Playing Lottery Tickets in Style Transfer Models

Meihao Kong
Nanjing University
Nanjing, China
kongmeihao@smail.nju.edu.cn

Jing Huo∗
Nanjing University
Nanjing, China
huojing@nju.edu.cn

Wenbin Li
Nanjing University
Nanjing, China
liwenbin@nju.edu.cn

Jing Wu
Cardiff University
Cardiff, United Kingdom
wuj11@cardiff.ac.uk

Yu-Kun Lai
Cardiff University
Cardiff, United Kingdom
laiy4@cardiff.ac.uk

Yang Gao
Nanjing University
Nanjing, China
gaoaoy@nju.edu.cn

ABSTRACT
Style transfer has achieved great success and attracted a wide range of attention from both academic and industrial communities due to its flexible application scenarios. However, the dependence on a pretty large VGG-based autoencoder leads to existing style transfer models having high parameter complexities, which limits their applications on resource-constrained devices. Compared with many other tasks, the compression of style transfer models has been less explored. Recently, the lottery ticket hypothesis (LTH) has shown great potential in finding extremely sparse matching subnetworks which can achieve on par or even better performance than the original full networks when trained in isolation. In this work, we for the first time perform an empirical study to verify whether such trainable matching subnetworks also exist in style transfer models. Specifically, we take two most popular style transfer models, i.e., AdaIN and SANet, as the main testbeds, which represent global and local transformation based style transfer methods respectively. We carry out extensive experiments and comprehensive analysis, and draw the following conclusions. (i) Compared with fixing the VGG encoder, style transfer models can benefit more from training the whole network together. (ii) Using iterative magnitude pruning, we find the matching subnetworks at 89.2% sparsity in AdaIN and 73.7% sparsity in SANet, which demonstrates that Style transfer models can play lottery tickets too. (iii) The feature transformation module should also be pruned to obtain a much sparser model without affecting the existence and quality of the matching subnetworks. (iv) Besides AdaIN and SANet, other models such as LST, MANet, AdaAtTN and MCCNet can also play lottery tickets, which shows that LTH can be generalized to various style transfer models.

CCS CONCEPTS
• Computing methodologies → Computational photography;
  Non-photorealistic rendering.

∗Corresponding author

KEYWORDS
style transfer, neural network pruning, lottery ticket hypothesis

ACM Reference Format:

1 INTRODUCTION
Recent years have witnessed rapid development in the area of neural style transfer, which aims at composing a content image with new styles from reference images. Extensive research has focused on improving visual quality [2, 3, 28, 30, 33, 39], efficiency [25, 26, 29, 41, 47, 49] and flexibility [1, 17, 21, 22, 24, 43]. Although great success has been achieved in these aspects, the memory and computational footprints required for these style transfer models are large owing to the widely adopted autoencoder-based architecture. In particular, they usually consist of a pre-trained VGG encoder [46], a feature transformation module and a corresponding decoder, with a large number of parameters, which makes these models infeasible to be used in resource-constrained scenarios. This naturally raises a question: “Can we prune a large style transfer model while preserving its performance?”

In this paper, we aim to answer the above question via the lens of lottery ticket hypothesis (LTH) [13]. LTH states that there exist small subnetworks in dense neural networks that can be trained in isolation from initialization to match the performance of the original network after training for at most the same number of iterations. Existence of LTH has been successfully shown in various fields [5, 16, 42, 50], and its property has been widely studied [6, 14, 27, 37]. Nonetheless, to the best of our knowledge, no prior work exists on understanding the lottery ticket hypothesis in style transfer, which otherwise could be a powerful tool to understand the parameter redundancy in the current prevailing style transfer models. This will be the focus of our work.

Specifically, we mainly investigate the existence of LTH in two representative style transfer models – AdaIN (a representative global transformation based method) [23] and SANet (a representative local transformation based method) [39]. AdaIN applies mean and standard deviation to transform the features globally, and is the basis for many global style transfer methods, while SANet is the pioneering work that introduces attention mechanism to consider local feature matching.
Are there winning tickets in the whole style transfer models? We further conduct widely verified and comparative experiments, works (winning tickets), but also establishes a stronger baseline of verify the performance of the found ticket. The feature transformation module should also be pruned. We experimentally find that, not only the autoencoder, the feature transformation module can also be pruned to get a sparser matching subnetwork without affecting the existence and quality of winning tickets.

