Neural Style Transfer for Videos

Stefan Himmel

Kyle Meredith

Motivation

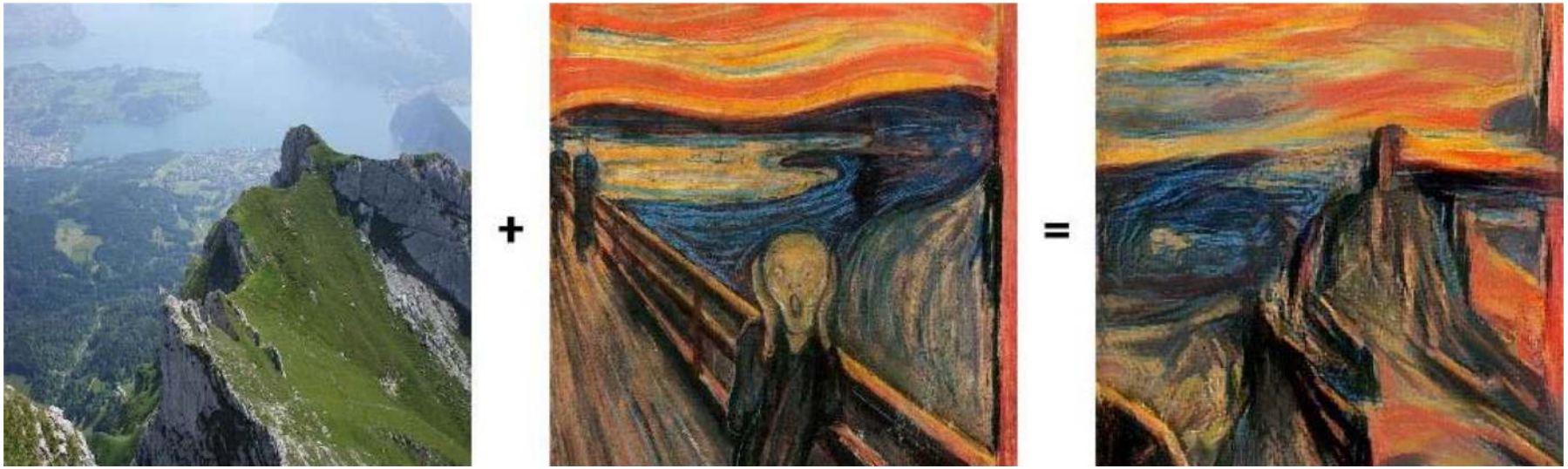
- To learn about machine learning, specifically Convolutional Neural Networks (CNN)
- To create generative art and apply our learning to a creative purpose

Problem Statement

1) Implementation
    - Learn how to build and train convolutional neural networks
    - Specifically, implement a real-time style transfer for videos using Pytorch
1) Innovation
    - Use existing neural style transfer methods to create novel generative art
    - Specifically, adapt single frame transfer system to support style videos

Neural Style Transfer

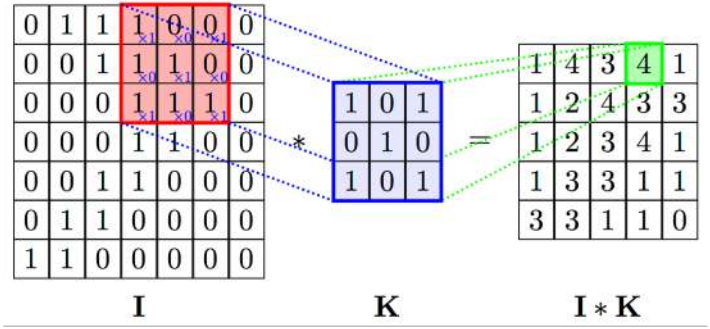
“Content” “Style”



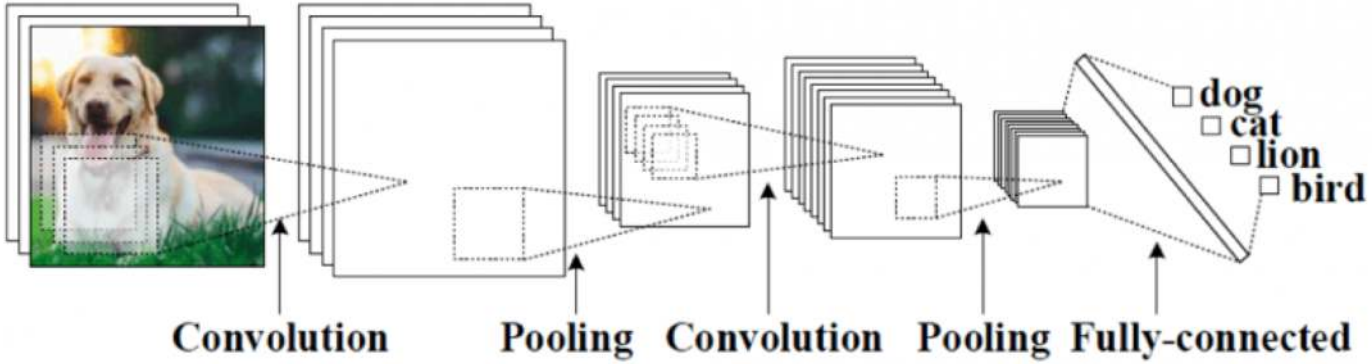
https://medium.com/data-science-group-iitr/artistic-style-transfer-with-convolutional-neural-network

