are generally not consistent with human perception [23]. Recently, Wright *et al.* [24] proposes a quantitative method by combining two factors: 1. content preservation between generated images X_g and content images X_c and 2.style feature distributions difference between generated images X_g and style images X_s. We adopt this metric for quantitative evaluation.

$$ArtFid(X_g, X_c, X_s) = (1 + \tfrac{1}{N}\textstyle\sum_{i=1}^{N} d(X_c^{(i)}, X_g^{(i)})) \cdot (1 + FID(X_s, X_g))$$

3 Methodology

As mentioned earlier, this project is to experience how model speed and size have been improved in this field. So we choose the basic and first algorithm in Neural Style Transfer [1] as the baseline. To witness the effect of model speed-up, we select the perceptual losses feed-forward transformation network [3] to implement. In addition, we choose one breakthrough paper [2] about model pruning and apply it to the speed-up model, to get a feel of size reduction. Ultimately, we apply this model in our proposed application scenario - real-life photo style transfer - and measure the changes in speed and size. It is worth mentioning that we did not include the most cutting-edge technology in this project. Because as beginners in this field, the first thing we want to do is to experience the basic principle and algorithms, to figure out the key development and improvement of this field.

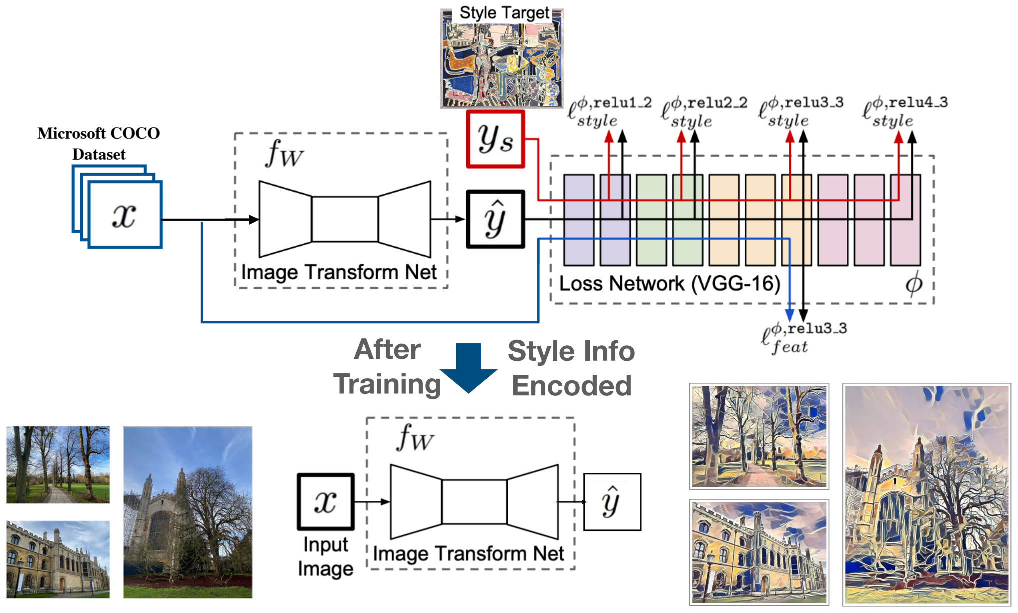

Figure 1: Our Artist-Eye Net based on the framework of [3].

3.1 Baseline

We use the work of Gatys *et al.* [1] as our baseline. y_c and y_s are input content and style image. The network ϕ is a pre-trained VGG-16 [25] on ImageNet [26]. We use a pre-trained VGG-16 instead of VGG-19 for consistent with the network structure used in [3]. The style image y_s is passed through the network and its style representations on each layer j are stored. The content representation of content image y_c is calculated on one layer i and is also stored. The image y is initialized with white noise. y are updated iteratively until it can simultaneously match the style features of the y_s and the content feature of y_c. An output image $\hat{y}$ evolves from y by minimizing the total loss in two aspects - content loss $\lambda_c \ell_{content}^{\phi,i}(y, y_c)$ and style loss $\lambda_s \ell_{style}^{\phi,j\in J}(y, y_s)$. i and j represent different network layers. λ_c and λ_s are scalars which represent weights for different loss items.

