Image Augmentation with Neural Style Transfer

Borijan Georgievski[1,2](✉)

[1] Netcetera, Skopje, North Macedonia
borijangeorgievski@protonmail.com
[2] FCSE, Ss. Cyril and Methodius University, Skopje, North Macedonia

Abstract. The amount of training data is of crucial importance for the performance of machine learning, and especially deep learning models. It is one of the most important factors that determine whether the developed model is effective or not. When the quantity of training data for a computer vision problem is insufficient, various data augmentation techniques are used to artificially extend the training dataset with samples that retain the natural distribution of the original data. This paper proposes and evaluates a deep learning model that will be used for image augmentation. A complex deep neural network makes use of transfer learning in order to learn the characteristics of the content and style of the training images, create random style embeddings via learned multivariate normal distribution, and ultimately generate images to extend the original dataset. The model is trained on two datasets which are frequently used in computer vision: ImageNet and Painter by Numbers (PBN). Afterwards, the model is used to generate new images from the CIFAR-100 and Tiny-ImageNet-200 datasets. The performance of the augmentation model is evaluated by a separate convolutional neural network. The evaluation model is trained on the combined dataset, consisting of both, the original and augmented images, and then compared to the performance of the same model trained on the original datasets.

Keywords: Image augmentation · Neural style transfer · Computer Vision Convolutional Neural Networks (CNNs) · Deep learning

1 Introduction

In the current era of deep learning, there is one thing that can always improve a developed model, and that is more data. On one hand, deep learning models are becoming more accurate than every other carefully developed and hand-designed machine learning method, but on the other hand, they also need much more data. Having a small dataset is one of the biggest setbacks in computer vision projects, especially those with a deep learning approach. This paper focuses on exploring image manipulation through neural style transfer, and adopts an approach for randomizing style, in order to achieve arbitrary image augmentation.

Neural style transfer is a technique for reconstructing images by changing their style. It all started when Gatys et al. (2015) showed the possibility of using convolutional neural networks for transforming images, such that they are altered by applying styles of other chosen images, whilst preserving their content [2]. A steady progress has been made in a number of research studies [1, 3, 5, 15] since the original idea was proposed. The current state-of-the-art models can generate a new image based on content and style input images in a single forward pass. Even though, neural style transfer was intended for creating appealing images by combining content and style images of our choosing, the ability to extract style and content separately from an image seems to have promising uses in image augmentation. Although the pioneering approaches were usually limited in the number of styles that can be applied to images, nowadays the content and style images can be completely arbitrary. Stylizing an image requires us to have an already sampled style image, but a data augmenter capable of generating only a limited (or finite) number of different augmentations may be undesirable.

The model presented in this paper is not limited to any number of styles. Arbitrary augmentation is achieved by using a style embedding as style input instead of a real image. In this manner, the style embedding can still be extracted from a style image, or like in our case, it can be randomized. A pretrained neural network is fine-tuned and trained to extract a style embedding from an image, so that afterwards the channel-wise mean and covariance matrix of the styles observed in training phase are used to define a multivariate normal distribution. Therefore, this distribution can be sampled for an arbitrary number of times to generate a randomized style embedding which still maintains the natural distribution of the style images dataset. The complex loss function defined for this model uses a pretrained VGG19 model to extract the content and styles from the input and output images. Finally, the model's performance is evaluated with a separate convolutional neural network on the CIFAR-100 and Tiny-ImageNet-200 datasets.

2 Related Work

This section provides theoretical analysis of the concepts and methods used in this paper. A summary of previous related research on data augmentation and neural style transfer will provide a background against which this work should be positioned and compared.

2.1 ImageNet and Successful Architectures

Nowadays, there are many excellent neural networks which are available to the public, either as pretrained networks, or as architectures that only require data for the input, so that the users can train them themselves. Usually, the pretrained models are benchmarked at the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competition, active since 2010. Each year, the ImageNet organizers release high-quality, substantial dataset to be used for training, and many researchers use this data for other projects. The ILSVRC2017 image dataset contains over 200 GB of images, having two separate datasets for object localization and object detection. The two datasets combined contain approximately two million images. Our style transformer module is being trained with 537K images from this dataset.