2 RELATED WORK

2.1 Neural Style Transfer

We have witnessed a boom of neural style transfer methods in the past few years. Numerous research works have been conducted to improve the visual quality [8, 23, 28, 30, 31, 33, 39, 45], efficiency [25, 26, 29, 41, 47] and flexibility [17, 22, 24, 43] of style transfer models. Nevertheless, most of these approaches have the common problem of large model sizes due to the widespread adoption of the over-parameterized VGG-based backbone in conjunction with its corresponding feature decoder.

Different from those efforts on improving style transfer model capability regardless of model computational complexity, we focus on making style transfer models sparser and smaller. Note that recently, Wang et al. [49] have also attempted to train smaller style transfer models based on WCT [31] and AdaIN [23]. However, our focus is different from theirs. Specifically, Wang et al. [49] aim to handle the ultra-resolution style transfer task via knowledge distillation. In addition, they have only compressed the encoder without compressing the decoder, hence the overall network is still large. Here, we study the over-parameterization of the whole style transfer networks from the perspective of lottery ticket hypothesis, a popular concept in deep neural network nowadays which has not been introduced into the field of style transfer yet.

2.2 Lottery Ticket Hypothesis

Dating back to 2018, Frankle et al. [13] firstly proposed to find winning tickets via Iterative Magnitude Pruning (IMP). The lottery ticket hypothesis (LTH) has attracted widespread attention and has.
been evidenced in various traditional computer vision fields, such as image classification [35, 36, 44, 48], and object detection [18]. Recently, the properties of LTH has also been widely studied across other fields, such as natural language processing [5, 15, 40, 50], reinforcement learning [50], graph neural networks [6], life-long learning [7], and generative adversarial networks [4, 9, 27]. Besides, the “rewinding late” rule is found by [14, 42] to scale up LTH to larger networks and datasets.

Although LTH has made pioneering progress in various deep learning fields, to our best knowledge, the research of lottery tickets hypothesis in the style transfer field remains untouched. At present, style transfer has played an important role in image and video processing areas, e.g., movie synthesis and photo art. Therefore, it is critical to understand the parameter redundancy in such models and make them small without sacrificing the performance.

3 PRELIMINARIES

In this section, we describe the techniques that we use to find winning tickets and the metrics to evaluate the performance of the subnetworks. We also present our setup for the empirical study.

3.1 Original Full Networks

We use two representative style transfer networks as the main testbeds: AdaIN [23] and SANet [39].

AdaIN is one of the most popular global transformation based style transfer approaches. The core idea is to adaptively transfer the mean and standard deviation from the style feature map to the content feature map. Specifically, given a pre-trained VGG-19 network to encode content features and style features respectively. Secondly, the feature transformation stage:

\[ F_c, F_s = E(I_c, I_s; \theta_E) \]  

where \( E \) is the encoder with parameters \( \theta_E \). Next, the AdaIN module replaces the channel-wise mean and standard deviation from one feature map to the other:

\[ AdaIN(F_c, F_s) = \sigma(F_s)(F_c - \mu(F_c) / \sigma(F_c)) + \mu(F_s) \]

where \( \mu(F_c) / \sigma(F_c) \) calculates the mean of \( F_c \) and \( \sigma(F_c) \) calculates the standard deviation of \( F_c \). For simplicity, we assume \( F_{cs} = AdaIN(F_c, F_s) \). Then, target features \( F_{cs} \) are fed into the decoder \( D \) to obtain the stylized image \( I_t \):

\[ I_t = D(F_{cs}; \theta_D) \]

where \( \theta_D \) denotes the parameters in \( D \).