$$\hat{y} = arg\min_{y} \lambda_c \ell_{content}^{\phi,i}(y, y_c) + \lambda_s \ell_{style}^{\phi,j\in J}(y, y_s)$$

Each iteration (the updation of y) requires a forward and backward pass through the VGG-16. A forward pass is needed to calculate the style and content loss, and then a backward pass is needed to calculate the total loss gradient - the loss derivative with respect to the pixels value - to update the image y's pixels. This makes the process slow. Thus, it is difficult to implement in production due to these time and computation constraints.

3.2 Artist-Eye Net

Our Artist-Eye Net is mainly based on the framework proposed by Johnson *et al.* [3]. This utilised a single feed-forward pass of the image transformation process. The limitation is that it is only capable of producing in the one style that it was trained on. But in our application scenario, this is acceptable. After taking pictures, users prefer to use the style templates that come with the photo editing software directly. Compared with the slow image generation speed, this limitation does not affect the user experience too much. We need to train multiple networks, each corresponding to a style template. Figure 1 shows how it works. The concept is to train an image transformation network to transform input images into output images in sigle feed-forward pass. The loss function of this image transferm net is also a deep convolutional network - a pretrained VGG-16 same as [1]. The differences between certain layers in the loss network can help to measure the loss in content and style. The output image $\hat{y}$ is measured by its content loss with the content image x (also act as input image), and its style loss with the style target image y_s.

Algorithm 1 Our Stepwise Global Pruning and Fine-tuning Strategy

```
Input: target sparsity $s_{target}$ and dataset $X$
  $W \leftarrow initialize()$
  $W \leftarrow trainToConvergence(f(X; W))$
  $M \leftarrow 1^{|W|}$
  repeat
      $M \leftarrow globalPrune(M, score(W), \Delta s_i)$
      $W \leftarrow fineTune(f(X; M \bigodot W))$
  until $s_{current} >= s_{target}$
  return $M, W$
```

3.3 Stepwise Global Pruning and Fine-tuning

We follow iterative pruning and fine-tuning schema in [2]. Based on the knowledge mentioned in Background Section, we select global pruning strategy. As shown in Algorithm 1, our Artist-Eye Net is first trained to convergence. The trained network can be described as a particular parameterization of an architecture $f(X; W)$, where X is the training dataset and W is model parameters. The pruning process is to produce a pruned model $f(X; M \bigodot W)$ which still accords with the characteristic of X. M is a binary mask where $M \in \{0, 1\}^{W'}$ to achieve the pruning effect. $\bigodot$ is the dot product operation that applies M on W to make certain parameters equal to 0.

Δs_i is the sparsity increment. In each iteration, a global pruning process $globalPrune$ runs first with a given sparsity requirement Δs_i. The network is pruned based on $score(W)$ - the evaluation of each parameter or element's importance. A further training - known as fine-tuning $fineTune()$ - is needed to recover the pruned network's accuracy. The iteration process gradually reduces the network's size (increases the network's sparsity) and will be terminated once the model's current sparsity $s_{current}$ reaches the target sparsity s_{target}. Δs_i may be set with different value or keep as constant at each iteration.

4 Experiments and Evaluation

4.1 Datasets

ImageNet dataset[26] ImageNet dataset spans 1K object classes and contains 1,281,167 training images, 50K validation images and 100K test images. The ImageNet Dataset is used to - 1. pre-train the 16-layer VGG network in the baseline. 2. pre-train the loss network (16-layer VGG network) in Artist-Eye Net.

MS-COCO 2014 dataset [27] The MS COCO dataset is a large-scale object detection, segmentation, key-point detection, and captioning dataset. MS COCO 2014 version contains 164K images split into training (83K), validation (41K) and test (41K) sets. The training set (83K images) is used to train our Artist-Eye Net as content images. This can help our model to learn how to consistently represent daily common objects in a given art style.