The models which achieve best results at the annual competition are very often released to the public as pretrained networks, making their utilization very convenient to the independent researchers. One such successful model available for use is VGG [12], which is utilized in the proposed model as a pretrained loss network to extract low-level and high-level semantic representations. The input size of the model is 224×224 on three color channels (RGB), and only 3×3 convolutions and 2×2 pooling are used throughout the whole network. Every convolution in the VGG architecture uses rectified linear activation (ReLU). VGG also shows that the depth of the neural network plays an important role, in the sense that deeper networks give better results. One drawback of VGG is that this network is very big and resource heavy, containing around 160 million trainable parameters. In our model, the 19-layer pretrained VGG network is used as a feature extractor for the loss function.

He et al. (2015) find that training extremely deep neural networks are very hard to train because of vanishing and exploding gradient problems, but also propose a method for allowing to make extremely deep convolution neural networks (up to 152 layers) trainable [4]. The residual networks described in the paper yielded the best results at many subtasks of the ILSVRC 2015 competition (by the team name MSRA). Their main groundbreaking idea, which is also used in this paper, is the introduction of the residual block. It is a building block that can be implemented in very deep CNNs, where it can perform an identity mapping from a shortcut (or a skip connection), adding the shortcut to a feature map that is found several layers deeper, resulting in preserving past information in networks with even 100+ layers.

Szegedy et al. (2016) release the fourth iteration of Inception (Inception-V4) and Inception-ResNet-V2 which combines the best out of the two models, by utilizing the residual blocks inside the Inception architecture [13]. This model achieves superb Top5 accuracy of 95.3 on the ILSVRC-2012-CLS dataset, while VGG16 (16-layer model) and VGG19 (19-layer model) achieve 89.8 Top-5 accuracy. Kornblith et al. (2018) show that when the networks are used as fixed feature extractors or when they are fine-tuned, there is a strong correlation between the ImageNet accuracy and transfer accuracy, $r = 0.99$ and 0.96, respectively, suggesting that better ImageNet architectures are capable of learning better, transferable representations [9]. That is why the InceptionResNet-V2, pretrained on the ImageNet dataset, is used as a feature extractor for the style predictor module.

2.2 Neural Style Transfer

One of the most exciting applications of CNNs in the last years has been neural style transfer. Neural style transfer is a technique that extracts the content from an arbitrary image (content input), extracts the style of another image (style input), and generates a new image by combining the extracted content and style. Gatys et al. derive the neural representations of the content and style of an image from the feature responses of a 19-layer VGG network trained on object recognition. The feature space is provided by 16 convolutional layers and 5 average-pooling layers. The actual learning is performed by first initializing the generated image G randomly, and then iteratively improving the image by using gradient descent to minimize the cost function. They have designed the network such that the generated image G is an input to the network, modifying it after each forward propagation based on the total loss. The overall optimization objective is defined as a combination of the style loss, and content loss [2]:

$$\mathcal{L}_{total} = \alpha \mathcal{L}_c(x, c) + \beta \mathcal{L}_s(x, s) \tag{1}$$

where $\mathcal{L}_c(x,\ c)$ and $\mathcal{L}_s(x,\ c)$ are the content loss and style loss components, while α and β are parameters which define the weights of the style and content losses. These two loss components use a pre-trained CNN (VGG19), in order to extract semantic features from the images.

Although this pioneering approach yielded good results, it is very computationally inefficient. In order to generate a single image given a content and style input, this model would need multiple forward propagations before it converges. To generate a second image, all this needs to be repeated once again. However, since the release of this paper there are many alternate approaches, many of which succeed in generating an image through a single forward pass.

Fast approximations with feed-forward neural networks have been proposed to speed up neural style transfer. Yanai (2017) proposes a model which generates a stylized image through a single forward pass. This approach involves a style condition network which generates a conditional signal from a style image directly. By adding a CNN that takes a style image as an input and outputs a conditional signal, the whole network can learn unlimited number of styles. To achieve style transfer for unseen styles, the CNN is expected to generate a conditional style signal by combining the conditional signals of the trained styles [15].

Ioffe et al. (2015) show how including batch normalization or instance normalization as part of the model architecture allows us to use much higher learning rates and be less careful about initialization, by performing normalization for each training mini-batch [6]. Ulyanov et al. (2017) show that by using instance normalization instead of regular batch normalization, it is possible to dramatically improve the performance of deep neural networks for image generation [14].

