CSCI 1430 Final Project Report: Cartoon Network In Real Life

Wayland Mafia: Logan Bauman, David Doan, Ben Fiske, Michael Foiani.
Brown University

Abstract

Popular in today's generation, humans often find different forms of representing who they are online. Whether this be in context of SnapChat bitmojis, or Xbox Kinect avatars, humans find themselves creating a representation the strikes a balance between the content, themselves, and the style, the form of the cartoon, in order to represent themselves online to others. In this project, we introduce a deep learning model, a Convolutional Neural Network, that creates a style transfer to the content images. This network utilizes neural representations of the separate input images, and recombines content and style of arbitrary images resulting in a model that is capable of the reconstruction of stylized images. This creates a way for humans to represent themselves more accurately in particular styles to others.

1. Introduction

We wanted to automate the process of turning real life people into cartoons. Previous techniques we have learned about in class to combine images are more naive, and won't adequately solve this problem. Combining the high pass and the low pass could get us an image of a person with some cartoony stuff around it, but in order to actually have the human picture adopt the style of the cartoon image, we need a newer approach.

To solve this problem, we decided to implement Neural Style transfer. With this approach, we can theoretically do style transfer with an image of a human face, and an image with a cartoon style, and wind up with an image that has the content of the humans face, but that is drawn in the style of a cartoon. As long as we pick a good choice for the cartoon image, we should be able to produce images that still contain the originals humans face, but with a very cartoony style. If we succeed in this project, people with no artistic background will have the ability to create images of themselves, or of their friends with the style of their favorite cartoon, or really of any image in general once we implement the style transfer.

2. Related Work

To begin, we started by reading the popular style transfer reference paper linked here, by Bethge, Ecker, and Gatys: `https://arxiv.org/pdf/1508.06576.pdf`. The paper outlines a way to combine the content of one image with the style of another by using a network pretrained for only a classification task (like VGG18). [1]

3. Method

Our problem was to implement style change mapping a cartoon style onto a human face. Our chosen approach was to implement style change in general, and then try to tune our model to work well for the cartoon task. For our methodology, we built off of that in the Bethge, Ecker, and Gatys paper. We used the VGG18 architecture and weights, a CNN trained only for scene classification, and we only used the convolution portion of the network with no classification head.

Instead of running input through the CNN, the model functions as follows: the generated image starts as the content image and then is iteratively updated through a number of steps to minimize the combination of a content loss function, measuring difference from the content in the content image, and a style loss function, measuring difference from the style of the style image. These loss functions are computed by looking at the output of several selected layers in the CNN when the generated image, the content image, and the style image are run through the model. The selected layers are the first ones in each convolution block.

3.1. Loss function

The total loss function is a weight of the content and style loss.

$$L_{total} = \alpha * L_{content} + \beta * L_{style}$$

To find each respective loss, we develop the 'target" and minimize their differences from the image created.

The "target" is found by running the inputted content and style images through the vgg16 network and storing the output from each desired layer.

The loss, in general terms, is some difference metric (square mean in our case) between this target and the image being progressively generated over the epochs. Minimizing this loss will theoretically result in an image that represents a good middle ground between the content and style targets, where this middle ground can be influenced in either direction by alpha and beta parameters.

We chose the content to be represented from only one block layer in the vgg network, 'block5_conv2', so this loss doesn't require a Gram Matrix.

$$\mathcal{L}_{content} = \frac{1}{2}\sum_{i,j}(F_{ij} - P_{ij})^2$$

Where F_{ij} is the activation of the jth position in the ith filter in the last layer of the CNN.

We chose our style to be represented from 5 layers:

['block1_conv1', 'block2_conv1', 'block3_conv1', 'block4_conv1', 'block5_conv1']

Therefore, we needed to use a Gram Matrix to consider each combination of the 5 style layers. Then, we iterate over each layer, apply them to and sum their losses to get the total loss.