SANet is a local transformation based method. It is able to flexibly match the local semantically nearest style features onto the content features via a learnable style-attentional transformation module. Similar to AdaIN, SANet also utilizes the pre-trained VGG-based autoencoder architecture. The whole process of SANet can be divided into three stages. Firstly, the feature encoding stage:

\[ F_{c1}, F_{s1}, F_{c5}, F_{s5} = E(I_c, I_s; \theta_E) \]

where \( F_{c1} \) (\( F_{s1} \)) and \( F_{c5} \) (\( F_{s5} \)) denote the corresponding layer VGG feature maps (i.e., relu4_1 and relu5_1) of content (style) images respectively. Secondly, the feature transformation stage:

\[ SANet(F_c, F_s) = T(F_{c1}, F_{s1}, F_{c5}, F_{s5}; \theta_T) \]

where \( T \) is the attention based feature transformation module with trainable parameters \( \theta_T \) (different from the parameter-free transformation in AdaIN). Finally, the stylized output image \( I_t \) is synthesized by feeding \( F_{cs} \) into the decoder just like Eq. (3).

Actually, almost all the mainstream feed-forward style transfer methods \[8, 10, 11, 23, 30, 33, 39, 45\] have similar architectures as AdaIN [8, 23, 45] or SANet [10, 11, 30, 33, 39]. It indicates that if LTH exists in AdaIN and SANet, it will also exist in these methods. Our experiments in Section 4.7 demonstrate this point.

3.2 Subnetworks

For a network \( f \) that maps samples \( x \in \mathcal{X} \) with parameters \( \theta \in \mathbb{R}^d \) to \( f(x; \theta) \), a subnetwork is defined as \( f(x; m \odot \theta) \), where \( m \in \{0, 1\}^d \) is a binary pruning mask indicating which part of the network parameters is set to 0, with \( \odot \) denoting element-wise multiplication. For any configuration \( m \), the effective parameter space of the induced network \( f(x; m \odot \theta) \) is \( \{m \odot \theta \mid \theta \in \mathbb{R}^d\} \), a \( \|m\|_0 \)-dimensional space, hence we say that the subnetwork has \( \|m\|_0 \) parameters instead of \( d \). Specifically, for both AdaIN and SANet, two separate masks, \( m_E \) and \( m_D \), are required for the VGG based encoder and decoder. Moreover, a transformation module mask \( m_T \) is needed for SANet. Accordingly, a general subnetwork of style transfer methods consists of: a sparse encoder \( E(\cdot; m_E \odot \theta_E) \), a sparse transformation module \( T(\cdot; m_T \odot \theta_T) \) and a sparse decoder \( D(\cdot; m_D \odot \theta_D) \).

Algorithm 1 Iterative Magnitude Pruning for Style Transfer Winning Tickets

\[ \text{Input:} \ \text{Total training iteration } N; \ \text{Rewind iteration } r \geq 0; \ \text{Desired sparsity } s \]

\[ \text{Output:} \ \text{A sparse style transfer model } E(\cdot; m_E \odot \theta_E), D(\cdot; m_D \odot \theta_D) \text{ and } T(\cdot; m_T \odot \theta_T) \]

1. Set \( \theta_E^{(r)}, \theta_D^{(r)} \) and \( \theta_T^{(r)} \) as initial weights of \( E(\cdot), D(\cdot) \) and \( T(\cdot) \) respectively.

2. Train \( \cdot; \theta_E \), \( \cdot; \theta_D \) and \( \cdot; \theta_T \) for \( N \) iterations to get parameters \( \theta_E^N \) and \( \theta_D^N \).

3. Set \( \theta_E = 1 \in \mathbb{R} \|\theta_E^N\|_0, \theta_D = 1 \in \mathbb{R} \|\theta_D^N\|_0 \).

4. and \( m_T = 1 \in \mathbb{R} \|\theta_T^N\|_0 \), assume \( m = \{m_E, m_D, m_T\} \).

5. while the sparsity of \( m < s \) do

6. Train \( E(\cdot; m_E \odot \theta_E), D(\cdot; m_D \odot \theta_D) \) for \( N \) iterations to get parameters \( \theta_E^N \) and \( \theta_D^N \).

7. if pruning the transformation module \( T(\cdot) \) then

8. Prune 20% of the parameters in \( \theta_E^N, \theta_D^N \) and \( \theta_T^N \), calculating three mask \( m_{E}', m_D', m_T' \).

