Neural Style Transfer for Data Augmentation

Siddhesh Shinde

Department of Information Technology

B.E. Xavier Institute of Engineering

siddhesh1598@gmail.com

Abstract

The idea of style transfers to be used as data augmentation can be done to generate more images which closely resembles to our original dataset. These generated images preserve the shape and semantics and randomizes the texture, contrast and color. Furthermore, we can decide to transfer most influential style feature so as to increase the robustness of out convolution neural network(CNN). Also transforming style features from multiple images while we retain the original content, we can develop more number of images similar to our dataset.

Keywords: *Deep Learning, Convolution Neural Networks, Data Augmentation, Style Transfer*

1. Introduction

Availability of ample of quality data and a good neural network model are the prime factors which decide the accuracy of prediction in Machine Learning/Deep Learning. We can create a state-of-the-art deep neural network model but if we do not have sufficient data to train on, we cannot acquire good results. The model will fail to recognize important features from the sparse dataset and hence the we will be left with poor results. The idea of data augmentation is used to generate data similar to our dataset. In data augmentation, we perform some geometrical changes in the current dataset whilst retaining their original characteristics so as to generate new images which have similar characteristics to our original dataset images.

But since we are retaining the original characteristics, the problem of over fitting can still arise where our model is used to capture the low-level visual features of images in our dataset. So to overcome this drawback, we can use Neural Style Transfer(NST) to avoid over fitting our model. NST is used to imbibe features of one image into another so as to generate whole new image which have more features now. We can generate more data to train our neural network.

2. Literature

2.1 Data Augmentation and various approaches to it

Traditional approach to data augmentation uses methods like horizontal reflection, vertical reflection, cropping, scaling, alternating intensities of RGB color channel, etc., to generate new images. It was found that the error rate can be significantly reduced when data augmentation was used on supervised learning and by maintaining the depth of the neural network [1].

Another approach towards data augmentation can be to use current state-of-the-art GANs to generate new images which uses game-theory method where generator generates new images to deceive the discriminator. But the there is a threshold to how many images GAN can generate which have the original features. After certain level, it is not possible for GANs to keep producing new images. Also the amount of computational power required by GAN to generate a new image which resembles the images in our original

dataset is very high. For a simple dataset like 28x28 gray scale MNIST digits, it takes about 10k epochs to finally generate an images that cannot we easily distinguished from the original dataset. So for more complex datasets the GANs will be power hungry to generate images which cannot be easily distinguished as fake or real [2].

2.2 Style transfer

Neural Style Transfer is a method to transfer features of one image to another so as to mix styles of two images. Style Transfer takes 3 inputs – the style image as the artwork whose style we want to transfer, the content image as the picture that we want to transfer the style onto and a noisy image.

Convolution Neural Network is a deep layer neural network where very layer in it extracts a feature from the training images. Later on while testing, the test images are checked for these features and if they contain those features than that associated neuron is activated. The lower(starting) layers of a CNN model extracts basic features like lines, spirals, circles, etc., and as we go deep into the network, the layers extract more complex features. This idea serves as the basis to the style transfer.

Although in style transfer we do not train our model, instead we try to improvise our initial blank image to have features that we extract from other images.

3. Proposed System

3.1 Working of Style Transformation

We define a single CNN model (usually VGG-19) through which we will pass all 3 of our images as shown in **Figure 1.**

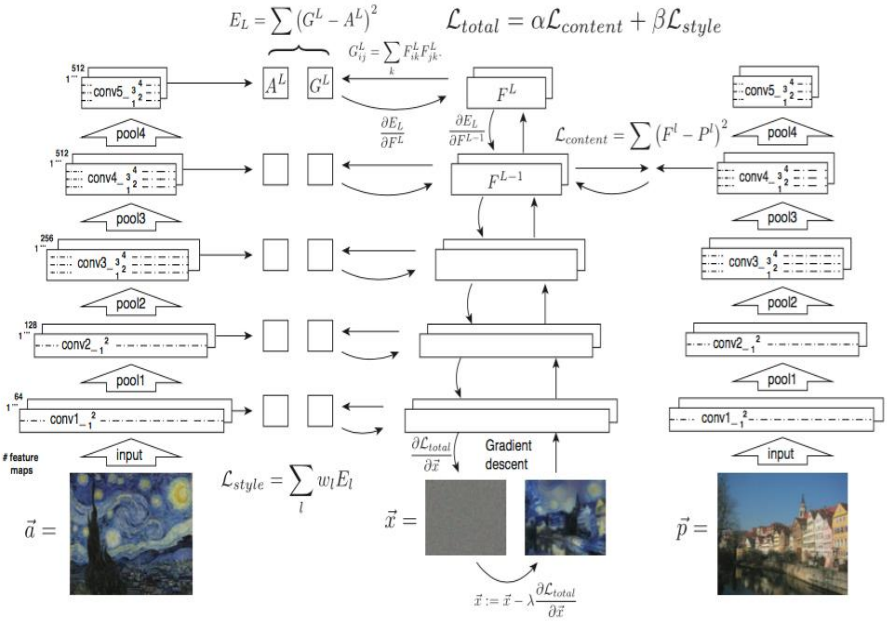

Figure 1. CNN model for generating new images using Style Transfer [3]

For when we pass our style image through the CNN and Gram Matrices are calculated for every layer in the model. Next when we pass our noisy image in through the model, we compute the Gram Matrices at every layer and compare it to the Gram Matrices of style image.