Huang et al. (2017) present an arbitrary style transfer approach for stylization in real-time by introducing an adaptive instance normalization (AdaIN) layer that adjust the mean and variance of the content features to match those of the style features. Most importantly in our case, these normalization techniques also act as a regularizer. Similarly to regular instance normalization, the mean μ and variance σ are computed across the spatial axes of an encoder network applied to a style image. However, unlike batch normalization, instance normalization and conditional instance normalization (detailed below), the AdaIN layer has no learnable affine parameters. Instead, it adaptively computes the affine parameters from the style input [5]:

$$AdaIN(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y) \tag{2}$$

Following up on instance normalization, Dumoulin et al. (2017) propose a conditional instance normalization, which generates a normalized activation $CIN(x,\ s)$, such that it learns a different set of parameters γ^s and β^s for each style s [1]:

$$CIN(x, s) = \gamma^s \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \beta^s \tag{3}$$

where μ and σ are the mean and variance of the input x across the spatial axes. With the use of conditional instance normalization, Ghiasi et al. (2017) propose a method for fast and arbitrary style transfer in real-time. They build a style transfer network $S(\cdot)$ as an encoder-decoder, such that it shares its representation across many paintings, providing a rich vocabulary for representing any painting. Next, a style prediction network $P(\cdot)$ predicts an embedding vector S from an input style image, which supplies a set of normalization constants for the style transfer network. The advantage of this approach is that the model can generalize to an unseen style image by predicting its proper style embedding at test time [3]. They employ a pretrained Inception-v3 architecture and compute the mean across each activation channel of the truncated model's output which returns a feature vector with the dimension of 768. Afterwards, they apply two fully connected layers on top of the pretrained network to predict the final embedding S.

Following the approach of Dumoulin et al., Ghiasi et al. employ conditional instance normalization to normalize activation channels using the embedding vector S. However, this method learns the mapping from the style image to style parameters directly, as opposed to providing a fixed heuristic mapping from style image to normalization parameters. The content and style losses are derived from the distance in representational space of the VGG image classification network. Their model architecture is used in this paper, and it was selected from the alternative approaches mainly because the model is very intriguing and intuitive. In addition, the paper provides detailed information for the model configuration.

2.3 Image Augmentation

The generalizability of any machine learning algorithm is very often determined by the size of the training datasets, the more training data are used the better generalization is expected from the model. Unfortunately, in practice, the amount of data available for a project is limited. One way to solve this problem is by increasing the size of the training set by adding artificially or synthetically created data to the training set.

Models can benefit from simple augmentation techniques, such as: random translations, random rotations, horizontal/vertical flips, scaling and blurring. Simple operations like these can often greatly improve generalization, even if the model has already been designed to be translation invariant by using the convolution and pooling techniques that were discussed in previous sections/chapters. More advanced transformations exist, such as, nonlinear geometric distortions of the input or altering the intensities of the RGB channels in training images along with image translations and horizontal reflections [10].

Adding noise in the input to a neural network is also a form of data augmentation. For many classification and regression tasks, the task should still be possible to solve, even if small random noise is added to the input. Noise injection can also work when the noise is applied to the hidden units in a network, which can be interpreted as performing a dataset augmentation at multiple levels of abstraction. Dropout can also be seen as a process of constructing new inputs through multiplying by noise. Data augmentation is effective for speech tasks as well, where the speech can be synthesized with noise in order to mimic actual human speech [11].

Jackson et al. (2018) explore the option of augmenting images using neural style transfer. They adapt the style transfer model architecture proposed by Ghiasi et al. to perform style randomization, by sampling input style embeddings from a multivariate normal distribution instead of inferring them from a style image [8]. During training, the style augmentation randomizes texture, contrast, and color, while preserving shape and semantic content. They chose the approach of Ghiasi et al., for its speed, flexibility, and visually compelling results. This research is of crucial importance for our work, as the model architecture very closely follows the ideas described in the paper.

3 Methodology

This section is dedicated to demonstrating the construction of the model, the learning phase and the project setup. The model progress throughout the training epochs will also be presented.