9. else

10. Prune 20% of the parameters in \( \theta_E^N \) and \( \theta_D^N \), calculating two mask \( m_{E}', m_D' \), \( m_T' \) remains 1 in \( \mathbb{R} \|\theta_T^N\|_0 \).

11. end if

12. Update \( m_E = m_{E}', m_D = m_{D}' \) and \( m_T = m_{T}' \).

13. end while

3.3 Matching Subnetworks

For a network \( f \) and randomly-initialized parameters \( \theta^{(0)} \), a matching subnetwork \( f^* \) is given by a configuration \( m \in \{0, 1\}^d \), which
is trained in isolation from $\theta^{(0)} = \theta^{(k)} \odot m$, where $\theta^{(k)}$ is the collection of parameter values obtained by training $f$ from $\theta^{(0)}$ for $k$ iterations. Furthermore, to be a matching subnetwork, $f^*$ should reach or even surpass the performance of a trained $f$ given the same training iterations.

**Winning Ticket.** A matching subnetwork $f^*$ is a winning ticket if it can be trained in isolation from initialization. In other words, a winning ticket is a matching subnetwork such that $k = 0$ in the definition above. Ticket search is to identify such a subnetwork, given an unpruned dense network $f$ and randomly initialized parameters $\theta^{(0)}$.

### 3.4 Identifying Subnetworks

Identifying subnetworks is to find three masks $m_E$, $m_D$ and $m_T$ for the encoder, decoder and transformation module, respectively. Note that $m_T$ is not needed for AdaIN. To achieve this, we utilize the Iterative Magnitude Pruning (IMP) algorithm [19]. In particular, we determine the pruning mask $m = (m_E, m_D, m_T)$ by training the full unpruned style transfer network. Then, we prune individual weights with the lowest-magnitudes throughout the network globally. In detail, the position of a remaining weight in $m$ is marked as 1, and the position of a pruned weight is marked as 0. We set the weights of this subnetwork to $\theta^{(i)}$ for a specific rewinding step $i$ during training. For instance, to set the weights of the subnetwork to their values from the initialization, we set $\theta = \theta^{(0)}$. As previous work has shown, to find the smallest possible matching subnetworks, it is better to repeat this pruning process iteratively. Intuitively, we prune a certain amount (e.g., 20%) of non-zero parameters each step and retrain the network several times to reach the desired sparsity rather than pruning the network only once to meet the sparsity requirement. Algorithm 1 presents details of the IMP procedure to find matching subnetworks. In addition, Figure 1 also provides a flowchart to illustrate the process.

### 3.5 Evaluation of Subnetworks

To evaluate whether the subnetwork is a matching subnetwork or not, after obtaining the subnetwork $E(\cdot; m_E \odot \theta_E)$, $D(\cdot; m_D \odot \theta_D)$ and $T(\cdot; m_T \odot \theta_T)$, we reset the weights to $\theta_E^{(r)}$, $\theta_D^{(r)}$ and $\theta_T^{(r)}$ ($r > 0$ if rewinding strategy is used, where $r$ is the rewind iteration). We then re-train the subnetwork, and evaluate whether it can still achieve the performance as the original full network. Note that, all the test content images, as well as all the test style images have never been seen during the training process.

**Quantitative evaluations.** We compare the subnetworks and the original full network via the average style transfer test error $E$ calculated from numerous stylized results based on test images.

$$E = E_{content} + E_{style}$$

For both AdaIN and SANet models, the average style error $E_{style}$ is calculated as:

$$E_{style} = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{I} \left( \| \mu(\Phi_I(I_n)) - \mu(\Phi_I(I_n)) \|_2 + \| \sigma(\Phi_I(I_n)) - \sigma(\Phi_I(I_n)) \|_2 \right)$$

where $N$ denotes the number of test styled images and each $\Phi_I$ denotes the $i$-layer in VGG-19. The average content error $E_{content}$ of AdaIN is calculated as:

$$E_{AdaIN} = \frac{1}{N} \sum_{n=1}^{N} \| F_{c_1} - F_{c_1} \|_2$$

where $F_{c_1}$ denotes the rel_4 1 layer feature maps of the content image and $F_{c_1}$ denotes the content features after AdaIN transformation. Similarly, for SANet:

$$E_{SANet} = \frac{1}{N} \sum_{n=1}^{N} (\| F_{c_1} - F_{c_1} \|_2 + \| F_{s_1} - F_{s_1} \|_2 + \| \Phi_I(I_n) - \Phi_I(I_n) \|_2)$$

where $I_n$ (or $I_s$) denotes the generated results using a common natural image (or painting) as content and style images simultaneously.