Artworks dataset [24] Artworks dataset is a dataset where artworks are labelled by artists and stylistic periods. Due to the relatively long time required to train each Artist-Eye Net (about two hours running time on ampere GPU provided by the Cambridge CS department), we only selected 6 paintings from this database. Although the quantity is small, the styles of these pictures are very different. Some of them are abstract styles and some are realistic styles. Some paintings prefer to use dots as the basic elements, while some paintings mainly consist of rough lines. With this dataset, we hope to test the generality of the model for learning different styles. These painting images act as style inputs for - 1. the baseline network and 2. the Artist-Eye Net, where each painting is used to train a unique Artist-Eye Net, resulting six Artist-Eye Nets with different style target.

Cambridge-Life dataset (self-collected) This dataset has 110 images, all of which were taken in Cambridge. The scenes in the pictures involve modern buildings, old colleges and churches, trees and lawns, dining halls and classrooms, and even common animals (e.g. ducks and college cows). The reason why we create our dataset is that we want to test the generality and practicality of Artist-Eye Net in real-life scenarios.

Table 1: Comparison of models' average size, speed and visual results quality (Art-FID)

	GPU (ms)	CPU (s)	zip size	Art-FID ↓
Baseline [1]	42,978.99	2,780.44	45.76 MB	37.11
Artist-Eye (w/o pruned) [3]	412.78	1.72	6 MB	38.10
Iterative global pruned w/ tuned Artist-Eye	361.66	2.01	881.00 KB	40.96
Global pruned w/ tuned Artist-Eye	374.11	2.02	898.33 KB	43.61
Global pruned w/o tuned Artist-Eye Net	354.98	2.03	896.34 KB	49.29

4.2 Implementation and Training Details

Baseline
We follow the implementation of the basic Neural Style Transform algorithm [1] by Gatys *et al.* [1] and build a colab version of it for better visualization. We use the pre-trained VGG-16 (total params: 12,944,960) on ImageNet dataset [26] to extract the features of the content and style image at various layers. The optimization is performed using L-BFGS. We set the number of iterations as 500 because in most cases optimization converges to satisfactory results within 500 iterations. We implement the training and stylization process on Google Colab.

Artist-Eye Net
We adopt the perceptual loss image transformation network proposed by Jognson *et al.* [3] (total params: 1,679,235). Our implementation follows the suggestions by [28] and use instance normalization and nearest-neighbor up-sampling. With a certain style image, each style transfer network (per style target) is trained on the Microsoft COCO dataset [27] with 83k training images. Since our Artworks dataset has six artworks, we will train each network separately. Finally, we obtain six trained networks corresponding to different style templates. Same as the settings in [3], we use Adam as an optimizer. The batch size is set as 4 for 40,000 iterations, and the epoch number is 2. Training takes roughly 3 hours (per model) on ampere GPU compute nodes provided by the Cambridge CSD3. Once models are trained, we conduct the stylization and visualisation process on Google Colab.

Pruned + Fine-Tuned Artist-Eye Net
We use the PyTorch library **torch.nn.utils.prune** to implement the pruning and fine-tuning process. Each pre-trained Artist-Eye Net (per style target) is pruned respectively. Δs_i is set with a value of 0.1 and s_{target} is 0.95. We use a global pruning strategy to prune the weights in down-sampling convolution layers, residual layers as well as up-sampling convolution layers.

4.3 Quantitative Results

Table 1 shows the quantitative results of the baseline, unpruned Artist-Eye Net, and pruned Artist-Eye Nets. Three pruned Artist-Eye Net are generated by using different pruning strategies - 1. iterative global pruned w/ tuned, 2. global pruned w/ tuned, 3. global pruned w/o tuned.

Both GPU and CPU time cost measurements for the stylization process are conducted on Goole colab. The GPU environment is NVIDIA-SMI 460.32.03, Driver Version: 460.32.03 and CUDA Version: 11.2. and CPU environment is Intel(R) Xeon(R) CPU @ 2.20GHz.