3.1 Model

The model structure, which is based on the approach proposed by Ghiasi et al., can be presented as a form of complex concatenation of two networks. One of them is the style predictor (•), which has the task of predicting the style embedding vector S, based on the input image. The style predictor uses the Inception-ResNet-V2 model, as a feature extractor, pretrained on the ImageNet dataset. The spatial hierarchy of features learned by the pretrained network effectively acts as a generic model of the visual world. This model is truncated at the layer mixed-6a, the output of which takes the shape of (Batch size, 17, 17, 1088), where we have 1088 features for each point in the 17×17 spatial map. The output of the layer mixed-6a is mean pooled across the tensor's spatial axes. The resulting tensor is finally passed through a 1×1 convolutional layer that produces a style embedding of shape (Batch size, 1, 1, 100). The low-level layers of this pretrained network would capture very low-level (and thus not useful) features, while the high-level layers would capture very complex concepts. The downside from extracting features deep in the network is that they are getting increasingly specific to the task that the model was previously trained on.

The style predictor has a total of 4,451,140 parameters, but only 108,900 of them are trainable. Most of the features are frozen, which is good for maintaining the complexity of this network as small as possible, considering the style transformer and the loss network (VGG19) are computationally expensive to run. The frozen layers do not update any parameters during backpropagation, and this is useful for many different reasons, such as: reducing training time by having less trainable parameters [7] and for achieving better results since the pretrained model was trained on a much bigger dataset.

The second network is called style transformer, which is a CNN in the form of encoder-decoder that uses residual blocks. It receives as input an RGB image, which is an array of shape (256, 256, 3), and produces a stylized RGB image of the same shape. The input is preprocessed such that the pixels are scaled between −1 and 1, sample wise. Furthermore, the input images use reflection padding instead of zero-padding, in order to eliminate border artifacts. The style embedding vector S, generated from the style predictor, controls the style transformer via conditional instance normalization (Eq. 3). The values of γ^s and β^s are calculated by passing S, once for each parameter, through a 1×1 convolutional layer, which acts like a memory efficient fully connected layer.

In contrast to the model presented in Ghiasi et al., in which a style image is taken as an input for each content image, S is a weighted combination between a random embedding and the style of the input (content) image. This allows us to generate an arbitrary number of styles during run-time, ultimately augmenting the data. The random embedding vector is sampled from a multivariate normal distribution with defined mean and covariance matrix that are calculated as the channel-wise metrics for all training style images, which were observed during the style predictor's training phase. Therefore, the final embedding is a function of the content image c [8]:

$$S = \alpha \mathcal{N}\left(\sigma, \sum\right) + (1-\alpha)P(c) \tag{4}$$

P(c) is the output of the style predictor with c as the input image, (σ, Σ) is the multivariate normal distribution with the observed mean and covariance matrix, and α is the hyperparameter augmentation strength, i.e. the ratio of randomness in the style embedding, as opposed to the actual style of the input.

3.2 Loss Function

The overall optimization objective is defined as a combination of the style loss, and content loss [3]:

$$\min_x \; \mathcal{L}_c(x, c) + \lambda_s \mathcal{L}_s(x, s) \tag{5}$$

where $\mathcal{L}_c(x, c)$ and $\mathcal{L}_s(x, c)$ are the content loss and style loss components, while λ_s is a scalar hyperparameter which defines the relative weights of the style and content losses.

It is known that higher layers in the network capture the high-level content in terms of objects and their arrangement in the input image, but do not constrain the exact pixel values of the reconstruction. Furthermore, the content loss is defined as an average distance (Frobenius norm) between the high-level semantic features of the content image and the generated image when passed through the pre-trained network:

$$\mathcal{L}_c = \sum\nolimits_{i \in C} \frac{1}{n_i} \| f_i(x) - f_i(c) \|_F^2 \tag{6}$$

where c and x are the content and restyled images, f is the loss network, $f_i(x)$ is the activation tensor of layer i after passing x through f, and n_i is the number of units in the layer i. Consequently, the style can be represented as a set of Gram matrices that describe the correlations between low-level convolutional features. Therefore, the style loss can be expressed as:

$$\mathcal{L}_s = \sum\nolimits_{i \in S} \frac{1}{n_i} \|G[f_i(x)] - G[f_i(s)]\|_F^2 \tag{7}$$

where s and x are the style and restyled images, and $G[f_i(s)]$ denotes the Gram matrix of layer i activations of f.

