Image Style Transfer

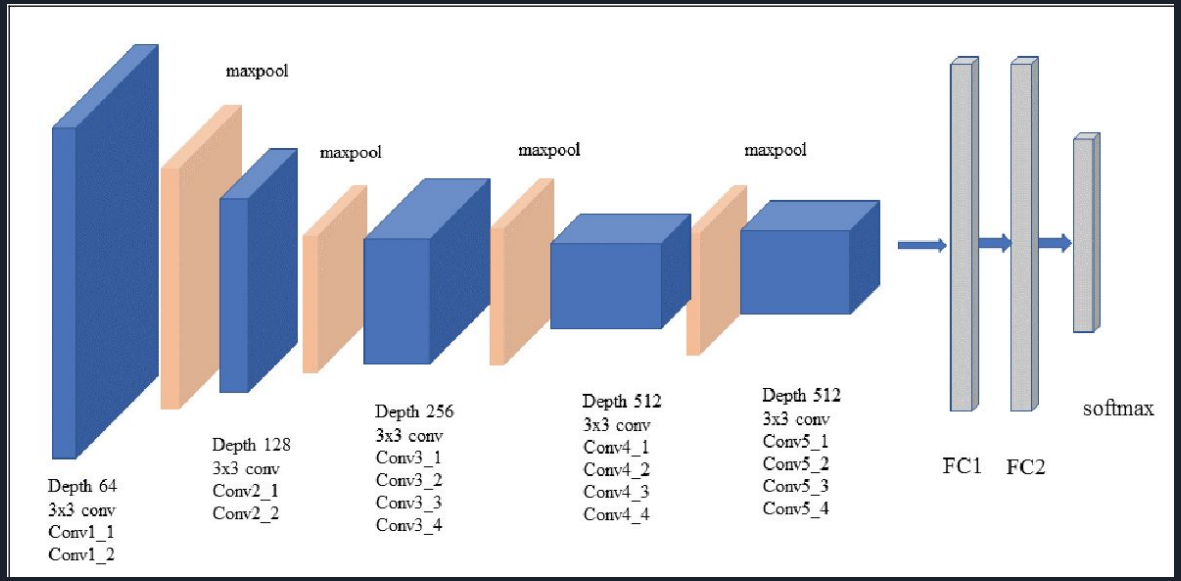
Mateo Parrado
Zesheng Liu

Goals

Architecture

-One of the models we use is the VGG19 architecture for our neutral net

-This is a pre-trained 19 layer deep CNN designed to classify images into 1000 categories



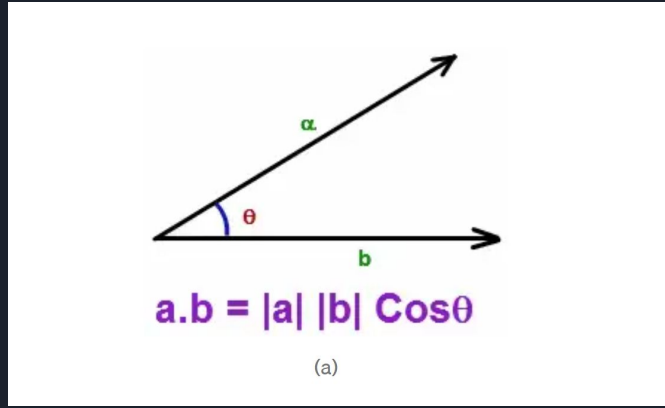
Loss for image

$$Loss = \alpha Loss_{style} + \beta Loss_{content}s$$

-Content Loss: defined as the mean squared difference between the generated image and the original image

-Style Loss: defined as the mean squared difference between the gram matrix of the generated image and the gram matrix of the original image

-What is a gram matrix? It is a way of estimating the similarity between two sets of features



Loss for Video

-Use same loss functions as above but add one more:

-Temporal Loss: defined as mean squared difference between two consecutive generated frames

-the goal is to minimize artefacts between frames and keep the drawn style as consistent as possible

Framework we use:

-JAX

Autograd+JIT

What is JAX?

JAX is Autograd and XLA, brought together for high-performance machine learning research.

With its updated version of Autograd, JAX can automatically differentiate native Python and NumPy functions. It can differentiate through loops, branches, recursion, and closures, and it can take derivatives of derivatives of derivatives. It supports reverse-mode differentiation (a.k.a. backpropagation) via `grad` as well as forward-mode differentiation, and the two can be composed arbitrarily to any order.