The result shows that the method - perceptual losses feed-forward transformation network [3] - can quickly approximate solutions to their optimation problem. We observe the run-time is 118x speedup in GPU and 1390x speedup on CPU. The baseline method is slow because each L-BFGS iteration requires a forward and backward pass through the VGG-16 network. The reason why there is no significant run-time difference between pruned and unpruned Artist-Eye is that we haven't adopt the beta version torch.SPARSE in this project [will do it as future work]. Pytorch doesn't officially support sparsity operations yet. It currently cannot automatically detect and optimize sparse matrix operation.

We observe that the effectiveness of model pruning is also significant. After pruning with 95% sparsity target, the pruned model size shrinks 53x scale, which is less than 1MB. The three pruned models (with 95% sparsity) have similar compressed sizes.

[1]https://github.com/gordicaleksa/pytorch-neural-style-transfer

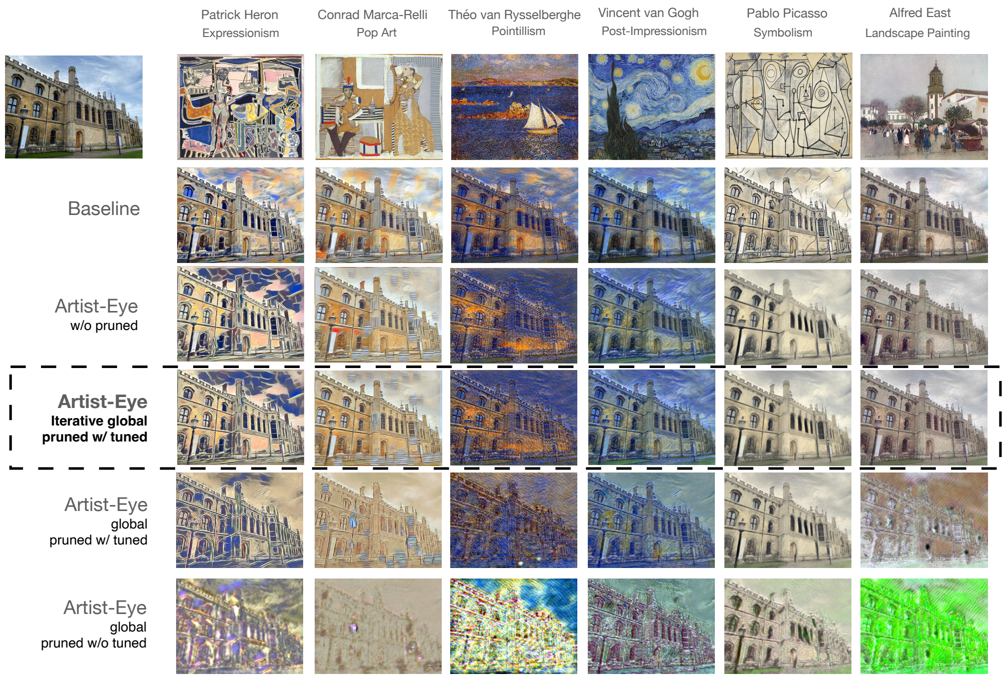

Figure 2: Qualitative results on Cambridge-Life dataset with different style targets. All pruning methods share the same pruning sparsity target, where s_{target} = 95%. More experiment results can be found here: click here

We adopt the state-of-the-art metric ArtFid [24] to evaluate our models as explained in Background Section. It combines the measurement in both content and style aspects. Only models that excel at both style matching and content preservation are given a low ArtFID score. The result shows that the speed-up and compression process will slightly degrade visual result quality. The result also shows that the iterative global pruned and tuned model achieves better Art-FID than the other two pruning strategies. Global pruned without tuned model has the lowest Art-FID.

4.4 Qualitative Results

Figure 2 demonstrates the visualization of the generated images on the Cambridge-Life dataset. We divide the analysis into two steps. First, we compare the difference between **Artist-Eye (w/o pruned)** and the **baseline**, because the speed-up Artist-Eye (w/o pruned) changes the design and structure of the neural network. This comparison allows us to understand the impact of structural change on visual outcomes. Afterwards, we compare the differences between **pruned models** and **Artist-Eye (w/o pruned)**. In this way, we can better feel the impact of various pruning strategies.