Technology

- Convolutional Neural Networks
  - Terminology: channel, tensor, convolution, filter, activation
  - Types of layers: convolution, pooling, fully-connected
  - Commonly used for image classification



https://github.com/PetarV-/TikZ/tree



https://www.ayasdi.com/blog/artificial-intelligence/using-topological-data-analysis-understand-behavior

Neural Style Transfer Architecture

- Loss functions
  - Content loss
    - MSE between pixels of “content” activation and “generated” activation
  - Style loss
    - MSE between *Gram matrices* of selected “style” and “generated” activations
- Optimization method
  - Uses backprop to directly update the generated input image based on loss
  - Allows arbitrary style images
- Feedforward method
  - Uses feedforward “stylization network,” updating its parameters with backprop from separate loss network
  - Allows real-time stylization for a single style

Neural Style Transfer for Videos

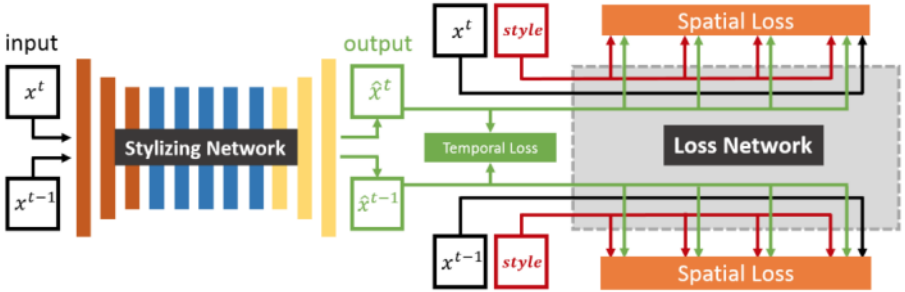
- Research teams have recently managed to apply this technique to videos
- Both optimization (Ruder) and feedforward (Huang) architectures have been developed
- Video

Methods

- Implementation
  - Pytorch implementation of Real-time Neural Style Transfer for Videos (Huang et al.)
  - Uses pre-trained stylization network with temporal loss for smooth real-time transfer
- Invention
  - Python script that leverages Torch implementation of Gatys image transfer system
  - **Input**: Style video and content video **Output**: Generated multi-style video

Implementation

- Architecture
  - Feed-Forward Stylization Network (Custom CNN specified in paper)
  - Loss Network
    - Spatial Loss (pre-trained VGG-19 CNN from Torchvision library)
    - Temporal Loss (MSE between adjacent generated frames warped by optical flow)
- Training
  - 100 videos scraped from videvo.net
  - Used Gattaca GPU with CUDA-based cuDNN library
  - Trained on 80,000 frames for each stylization model (8 hours at 240 x 426)



http://openaccess.thecvf.com/content_cvpr_2017/papers/Huang_Real-Time_Neural_Style_CVPR_2017_paper.pdf

Implementation Demo

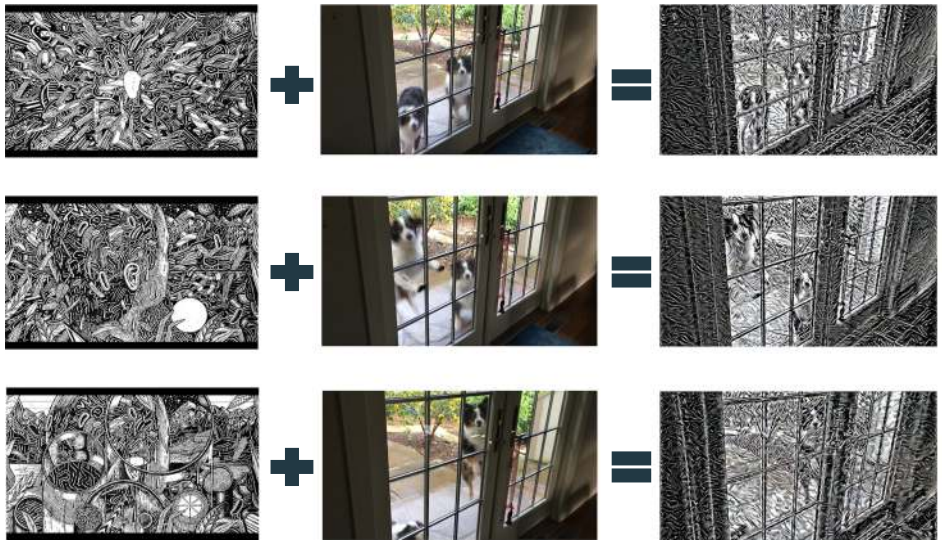


Style Img:http://wallpaper.istriku.site/picasso-famous-paintings-worth/

Invention

- Architecture
  - Pre-made optimization network implemented by Justin Johnson
  - Stylizes 426 x 240 frame in ~2 minutes on Gattaca
- Pseudocode:
  1. Split style and video frames into sequence with openCV
  2. In a loop, feed paths to corresponding frames into neural_style.lua script
  3. Combine stylized outputs into video with ffmpeg