Overall, the average style transfer test error is consistent with the optimization loss function for training AdaIN and SANet networks (more details can be found in [23, 39]). Note that in the absence of ideal evaluation metrics for assessing the style transfer models' performance, the average style transfer test error is a feasible alternative to accomplish the measurement task quantitatively. For example, combining the results of Table 1 and Figure 2 (or Figure ?? and Figure 8), we can see that there is a strong correlation between visual quality and the test error, i.e., better performance obtained in terms of content preservation and stylization degree where there is a smaller average style transfer test error.

**Qualitative evaluations.** We also conduct a user study to qualitatively evaluate the subnetworks. Specifically, 15 content images and 20 style images are randomly picked to form 300 images pairs in total. Then we randomly sample 100 content-style pairs and synthesize stylized images using both the original full networks and the corresponding subnetworks. Results are presented side-by-side in a random order and we ask subjects to select the most visually pleasant one from three views: content preservation, stylization degree, and overall preference. We collect 2000 votes from 20 users and present the statistical results in Figure 5.

**Datasets and settings.** In the training phase, we use MS-COCO dataset [32] and WikiArt dataset [38] as our content image set and style image set, respectively. Each dataset contains roughly 80,000 training examples. Besides, we follow the same settings (e.g., hyperparameters, image resolution, etc.) as described in [23, 39] to train the original full model or its corresponding subnetworks. In the testing phase, 40 content images and 100 style images are randomly selected from the test set of the MS-COCO [32] and WikiArt dataset [38] to calculate the average style transfer test error. All models are trained on a GeForce RTX 2080 Ti GPU.

## 4 EXPERIMENTAL RESULTS AND ANALYSIS

### 4.1 Training Together Gains A Lot

Before starting the experimental verification of the LTH, we perform some preparatory experiments. As we all know, most of the
Table 1: Quantitative performance of different training strategies. AdaIN+: training encoder together based on AdaIN, SANet+: training encoder together based on SANet.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Avg content error</th>
<th>Avg style error</th>
<th>Avg error</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaIN [23]</td>
<td>2.284</td>
<td>6.013</td>
<td>8.297</td>
</tr>
<tr>
<td>AdaIN+</td>
<td>2.199</td>
<td>3.486</td>
<td>5.685</td>
</tr>
<tr>
<td>SANet+</td>
<td>5.618</td>
<td>3.009</td>
<td>8.627</td>
</tr>
</tbody>
</table>

Figure 2: Image style transfer results of different model training strategies. Zoom in to have a better view.

existing style transfer methods adopt a common training strategy that fixes the VGG-19 encoder while only training the decoder and the feature transformation module. However, on the one hand, we are not able to search for winning tickets across the whole model while adopting the above training strategy, hence the VGG encoder is not considered during the training phase. On the other hand, it overly relies on the content and style patterns representation ability of the pre-trained VGG network which is not trained for this purpose and may not always be reliable. Therefore, we perform experiments to compare the above training strategy and training together strategy. For simplicity, we name the training together strategy as a plus strategy. The quantitative and qualitative results are shown in Table 1 and Figure 2, respectively.

From Table 1, we can see that the plus strategy outperforms the original strategy overall in terms of content, style and total errors. In Figure 2, by comparing stylized results of the 2nd (4th) and 3rd (5th) columns, we can further determine that the plus strategy is generally better than the traditional training strategy, combined with the aspect of content preservation and stylization degree. Therefore, we claim that “Training Together Gains A Lot”, which not only makes it possible to search for winning tickets across the whole network, but also establishes a stronger baseline of the original full model.