This loss function is using a pretrained VGG19 model, pretrained on ImageNet as a feature extractor for the input and output images. For each input, there are 6 total outputs from the VGG19, 3 of them are low-level and the other 3 are high-level feature maps. Both images are passed through the VGG19 model, followed by calculating the content and style losses (Eqs. 6 and 7) and the total loss, which is a weighted sum of both. Using mini-batch gradient descent for training, the total loss in one forward pass is computed as the average loss out of every sample in the batch. This method does not always result in the fastest model convergence, but it is very computationally efficient.

3.3 Training

The model was trained on a Microsoft Azure Data Science Virtual Machine on Ubuntu, powered by four NVIDIA Tesla K80 graphic cards. The style predictor was trained on 103K images from the Painter by Numbers dataset, while the style transformer was trained on 537K images from the ImageNet dataset. Training was done using mini-batch gradient descent with batch size of 16. The chosen optimizer was Adam, with learning rate = 0.001, $beta_1 = 0.9$, $beta_2 = 0.999$ and no learning decay. Surprisingly, despite the huge computational power of the virtual machine, the training was very slow.

Unfortunately, because of time limitations, the style transformer was trained for a total of only 8 epochs. On the bright side, since the ImageNet dataset is very big, during these 8 epochs the model has observed around 4.3M input images. Because it took so long to train this network, it was impossible to do hyperparameters optimization search, therefore the augmentation strength α was manually tuned. The first 4 epochs had α fixed on 0.45, while the last ones were trained with $\alpha = 0.2$. On every ten thousand steps, the training progress is visualized by generating an image from a random sample (see Fig. 1).

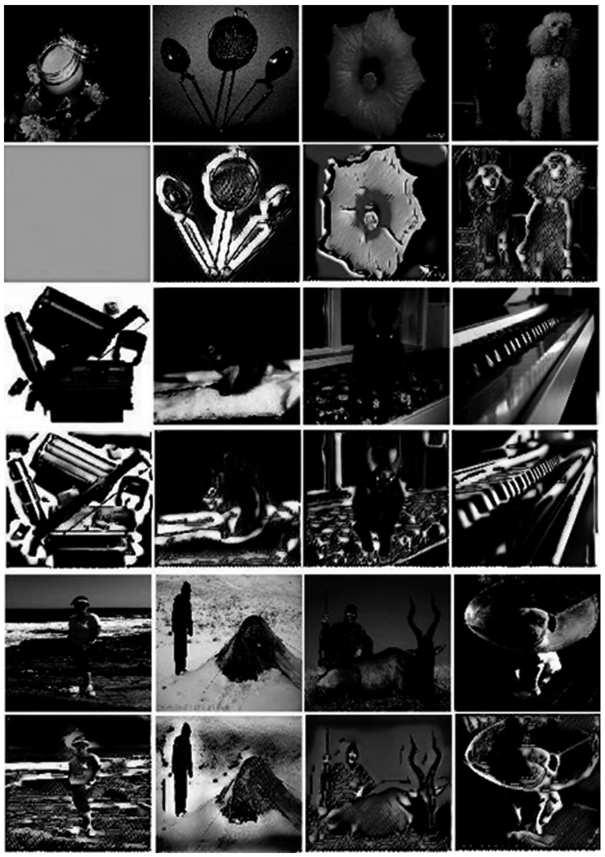

Fig. 1. Training progress of the model, starting from epoch 0 to epoch 7. Odd rows represent the original (input) images, and the even rows represent the corresponding stylized images.

3.4 Experiments and Results

In order to evaluate the augmenter model, we need to check if it is effective in improving the performance of an arbitrary model, by augmenting a dataset the model was trained on, and train it again with the additional data. With the computational limitations we faced, choosing the CIFAR-100 and Tiny-ImageNet-200 datasets to be augmented was an easy choice, since the images are small (32×32 and 64×64 pixels, respectively), so it was easy to compute them in batches. Aside from that, these datasets consisting of 100 and 200 classes respectively are not yet completely solved, as opposed to MNIST or CIFAR-10.