With the content loss, we want the features from the generated image to be the same as in the content image, so we calculate the euclidean distance between their feature maps. In the style loss, we care less about about the value of exact features in the feature map, but rather the non-local style of the image. That is why we use the gram matrix, which can be thought of as computing the correlation between different feature levels. That way, the style loss doesn't care about the pixel values at specific locations like the content loss does, so the content loss can ensure that the values it cares about are correct, while the style loss fills in the rest to make the style correct.

3.2. Hyperparameters

Learning rate: 0.0001. We originally used a higher learning rate but after doing some comparisons found that a lower rate produced smoother results and avoided style taking over from content.

Epochs: Most of our output images were produced using between 1000 and 5000 epochs. Using more than 5000 epochs would sometimes lead to loss of some of the content elements of the images, although changes slowed down significantly the more epoch there were. Epochs also depended highly on the learning rate chosen.

Alpha and **beta** are the weighting factors in the total loss calculation between content loss (α) and style loss (β). We chose $\alpha = 0.05$ and $\beta = 5$.

4. Results

We exhibit a gallery of images generated by or model (Figure 2) and the evolution of an image over training starting with the content image (Figure 3). We also can see what happens when images are overtrained (Figure 4); note the similarity to the style images (Figure 1).

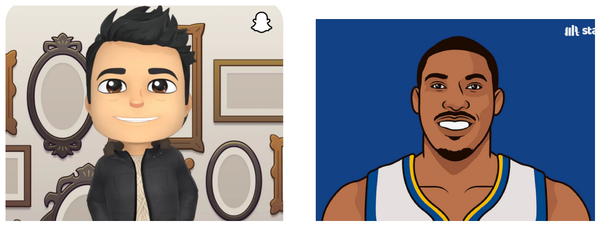

Figure 1: Two of the style images we used

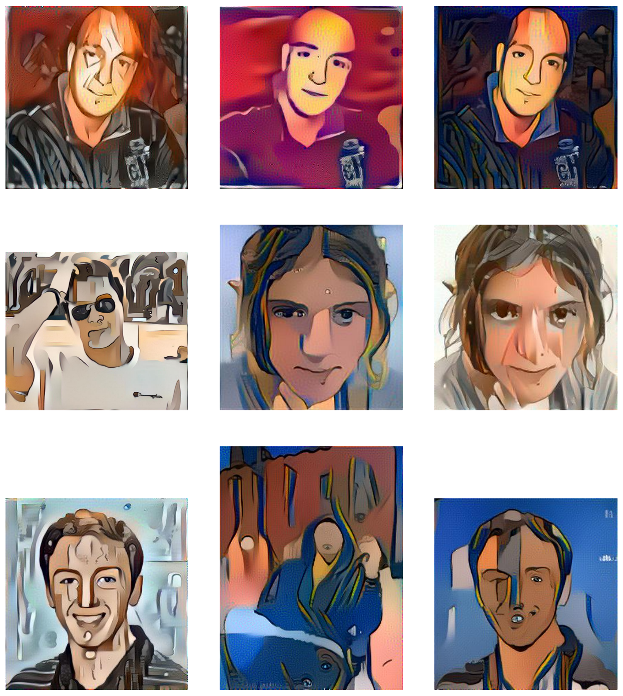

Figure 2: A gallery of result images generated using our model

4.1. Technical Discussion

Although we have results, and we can see changes that occur after every 100 epochs, because we are using a neural network, we still are unsure of the exact way that the network itself is combining the content and style images. Although we know that they style is weighed more than the content, in addition to the fact that we know the loss functions that go