4.2 Style Transfer Models Can Play Lottery Tickets Too

To prove this point, we follow the same procedure in [13]. We first conduct experiments on AdaIN by pruning the VGG encoder and decoder with the following steps: 1) Run IMP to obtain the sparsity pattern \( \{m_E', m_D'\} \), with \( s\% \) sparsity; 2) Initialize the resulting subnetwork to \( \theta_E^{(0)} \) and \( \theta_D^{(0)} \). This produces a subnetwork \( E(\cdot; m_E' \odot \theta_E^{(0)}), D(\cdot; m_D' \odot \theta_D^{(0)}) \); 3) Train this subnetwork again to evaluate whether it is a winning ticket. In detail, we set \( s\% = (1 - 0.8^i) \times 100\% \), which we use for all the experiments that involve iteratively pruning hereafter. For SANet, besides \( m_E' \) and \( m_D' \), \( m_T' \) is also required to produce the subnetwork \( E(\cdot; m_E' \odot \theta_E^{(0)}), T(\cdot; m_T' \odot \theta_T^{(0)}), D(\cdot; m_D' \odot \theta_D^{(0)}) \). The overall process is similar to AdaIN. More specific details can be found in Algorithm 1.

Figure 3 shows the quantitative performance of the subnetworks generated by IMP with different sparsity levels. As can be seen, we are still able to find the winning tickets by iteratively pruning the entire networks to very high sparsity, around 90% in AdaIN+, and around 74% in SANet+, where the test errors of these subnetworks are comparable or even less than the full networks.

To give a more intuitive impression, we visualize the stylized results in Figures 4. For AdaIN+, we can see that the extremely sparse subnetwork performs very well even with only 10.8% parameters of the full network. Surprisingly, the found winning tickets in AdaIN+ can even yield more pleasant stylized results with fewer artifacts and better content preservation in detail, e.g., less artifacts at the edges of the buildings (row 2), and in the sky (row 6), more complete bridge structure (row 5). We speculate that this phenomenon may be explained since a model with fewer parameters tends to be less overfitting. Meanwhile, from the stylized results generated by winning tickets found in SANet+, we can also observe that in the case of high sparsity, i.e., 73.7%, the matching subnetwork can still achieve comparable performance to the original full model.

Moreover, we conduct a user study to evaluate the quality of the winning tickets more objectively. The statistical results are presented in Figure 5. We can see that no matter AdaIN+ or SANet+, the found winning tickets can achieve the performance that is comparable or even outperforms the full models, i.e., 38.4% vs. 25.7% for AdaIN+ and 35.5% vs. 32.4% for SANet+. Besides AdaIN and SANet, the existence of the LTH can also be confirmed in many other style transfer models as shown in Section 4.7. In summary, through the comparison and analysis above, we can conclude that "Style Transfer Models Can Play Lottery Tickets Too".

4.3 Feature Transformation Module Should Also be Pruned

Actually, for a range of feature transformation based style transfer models with learnable parameters [10, 11, 30, 33, 39], the intermediate transformation modules also contain a large number of
parameters, sometimes even more than the total parameters of the decoder. Taking SANet as an example, the parameter sizes of the encoder, transformation module and decoder are 49.38MB, 17.02MB and 13.37MB respectively, which means that even all parameters of autoencoder are pruned, only up to 78.7% sparsity can be achieved.

Therefore, a natural question that came to our mind is: Can we prune the feature transformation module to get a much sparser style transfer network without compromising performance?

To answer this question, we compare two different iterative pruning settings for both SANet [39] and LST [30]: 1) Prune the

Figure 4: Image style transfer results of Winning Tickets found by IMP. Left: winning tickets of AdaIN+; Right: winning tickets of SANet+; FM: Full AdaIN+ (SANet+) model; Best WT: Winning Tickets with best performance, i.e., lowest test error (sparsity of 59% for AdaIN+ and sparsity of 48.8% for SANet+); Extreme WT: Winning Tickets with highest sparsity (89.2% for AdaIN+ and 73.7% for SANet+). Zoom in to have a better view.
Figure 5: User study of Winning Tickets found by IMP. Left: winning tickets of AdaIN+; Right: winning tickets of SANet+. The percentage in red shown underneath each method indicates the sparsity of corresponding network.

autoencoder only (noPT); 2) Prune both the autoencoder and the transformation module (PT). All parts of the network are reset to the same random initialization $\theta^{(0)}$ after the masks are obtained.