The Gram Matrix loss is given by:

$$G^l_{i,j} = \sum_k F^l_{i,k} F^l_{j,k} \qquad (1)$$

where,
G^l_{ij} is the inner product between the vectorized feature maps *i* and *j* in layer *l*

Let $a\square$ and $x\square$ be the original style image and the image that is generated, and A^l and G^l their respective style representation in layer *l*. The contribution of layer *l* to the total loss is then

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2 \tag{2}$$

and the total style loss:

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l \tag{3}$$

where *wl* are weighting factors of each layer.

The content image is also passed through the CNN model and the content loss is calculated between the content image and the noisy image.

Layer *l* in the network has *Nl* distinct filters (feature maps) each of size (the height times the width of the feature map) M^l. Let $p\square$ and $x\square$ be the original content image and the image that is generated, and P^l and F^l their respective feature representation in layer *l*. We then define the squared-error loss between the two feature representations

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^l - P_{ij}^l\right)^2 \tag{4}$$

$F^l_{ij} \in \mathrm{R}^{Nl \times Ml}$ is the activation of the *ith* filter at position *j* in layer *l*.

Finally, the total loss function which we get is:

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x}) \tag{5}$$

where α and β are the weights for content and style respectively.

Now we have to minimize this loss function with respect to our pixels in noisy image. On each iteration the pixel values will be updates and after some 100 iterations, we will get our generated output which is a mixture of our content image and style image.

We can fine tune these weights according to our need to vary the amount of style and content features in our output image. We need to make a trade off while updating the pixels of our noisy image to how much we want out generated image to match the content versus how much do we want to match the style. Also resizing the style image before computing its gram matrix will lead to different types of features in our output image.

Now that we have features from multiple images in our generated image, it avoids the chances of over fitting our model. While trying to transfer style from style image, we need to make sure that we do retain the original content of our content image. If we imbibe a lot of style features, then we can risk losing important content features.

3.2 Finding maximally activation regions from style image

We can use occlusion experiment or saliency maps to detect which pixels from the style image have more impact in activating the neuron. So we can try to transfer those styles since we have to limit the number of style features in out generated image.

Occlusion experiment used the method of masking certain part of an image before feeding it to the CNN and thus getting a heat map of all the masked regions. In this way, we can figure out which region of the image has more impact in neuron activation.

Saliency Maps compute the gradient of class score with respect to every pixel in the image, take the absolute value and max over RGB channels to figure out which pixel matter the most for classification. We use guided backpropagation to pass only the positive values of the ReLU(activation function) to find only positive influences in the image.

Having a knowledge of which pixel in our image have maximum impact in activating a neuron help us to decide number of features we need to transfer. Sometimes increasing the number of style features will not have any significant improve in our generated image and it will accompany with heavy computational cost. So we need to consider carefully about what proportion of style features we need to transfer.

On knowing the pixels which cause most impact on the activation of the neuron, we can select those pixels while generating the gram matrix of our style image so that we can retain maximum style features. Suppose the on plotting the heat map of our style image giving us information on which pixels are more important, we can select two most important pixels and then considering them as i and j for calculating the gram matrix.

3.3 Using features from multiple style images

We can make use of different features from multiple images so as to add a variation in features. To do that we need to extract feature matrices of all the style images using predefined CNN and the take a weighted sum of all these features. It will then result to a single feature matrix from which certain features can be transferred. This method will help us to further remove the possibility of over fitting as it contains features of multiple images. Furthermore, we can vary the proportions of the features extracted as well as number of style images used to generate a single feature matrix in order to generate more number of style transferred images.

Though they are limitations to this method. It is still limited to a fixed selection of style images to be trained on. Also it requires a lot of time to train such images [4].

4. Uses and Characteristics

Using Style Transfer, we can increase the size of our dataset in multiple folds. Also the data generated do not make our model over fit on the same low-level features. It uses features from multiple images to avoid over fitting and generation good quality data. We can tweak the hyper parameters to try variation of desired output images which we like to generate. It gives us ample number of combinations to try from. But we have to find an equilibrium between adding more style features and retaining content features.

Furthermore, to increase the dataset, we can use Style Transfer alongside with traditional data augmentation. So if we have our original dataset, we can apply Style Transformation to generate more data. Then using traditional data augmentation on the previous dataset and the generated data to increase our dataset drastically.

To go a level higher, we can introduce GANs after Style Transformation to generate more images and then later perform traditional data augmentation over all these data.

Although implementing all these methods at once can generate ample of images but they come with a price of high computational cost and time required to generate these images.

5. Conclusion

Data availability has been an issue for a long time to get good accuracy even in state-of-the-art models. Style transfer helps to add features like color, contrast, texture, etc., from other images into a content image whilst preserving the features of our content image while not only generates new images, but also these images do not make the model over fit. Imbibing these subtle low-level visual features from images in same dataset along with traditional data augmentation can help us to achieve even more accuracy using the same model but with a cost of computing time requires to generate these new images.

But given that synthetic data are always easier to obtain than original data so this paper suggests that on computing the most impactful features and transfering them on other image along with fine tuning some hyperparameters, we can generate images similar to our dataset images with optimum computation cost.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. ACM Communications, 60(6):84–90, 2017.

[2] https://towardsdatascience.com/generative-adversarial-networks-for-data-augmentation-experiment-design-2873d586eb59

[3] https://www.renom.jp/notebooks/tutorial/image_processing/neural-style-transfer/notebook.html

[4] Brandon Cui, Calvin Qi and Aileen Wang. Multi-style Transfer: Generalizing Fast Style Transfer to Several Genres