Two Fully Convolutional Networks (FCN) with 142 K parameters were trained, in order to compare the results: one for the original dataset, and another for the dataset combined with its augmented images (with ratio 1:1). While training the evaluator model, none of the other augmentation techniques were used (horizontal/vertical flip, random crop, rotate, zoom, blur, etc.), in order to test the actual contribution of the method. Training the models on the Tiny-ImageNet-200 dataset lasted 100 epochs. On the other hand, an early stopping method is employed when training both models on CIFAR-100, so the model trained on the original dataset lasted 25 epochs before the validation loss starts to increase, while the latter model lasted 15 epochs (see Figs. 2 and 3). This makes sense because the amount of randomness applied in conditional instance normalization is smaller for low values of augmentation strength (0.45 and 0.2). This means the generated images follow a similar distribution to the input images, which saturate the model quickly.

We can see both models evaluate almost the same on the validation set, and it seems like doubling the CIFAR-100 dataset size using the developed image augmenter did not improve the evaluator model. Similarly, the evaluation on the Tiny-ImageNet-200 dataset shows no improvement on the model (see Fig. 4). The performed experiments showed that the evaluator models did not achieve any significant improvements by using the augmented images in the dataset. However, that may have happened because of multiple factors which are rectifiable, such as:

- The image augmenter was only trained for 8 epochs. This was due to limited computational power and time, but it would be easy to continue learning the model by loading the saved weights.
- No hyperparameters optimization. Most specifically, the augmentation strength may have been too small, and if that is the case, the generated images closely resemble the distribution of the input images, which could mean that the evaluator model did not learn additional significant information from the augmented images. It may be wise to try higher augmentation strength when training the model.
- The CIFAR-100 and Tiny-ImageNet-200 datasets are fully consisting of 32×32 and 64×64 pixel images, respectively, while the input of the image augmenter is 256×256 pixel images. Although nearest neighbor interpolation and reflection fill mode was used, it is possible that some valuable data may have been lost during downscaling and upscaling of the images.

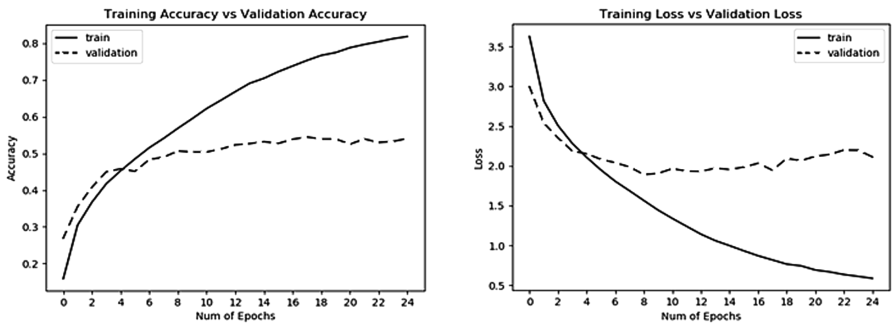

Fig. 2. Train and validation progress of the evaluator through the 25 training epochs, trained only on the original CIFAR-100 dataset.

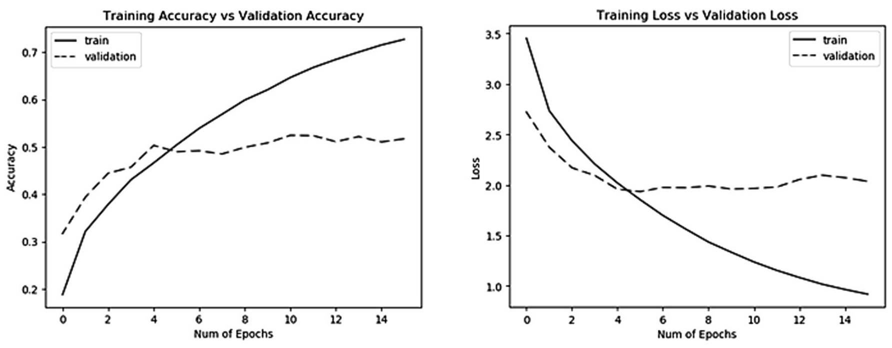

Fig. 3. Train and validation progress of the evaluator through the 15 training epochs, trained on the CIFAR-100 dataset combined with CIFAR-100 augmented images.