The experimental results are reported in Figure 6 and Figure 7, respectively. Both graphs suggest that the two settings share similar patterns when subnetworks are not too sparse (i.e., 0%-50% of sparsity). However, as the total number of parameters of the model is further reduced to 30% or less, PT strategy shows enormous advantage compared to noPT strategy. The explanation is probably that adopting the noPT strategy, the parameters in the encoder and decoder will be severely reduced to get higher sparsity overall, hence damaging the quality of the subnetworks. While the PT strategy can achieve the overall trade-off of the whole model parameters and still perform well with extremely high sparsity, as the transformation module is also pruned.

Figure 6: Results of whether to prune the feature transformation module of SANet. The dashed line indicates the performance of the full SANet+ model.

4.4 IMP Winning Tickets Are Sparser than OMP, Random Pruning, and Random Tickets

Previous work describes winning tickets as a “combination of weights and connections capable of learning” [13], which means both the specific pruned weights and the specific initialization are necessary for a winning ticket to achieve this performance. To extend such a statement in the context of style transfer models, we compare IMP with several other benchmarks, one-shot magnitude pruning (OMP), random pruning (RP), and random tickets (RT). Specifically, we train a subnetwork $\{E(\cdot; m^{OMP}_E \odot \theta^{(0)}_E), D(\cdot; m^{OMP}_D \odot \theta^{(0)}_D)\}$ with a one-hot magnitude pruning mask $m^{OMP}$ (which evaluates the importance of the iterative pruning strategy), a subnetwork $\{E(\cdot; m^{RP}_E \odot \theta^{(0)}_E), D(\cdot; m^{RP}_D \odot \theta^{(0)}_D)\}$ with a random pruning mask $m^{RP}$ (which evaluates the importance of the pruning masks) and a subnetwork $\{E(\cdot; m^{IMP}_E \odot \theta^{(0)}_E), D(\cdot; m^{IMP}_D \odot \theta^{(0)}_D)\}$ with a random initialization $\theta'$ (which evaluates the importance of the pre-trained initialization) based on AdaIN+ to see if IMP can obtain the best performance.

Table 2: Results of the best subnetworks and the extreme sparsity of matching networks found by different pruning settings. $E_{\text{Best}}$: Minimal test error of all subnetworks. $S_{\text{Extreme}}$: Extreme sparsity where matching subnetworks exist.

<table>
<thead>
<tr>
<th>Methods</th>
<th>$E_{\text{Best}}$(Sparsity)</th>
<th>$S_{\text{Extreme}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Model</td>
<td>5.685(0.0%)</td>
<td>–</td>
</tr>
<tr>
<td>IMP</td>
<td>5.134(59.0%)</td>
<td>89.2%</td>
</tr>
<tr>
<td>OMP</td>
<td>5.255(59.0%)</td>
<td>59.0%</td>
</tr>
<tr>
<td>Random Pruning</td>
<td>5.413(20.0%)</td>
<td>30.0%</td>
</tr>
<tr>
<td>Random Tickets</td>
<td>–</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

As shown in Table 2, the extreme winning ticket found by IMP is significantly sparser than that found by other pruning settings, and the minimal test error is smaller as well, which confirms the superiority of iterative pruning. Moreover, when adopting random tickets, we cannot obtain any matching subnetworks. This observation defends that specific initialization is essential for finding winning tickets. In Figure 8, we also show the visual performance of subnetworks with sparsity of 89.2% obtained by different pruning strategies. We observe that all other subnetworks have poor performance in style transfer with such a high sparsity, except for the IMP winning ticket.

4.5 Whether Rewinding Improves the Performance

In previous paragraphs, we demonstrate that we are able to find winning tickets in both AdaIN and SANet at non-trivial sparsities (i.e., sparsities where random pruning cannot find winning tickets).
Considering that the work [42] shows the rewinding paradigm is necessary to identify winning tickets for large networks, we would like to examine whether rewinding is helpful in the context of style transfer models. We perform experiments at different rewinding ratios. Specifically, after IMP training, we obtain the masks. Then, instead of resetting the weights to $\theta(0)$, we rewind the weights to $\theta(i)$, i.e., the weights after $i$ steps of training. The rewinding ratio = $i/N$, where $N$ is the total training iteration. The results are shown in Table 3, we can see that rewinding does not have a notable effect. In particular, subnetworks trained at different rewinding ratios have the similar highest sparsity and best performance.