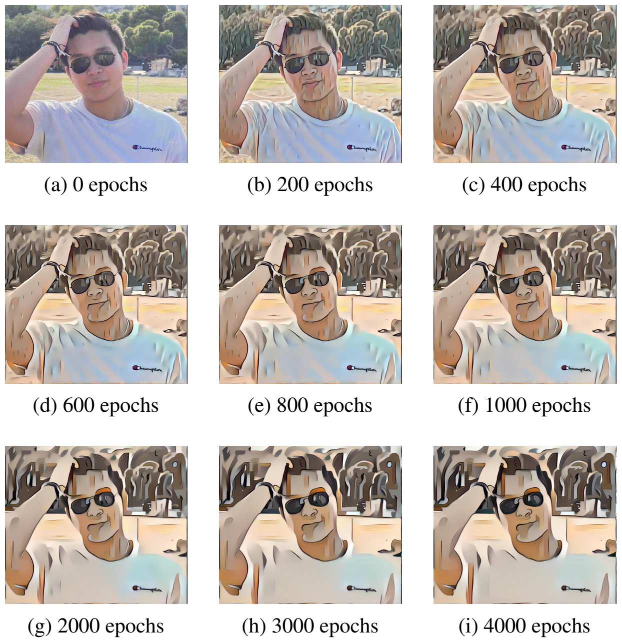
(a) 0 epochs (b) 200 epochs (c) 400 epochs (d) 600 epochs (e) 800 epochs (f) 1000 epochs (g) 2000 epochs (h) 3000 epochs (i) 4000 epochs

Figure 3: The evolution of an image during training of 4000 epochs

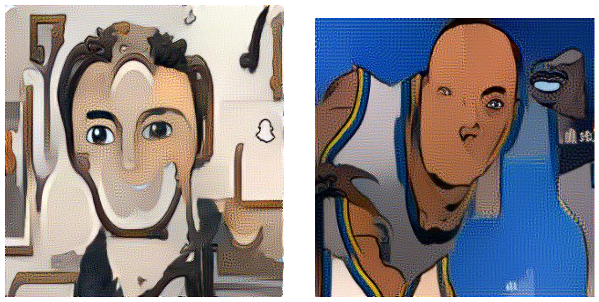
(a) 330000 epochs (b) 50000 epochs

Figure 4: Results when a higher than optimal number of epochs were used (style took over from content)

into determine the loss of the combined images to the style and content images, we can not pinpoint the exact process. Through the result we can slowly see how the image is being created, and see the style dominate the picture, and the content slowly being incorporated a specific style, but we have observed the eyes be incorporated first, then the rest of the face. Although this does give us insight into how humans process of styling themselves as a cartoon image, we cannot explain the process in its entirety. I think the right way to think about it is that we gain insight into how humans may process this task, and much like neural networks are a black box, we can use this as a tool to assist our understand rather than a tool to explain the human process.

A key consideration for us in our hyper parameters and how we calculated our loss was the trade off between getting a cartoonized version of an image, but also not get so cartoony that we lose enough of the content image so we can't see the original person, which is the whole point of our project. For example, in choosing what layer of the VGG16 to use in the calculation of or content loss. When we incorporate deeper layers, the style quickly starts to take over, but we still retain higher level image qualities, like the general shape of the face, or general locations. When we incorporated less deep layers, we kept way more of the low level features and the original face stayed recognizable. This difference is proof of the fact that deeper layer of a CNN tend to keep track of the more high level features of an image, and less to the finer details. The deeper the layers are, the more extreme the change is between the content image and the style, one second it looks a lot like the content image, and a couple of epochs later it is all style. The more shallow layers we incorporate, the more of a middle ground we get between the two pictures. The two approaches lead to wildly different types of result images. While we ended up turning in code that only uses the lowest layer for the content loss, the results we get from incorporating layers from the more shallow part are still really interesting, and if we were to deploy this project for anyone to use, we would probably want to give people an option to incorporate what ever layer they want to get a wider variety of interesting results to chose from.

Figure 5: An image created using the first layer in the CNN. Notice that more fine details remain in the image than in the cartoon ones, even though they are faint in this example.

4.2. Assessment

We believe we achieved the goals we set out to accomplish, and our model is able to successfully transition photographs into images in a cartoony style. Currently, our model needs some tuning for each image produced; for example, some content images work better with certain style images and there is not a standard number of epochs that produces the best images. When we had cartoon that matched the general form of the human image, we were able to get significantly better results then when it didn't. When the images watched up well, we could see that throughout training, the features of the content image slowly transform into cartoonized versions of itself. However, small differences in the picture usually stayed they way they were in the content image. For example, in the picture of David in his sunglasses, the cartoon had eyes instead of sunglasses, but the model knew not to transform the sunglasses into eyes. When images don't align, we find that the cartoon style still transfers but the content of the image gets distorted, as you can see in the bottom right image in figure 2. A bit more fine-tuning could get the model to work in a production-suitable manner.