Artist-Eye (w/o pruned) vs. **Baseline**:
Overall, the results from Artist-Eye (w/o pruned) are qualitatively similar to the baseline. It shows that in general, our Artist-Eye model successfully transfers the target style information on content images. However, there are some slight differences. When trying to transfer the style of Patrick Heron (Expressionism), the way Artist-Eye (w/o pruned) rendering the sky is different from the baseline. It prefers to divide the sky into small pieces, and then paint in the colour. Furthermore, when transferring Picasso's style, Artist-Eye (w/o pruned) also favours a block-by-block repaint of the entire building. One of the possible reasons is that Artist-Eye (w/o pruned) is trained heavily by MS-COCO dataset to match the style. So it is more inclined to classify and separate the picture's content information.

Artist-Eye (iterative global pruned w/ tuned) vs. **Artist-Eye (w/o pruned)**:
Although the sparsity of Artist-Eye (Iterative global pruned w/ tuned) reaches 95%, the images

Table 2: Model Pruning Comparison - The Cambridge-Life dataset results on six style artworks - **ArtFID** $\downarrow$, **ContentDist** $\downarrow$, **StyleFID** $\downarrow$. Interesting findings are in red and blue.

		P. Heron	**C. Marca-Rell**	**T. van Rysselberghe**	**V. van Gogh**	**P. Picasso**	**A. East**	**Overall**
	Expressionism	Pop Art	Pointillism	Post-Impressionismh	Symbolism	Landscape Painting		
ArtFID	Artist-Eye (w/o pruned) [3]	28.57	29.86	31.15	43.69	57.29	38.04	38.1
ArtFID	**Iterative global pruned w/ tuned**	34.14	34.69	31.76	49.66	**55.12**	40.40	40.96
ContentDist	Artist-Eye (w/o pruned) [3]	1.56	1.52	1.77	1.43	1.44	1.44	1.53
ContentDist	**Iterative global pruned w/ tuned**	**1.52**	**1.47**	**1.71**	**1.38**	**1.33**	**1.36**	**1.46**
StyleFID	Artist-Eye (w/o pruned) [3]	18.37	19.68	17.62	30.49	39.69	26.40	25.38
StyleFID	**Iterative global pruned w/ tuned**	22.52	23.64	18.59	36.09	41.42	29.79	28.68

generated from it are very similar to the Artist-Eye (w/o pruned). This shows that our compression results are relatively good, effectively removing redundant information in the network.

Artist-Eye (global pruned w/ tuned) vs. **Artist-Eye (iterative global pruned w/ tuned)**:
The comparison results show that even Artist-Eye (global pruned w/ tuned) and Artist-Eye (Iterative global pruned w/ tuned) have the same sparsity of 95%, the result of Artist-Eye (Iterative global pruned w/ tuned) is much better than Artist-Eye (global pruned w/ tuned). It helps to prove that learning the right connections is an iterative process [2]. Iterative pruning and tuning give the best result.

Artist-Eye (global pruned w/o tuned):
The results from Artist-Eye (global pruned w/o tuned) with sparsity 95% are more distorted. One interesting finding is that: the output images of six pruned Artist-Eye Nets (each is trained with a certain style target separately) are very different. For example, the outcome in the style of Pablo Picasso's Symbolism is still acceptable. However, the results in columns 3 and 6 contain strange visual colours. One possible explanation is that, although the six networks (each corresponding to a style) are based on the same structure, different target style inputs will affect the distribution of the trained weight in the network, the degrees of redundancy in these networks are also different.

4.5 Further Exploration

The influence of iteration pruning and fine-tuning on content and style similarity
Considering that ArtFid is composed of content similarity measurement ContentDist and style similarity measurement StyleFID, we list their their values in Table 2. An interesting finding is that, the ContentDist value is decreased after the iterative pruning and tuning process. This indicates that the content similarity between the stylized image and the content image increases. However, the style difference increases. One of the possible reasons is that texture information is dependent on the correlation of features in many layers. That is why in [1], style loss is calculated through multiple layers. Pruning reduces the parameters of the network, which reduces the storage dimension of style information. Because of the reduction in texture information encoded in the transform net, the degree of input content image transformation is reduced. The content of the output image will be more like the input image. This will lead to less content distance.