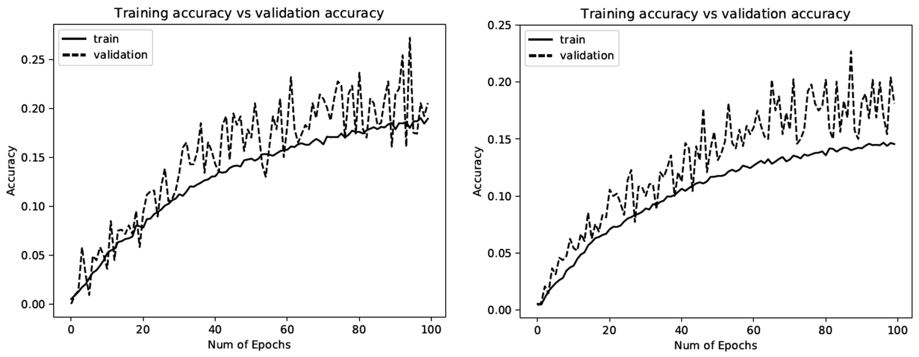

Fig. 4. Comparison between the two models (the left one trained on the original Tiny-ImageNet-200, and the right one trained on a combined dataset) regarding the train and validation accuracy of the evaluator through the 100 training epochs.

4 Conclusion

This paper was an effort to utilize some of the current novel approaches for image generation, with the goal of creating a model which can expand the size of an arbitrary image dataset. Building on top of the work of Dumoulin et al., Ghiasi et al and Jackson et al., two CNNs are trained together to form a complex model capable of applying a randomized style to the input image, by influencing the normalization parameters via the random style embedding. The model is being trained by employing a pretrained VGG19 network in order to extract the content and style from the input and output images. The total loss function is then the combination of content loss (average distance between the high-level semantic features), and style loss (average distance of the Gram matrices between the low-level semantic features). The trained model is later used to augment images from the CIFAR-100 and Tiny-ImageNet-200 datasets, and two identical evaluator models are trained and evaluated for comparing the learnability between the regular datasets and the augmented ones.

The most important conclusion of this paper is that embedding randomized vectors from a previously learned distribution into an encoder-decoder model via conditional instance normalization achieves random stylization in the generated images. By normalizing the feature maps and adding style noise to them, we produce some sort of creativity in the model. This creativity is positively correlated to the augmentation strength α, such that when $\alpha = 0$, the augmenter model should be simply performing the identity function with respect to the input image, while when $\alpha = 1$, the generated image should be random, i.e. its style is 100% drawn from the multivariate normal distribution. Albeit the results were unimpressive, there is much room for improvement. As future work, with enough resources, the potential problems outlined in the previous section can be improved, such as longer training and performing hyperparameter optimization (mostly for the scalar λ_s in the loss function, and the augmentation strength α). Image augmentation with fast neural style transfer seems very promising, and as shown by Jackson et al., such models can be very effective for increasing the size of image datasets.

References

1. Dumoulin, V., Shlens, J., Kudlur, M.: A learned representation for artistic style. arXiv preprint arXiv:1610.07629 (2016)
2. Gatys, L.A., Ecker A.S., Bethge M.: A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576 (2015)
3. Ghiasi, G., Lee, H., Kudlur, M., Dumoulin, V., Shlens, J.: Exploring the structure of a real-time, arbitrary neural artistic stylization network. arXiv preprint arXiv:1705.06830 (2017)
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
5. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1501–1510 (2017)
6. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
7. Islam, M., Murase, K., et al.: A new weight freezing method for reducing training time in designing artificial neural networks. In: 2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat. No. 01CH37236), vol. 1, pp. 341–346. IEEE (2001)
8. Jackson, P.T., Atapour-Abarghouei, A., Bonner, S., Breckon, T., Obara, B.: Style augmentation: data augmentation via style randomization. arXiv preprint arXiv:1809.05375 (2018)
9. Kornblith, S., Shlens, J., Le, Q.V.: Do better imagenet models transfer better? In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2661–2671 (2019)
10. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
11. Li, J., Gadde, R., Ginsburg, B., Lavrukhin, V.: Training neural speech recognition systems with synthetic speech augmentation. arXiv preprint arXiv:1811.00707 (2018)

12. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
13. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet andthe impact of residual connections on learning. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
14. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: the missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)
15. Yanai, K.: Unseen style transfer based on a conditional fast style transfer network (2017)