5. Societal Discussion - Critique Response

Response to concern 3: we moved away from caricature and to a more simple cartoon style change, which addresses this issue because the algorithm is no longer aimed to exaggerate any facial features. It also makes it unnecessary to encode anything about the prominence of any features in the model.

Response to concern 4: we can address the copyright issue by making sure that any style images used in the production version are fully owned by us. Since only one style image needs to be used to create each output image, and they can be reused, very few are needed in total, a benefit of this implementation. We could even tailor a style image to exactly fit our needs.

Response to concern 5: The ownership of the output image will reside with us, the creators of the model, with the ability for the user to use the image freely for any personal use. If this were implemented in production, we could also consider charging a fee to purchase ownership of the image for commercial or other uses. If such a fee were charged, we could vet such purchases and make sure the images used in them were ethically sourced and not infringing on another person's privacy.

Response to concern 7: our model is deterministic and can't be reverse engineered (to find the content image), unless the person has the style image, and with the style image it still may not be possible since information will be lost in the transformation. Even if it were possible to reverse engineer the initial image, our project is not geared toward identity protection so this is not a major issue.

Response to concern 8: Although we created the model, we are not providing a platform on which the images will be used. Perhaps our cartoon images would be popular with bot accounts, but there are plenty of real images to steal already out there, so our model does not contribute to the problem any more. It is also the responsibility of the platform images are used on to monitor their use as they see fit.

6. Future Directions

In the future, we could improve our model by making it more generalizable to different images without requiring tweaks (like using different numbers of epochs fro different images). This would be necessary if we were to use it in a production application. We could also try to create or find a style image that works well for any content image, or specifically design several to impart a few different cartoon styles. We also found that the cartoonization worked best when the content and style loss were aligned. This gave us the idea that we could perhaps add an extra step to this process where we identify the face of the content image first, and only run the style transfer on that part of the image. That way we could get better style transfer, and also not distort and change the whole background of the content image. That way, we could get any image, and cartoonize just the face in it, and perhaps get cooler, better results.

7. Conclusion

In culmination, with our implementation of loss function, our model successfully accomplished our goal of combining the content of one image with the style of another. With tuning to focus on cartoon style images in particular, in addition to the hyperparameter tuning and finding images that match we were able to create satisfactory outputs. After testing with different, style images and parameters, we were able to conclude that we can take almost any human portraits and convert them into a cartoonized version of themselves.

References

[1] Matthias Bethge Leon A Gatys, Alexander S Ecker. A neural algorithm of artistic style. 2015. 1

Appendix

Team contributions

Ben Contributed to progress reports and critique, wrote main training loop code, troubleshooted and updated loss functions, fixed bugs to get project to run for the first time, contributed to refining image outputs, contributed to final report.

Logan Did research on different ways to cartoonize in image, worked on implementing the loss functions and the model architecture. I also helped I also worked on tuning the hyperparameters and making lots of adjustments to our model to actually get our model showing images with good style transfer instead of gibberish. Implemented a lot of solutions that we ended up optimizing away, but showed that our model was capable of producing good images that allowed us to continue moving forward in our approach.

David Contributed to progress reports. Helped in the implementation of the loss functions. Contributed to optimizing the code, testing the model, and improving functionality. Helped with hyperparameter tuning and poster creation, in addition to the final report. Helped with project management, and supporting other's tasks.

Michael Contributed to progress reports. Helped in the implementation of the loss functions, as well as commentating and refactoring the code. Contributed to optimization. Helped to test the model, and creating output images and graphs for the poster and report. Helped improving functionality and hyperparameter tuning, as well as components of the final report.