Another interesting finding is that the iterative pruning and tuning process sometimes can produce a bit higher ArtFID scores than the unpruned network. We do not have a very confident explanation for this. One possible explanation is that the inconsistency between human judgements and machine evaluation metrics is inevitable. The ArtFid score can only roughly judge the quality of the picture, but its algorithm cannot accurately simulate the human brain's understanding of style and content.

The trade-off visualization of pruning
Figure 3 illustrates the trade-off for different pruning sparsity target s_{target}. With $s_{target} = 0.5$, as mentioned in [2], the free-lunch phenomenon occurs. It's noteworthy to note that even without retraining, we can cut connections by 2 without losing accuracy. This indicates that the neural network is indeed redundant. This figure also shows that, without retraining, accuracy declines significantly more quickly. While retraining allows us to reduce connection by 8x-9x. The optimum pruning method is iterative pruning. Further exploration on this phenomenon may help us design a smarter pruning strategy and reasonable design pruning pace. For example, is it possible to pruning 50% first and then gradually tune. Or, can the step-size of each iteration be gradually decreased?

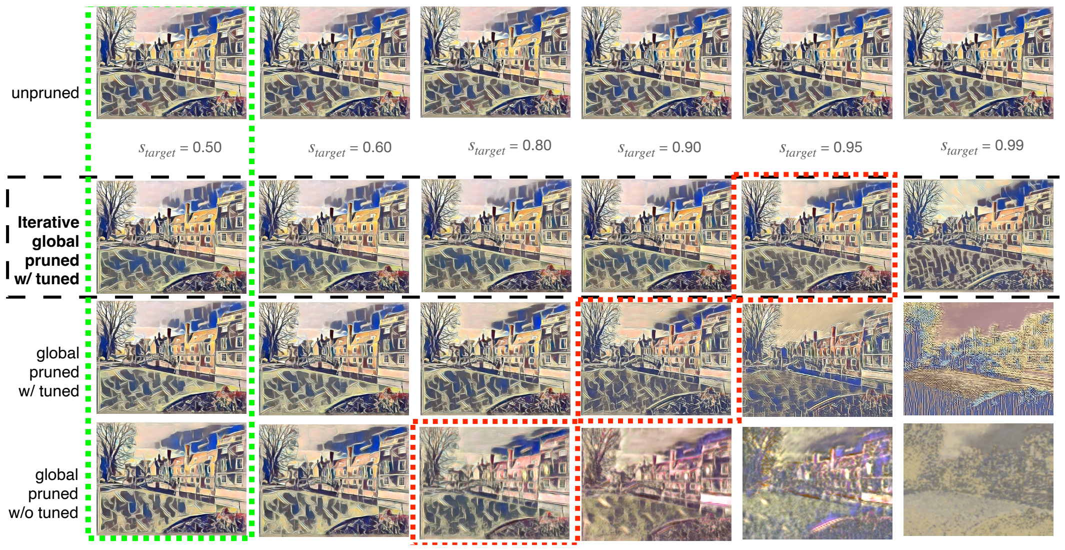

Figure 3: Trade-off visualization with different pruning sparsity target.

4.6 Discussion of Limitations

a) Due to time constraints, we did not adjust any model hyperparameters for any of the models we built. The model can be fine-tuned to perform better. b) The beta version of torch SPARSE has not been implemented in this project. Therefore, the running speed after model pruning has not been significantly improved. c) The images in our database are only the scenery around Cambridge, and may not fully measure the processing effects of different models on other scenery (for example, desert, sea, mountains)

5 Conclusion and Future Directions

In this project, we design an application scenario - an artist-style edition of real-life photos. We pick two techniques. 1. perceptual losses feed-forward transformation network 2. model pruning, and to test their effectiveness in speed acceleration and model compression. Compared to the baseline, our trained and pruned Artist Eye models - achieves 118x quicker speed and 53x smaller size without obviously degrading visual performance. We got a real taste of the powerful effects of these two technologies, and the efforts researchers have made to evolve the technology. As a simple application for photo style transfer, our Artist Eye models also provide us with lots of nice and fantastic stylized pictures [shown in Appendix].