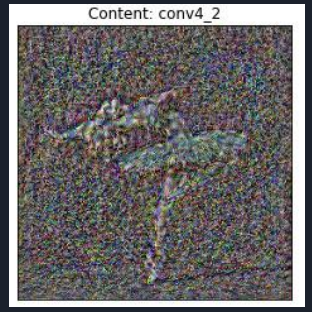
Framework we use:

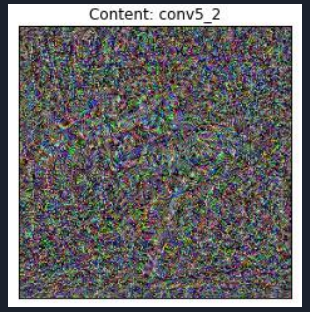
-FLAX: A neural network library and ecosystem for JAX designed for flexibility

-Flaxmodels: A collection of pre-trained models in Flax, by Matthias-wright

-Optax: A gradient processing and optimization library for JAX, provide optimizers and Huber loss

Results: What each layer represents for content





Training:

-Initialize with N(0,1)
-Adam with lr=1e-3
-2000 epochs
-Set style loss weight be 0 and content loss weight be 1

Results: Different chosen layer for style representation

Chosen Layers: Conv1_1, Conv2_1, Conv3_1, Conv4_1, Conv5_1

Chosen Layers: First 5 convolution layers

Training:

-Initialize with N(0,1)
-Adam with lr=1e-3
-2000 epochs
-Set style loss weight be 1 and content loss weight be 0

Results: Style/Content Tradeoff

Content weight = 1

Content weight = 10

Content weight = 100

Content weight = 1000

Content weight = 10000

Content weight = 100000

Training:
-Initialize with content image
-Adam with lr=1e-2
-2000 epochs
-Set style weights be fixed and content loss weight be 1,10, 100, 1000, 10000, 100000

Results: Style/Content Tradeoff

Style weight = 1

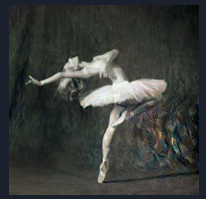
Style weight = 10

Style weight = 100

Style weight = 1000

Style weight = 1e4

Style weight = 1e5



Style weight = 1e6

Style weight = 1e7

Style weight = 1e8

Training:
-Initialize with content image
-Adam with lr=1e-2
-2000 epochs
-Set content weights be fixed and style weights be 1 to 100000000

Results: Different optimizer

Adagrad 1e-2

Adam 1e-2

RmsProp 1e-2

SGD 1e-2

Adagrad 1e-3

Adam 1e-3

RmsProp 1e-3

SGD 1e-3

Training:

-Initialize with Content image
-2000 epochs
-Set style loss weight be 1e5 and content loss weight be 1

Results: Different Loss

Huber Loss

L1 Loss

MSE Loss

SmoothL1 Loss

Training:

-Initialize with content image
-Adam with lr=1e-3
-2000 epochs
-Set style loss weight be 1e5 and content loss weight be 1

Results: Different Pretrained Model

ResNet 18

ResNet 34

ResNet 50

VGG16

VGG19

Training:

-Initialize with content image
-Adam with lr=1e-3
-2000 epochs
-Set style loss weight be 100000 and content loss weight be 1

Photorealism regularization

Formally, we build upon the Matting Laplacian of Levin et al. [9] who have shown how to express a grayscale matte as a locally affine combination of the input RGB channels. They describe a least-squares penalty function that can be minimized with a standard linear system represented by a matrix $\mathcal{M}_I$ that only depends on the input image I (We refer to the original article for the detailed derivation. Note that given an input image I with N pixels, $\mathcal{M}_I$ is $N \times N$). We name $V_c[O]$ the vectorized version ($N \times 1$) of the output image O in channel c and define the following regularization term that penalizes outputs that are not well explained by a locally affine transform:

$$\mathcal{L}_m = \sum_{c=1}^{3} V_c[O]^T \mathcal{M}_I V_c[O] \qquad (2)$$

Ref: Deep Photo Style Transfer

$$Loss = \alpha Loss_{style} + \beta Loss_{COntent} + \gamma L_m$$

Preserve the structure and make it photorealistic.

Total variation loss

Total Variation Regularization. To encourage spatial smoothness in the output image $\hat{y}$, we follow prior work on feature inversion [6,20] and super-resolution [48,49] and make use of *total variation regularizer* $\ell_{TV}(\hat{y})$.

Denoise and make the output smooth

$$\sum_{i,j} |x_{i,j} - x_{i+1,j}| + |x_{i,j} - x_{i,j+1}|$$

Ref: Perceptual Losses for Real-Time Style Transfer and Super-Resolution

Results: TV loss and Real Loss

$$Loss = \alpha Loss_{style} + \beta Loss_{COntent} + \gamma L_m + \delta L_{TV}$$

Training:

-Initialize with content Image
-Adam with lr=1e-2
-8000 epochs
-Set style loss weight be 1e6 and content loss weight be 1 RL to be 100 and TV to be 0.01

Original Content image

Results: TV loss and Real Loss

$$Loss = \alpha Loss_{style} + \beta Loss_{cOntent} + \gamma L_m + \delta L_{TV}$$

Without Regularization

RL Loss

RL Loss+TV Loss

Results: Larger Scale

Divide into 2 overlap patches

Results: Larger Scale

+

Use our project 4 code: Homography

Results: Larger Scale

Stitched Result

Result of whole content input

Current Results (Video)

-Currently very short (2 frame) videos are training, but longer videos take a very long time to train and therefore we do not have results for them

-One proposed solution is to break up a longer video into a collection of smaller 2 to 5 frame overlapping videos

-We could then train the model on each of these videos and take the average of the results for each frame

Next Steps

- Generate final submission for image style transfer
- Optimize video processing to allow it to finish in a realistic amount of time
- Fine tune video processing, experiment with different optimizers, loss functions, and architectures