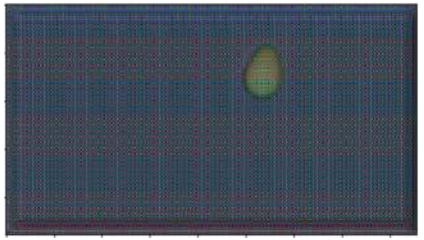
Invention Demo

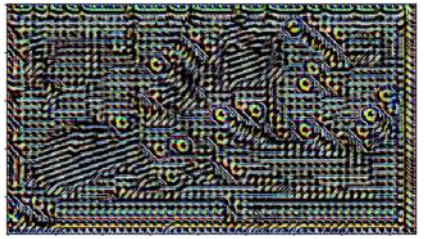


Style video: https://www.youtube.com/watch?v=TuJqUvBj4rE

Challenges

- Implementation
  - Hard to debug before structural completion
  - Slow iteration time (model fully trained in 8 hrs)
  - Only example implementation was flawed
  - Many potential sources of error (frame loading, multifaceted loss, model loading/saving, displaying results)
  - Not enough time to test temporal loss
- Invention
  - Misallocated time on Torch/Lua implementation of video style transfer system by Ruder et. al.
  - Limited time because implementation was the focus





```
for i = 1, #cnn do
  if next_content_i <= #content_layers or next_style_i <=
  #style_layers or next_temporal_i <= #temporal_layers then
    local layer = cnn:get(i)
    local name = layer.name
    local layer_type = torch.type(layer)
    local is_pooling = (layer_type == 'cudnn.SpatialMaxPooling' or
    layer_type == 'nn.SpatialMaxPooling')
    if is_pooling and params.pooling == 'avg' then
      assert(layer.padW == 0 and layer.padH == 0)
      local kW, kH = layer.kW, layer.kH
      local dW, dH = layer.dW, layer.dH
      local avg_pool_layer = nn.SpatialAveragePooling(kW, kH, dW,
      dH):float()
      avg_pool_layer = MaybePutOnGPU(avg_pool_layer, params)
      local msg = 'Replacing max pooling at layer %d with average
      pooling'
      print(string.format(msg, i))
      net:add(avg_pool_layer)
    else
      net:add(layer)
    end
```

Timeline

Week 4 — 10/2-10/9	Begin work on midterm deliverable and determine necessary cloud computing ✓
Week 5 — 10/9-10/16	Continue work on image —> image neural transfer prototype Gather funds for cloud computing ✓
Week 6 — 10/16-10/23	Finalize midterm deliverable and start final deliverable (image —> video) ✓
Week 7 — 10/23-10/30	Complete midterm presentation and continue work on final deliverable ✓
Week 8 — 10/30-11/6	**Midterm presentation is due** Finish code skeleton of final deliverable X
Week 9 — 11/6-11/13	Begin training stylization network for final deliverable X
Week 10 — 11/13-11/20	Finish training at least one model and start to optimize code architecture X
Week 11 — 11/20-11/27	Polish code and complete final presentation (get rid of unused import statements, write all docstrings for classes, write README) X
Week 12 — 11/27-12/4	Finalize code, report, and poster X
12/11	**Final deliverables due**

Week 8 — 10/30-11/6	**Midterm presentation is due** Continue code skeleton of final deliverable ✓
Week 9 — 11/6-11/13	Continue work on code skeleton for implementation, begin work on invention ✓
Week 10 — 11/13-11/20	Continue work on code skeleton for implementation, continue work on invention code ✓
Week 11 — 11/20-11/27	Begin debugging implementation, replan work on invention (no Lua) ✓
Week 12 — 11/27-12/4	Finish debugging implementation, train implementation, process video, write invention script ✓
12/11	Finalize code, write report, and create poster ✓

Conclusion

- Implementation – Despite incomplete functionality, accomplished our goal of learning how to implement CNNs using Pytorch
- Invention – Created proof-of-concept multi-style transfer system
- Future work
  - Fix style loss and debug temporal loss implementation
  - Expand multi-frame for robust functionality and interface

Sources

1. Huang, H., Wang, H., Luo, W., Ma, L., Jiang, W., Zhu, X., Li, Z., Liu, W.: Real-time neural style transfer for videos. CVPR, 2017

2. M. Ruder, A. Dosovitskiy, and T. Brox. Artistic style transfer for videos. In Proceedings of German Conference on Pattern Recognition, 2016.

3. O'Shea, Keiron Nash, Ryan. (2015). An Introduction to Convolutional Neural Networks. ArXiv e-prints.