In application aspect, one direct extension of this project is to accomplish video style transfer [29] by feeding each frame into our model. But we need to consider how to achieve consistency of the stylized content between frames. It is also interesting to explore the possibility of how neural style transfer model can be embedded into wearable electronic glasses [30]. This can help to render an immersive virtual reality of abstract art or cartoon world [31].

In the research aspect, we will finish our incomplete implementation of knowledge distillation on neural style transfer [32, 33] [we've started some experiments but haven't finished adding this technology to this project framework due to time constraints]. It is also worth to further exploring pruning strategies and pruning sensitivity analysis [34, 35]. Another interesting direction is to study visualization and interactive pruning [36] and to investigate how it is related to AI explanation.

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*, pages 2414–2423, 2016.

[2] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[3] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[4] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340, 2001.

[5] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, 2001.

[6] Song Han. *Efficient methods and hardware for deep learning*. PhD thesis, Stanford University, 2017.

[7] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

[8] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

[9] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

[10] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[11] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

[12] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

[13] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[14] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

[15] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[16] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[17] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):1943–1955, 2015.

[18] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9194–9203, 2018.

[19] Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. *Advances in neural information processing systems*, 30, 2017.

[20] Xueting Liu, Wenliang Wu, Huisi Wu, and Zhenkun Wen. Deep style transfer for line drawings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 353–361, 2021.

[21] Kibeom Hong, Seogkyu Jeon, Huan Yang, Jianlong Fu, and Hyeran Byun. Domain-aware universal style transfer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14609–14617, 2021.

[22] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[23] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[24] Matthias Wright and Björn Ommer. Artfid: Quantitative evaluation of neural style transfer. In *DAGM German Conference on Pattern Recognition*, pages 560–576. Springer, 2022.

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[27] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[28] Aleksa Gordić. pytorch-nst-feedforward. https://github.com/gordicaleksa/pytorch-nst-feedforward, 2020.

[29] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos and spherical images. *International Journal of Computer Vision*, 126(11):1199–1219, 2018.

[30] Daiki Taniguchi. Neural ar: immersive augmented reality with real-time neural style transfer. In *ACM SIGGRAPH 2019 Virtual, Augmented, and Mixed Reality*, pages 1–1. 2019.

[31] Akhil Singh, Vaibhav Jaiswal, Gaurav Joshi, Adith Sanjeeve, Shilpa Gite, and Ketan Kotecha. Neural style transfer: A critical review. *IEEE Access*, 2021.

[32] Tai-Yin Chiu and Danna Gurari. Pca-based knowledge distillation towards lightweight and content-style balanced photorealistic style transfer models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7844–7853, 2022.

[33] Harco Leslie Hendric Spits Warnars, Benfano Soewito, Ford Lumban Gaol, et al. Shrinking neural style transfer model with knowledge distillation. In *2022 3rd International Conference on Artificial Intelligence and Data Sciences (AiDAS)*, pages 101–106. IEEE, 2022.

[34] Maying Shen, Pavlo Molchanov, Hongxu Yin, and Jose M Alvarez. When to prune? a policy towards early structural pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12247–12256, 2022.

[35] Marc Riera, José María Arnau, and Antonio González. Dnn pruning with principal component analysis and connection importance estimation. *Journal of Systems Architecture*, 122:102336, 2022.

[36] Udo Schlegel, Samuel Schiegg, and Daniel A Keim. Vinnpruner: Visual interactive pruning for deep learning. *arXiv preprint arXiv:2205.15731*, 2022.

6 Appendix

Feel free to upload your own photos and experience the stylization result of our artist's eye. click here