### Table 3: Rewinding results of the best subnetworks and the extreme sparsity of matching networks found by IMP.

<table>
<thead>
<tr>
<th>Rewinding ratios</th>
<th>AdaIN</th>
<th>SANet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\epsilon_{\text{Best}}$</td>
<td>$\epsilon_{\text{Best}}$</td>
</tr>
<tr>
<td>Rewind 0%</td>
<td>5.134</td>
<td>89.2%</td>
</tr>
<tr>
<td>Rewind 10%</td>
<td>5.114</td>
<td>89.2%</td>
</tr>
<tr>
<td>Rewind 20%</td>
<td>5.097</td>
<td>89.2%</td>
</tr>
<tr>
<td>Rewind 30%</td>
<td>5.052</td>
<td>89.2%</td>
</tr>
<tr>
<td>Rewind 40%</td>
<td>5.083</td>
<td>89.2%</td>
</tr>
</tbody>
</table>

#### 4.6 IMP Winning Tickets vs. Other Pruning Methods

We further perform experiments to compare the IMP Winning Tickets with other mainstream pruning methods, i.e., the structured channel pruning method [34] (network slimming) and finetuning based magnitude pruning method [20]. Results are shown in Figure 9. It can be seen that the IMP winning tickets are significantly better than subnetworks found by network slimming and finetuning based magnitude pruning at high sparsity.

#### 4.7 Experiments on Other Style Transfer Models

To verify the universal presence of the lottery ticket hypothesis in diverse style transfer models, we also conduct experiments on LST [30], MANet [11], AdaAttN [33] and MCCNet [10]. Table 4 shows that LTH can be generalized to various style transfer models despite the different extreme sparsities. Besides, we are surprised to find that the smallest matching subnetwork of LST has only 6.9% of the parameters of the original full model.

### Table 4: Results on other style transfer models. $\epsilon_{\text{Full}}$: The test error of the full model. $\epsilon_{\text{Best}}$: The minimal test error of all subnetworks. $S_{\text{Extreme}}$: Extreme sparsity where matching subnetworks exist. (%) denotes the sparsity of the corresponding network.

<table>
<thead>
<tr>
<th>Model</th>
<th>$\epsilon_{\text{Full}}$</th>
<th>$\epsilon_{\text{Best}}$</th>
<th>$S_{\text{Extreme}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LST [30]</td>
<td>3.103(0%)</td>
<td>2.686(79.0%)</td>
<td>93.1%</td>
</tr>
<tr>
<td>MANet [11]</td>
<td>17.176(0%)</td>
<td>11.227(20.0%)</td>
<td>79.0%</td>
</tr>
<tr>
<td>AdaAttN [33]</td>
<td>24.467(0%)</td>
<td>22.873(48.8%)</td>
<td>73.7%</td>
</tr>
<tr>
<td>MCCNet [10]</td>
<td>7.875(0%)</td>
<td>7.574(59.0%)</td>
<td>83.2%</td>
</tr>
</tbody>
</table>

#### 5 CONCLUSION

In this paper, the LTH has been extended to the style transfer field for the first time. Through extensive experiments and comprehensive analysis, we verify the existence of winning tickets in a range of style transfer models. In future work, we plan to examine the speedup results on a hardware platform that is friendly to unstructured pruning. For instance, XNNPACK [12] has shown significant speedups over dense baselines on smartphone processors at 70%-90% unstructured sparsity. Besides, we will further explore the application of LTH in more diverse style transfer scenarios, e.g., video synthesis, caricature generation, etc.

#### 6 ACKNOWLEDGMENTS

This work is supported by Science and Technology Innovation 2030 New Generation Artificial Intelligence Major Project (2018AAA0100905, 2021ZD0113303), the National Natural Science Foundation of China (61806092, 62106100), the CAAI-Huawei MindSpore Open Fund and the Collaborative Innovation Center of Novel Software Technology and Industrialization.
REFERENCES


