kptan86@gmail.com

Notable Architecture / Application

1. Transfer learning
2. AutoEncoder
3. U-net
4. Neural style transfer
5. Generative / Generative Adversarial network
6. Variational autoencoder (VAE)

Transfer Learning

- It is often more economical to use pre-trained model rather than starting from scratch
  - (Deep neural network takes significant computational resources to train (weeks with multi-GPU))
- Transfer learning works by **assuming there are similarities in the neuronal weights** between the pre-trained and transferred models
  - (e.g. the first layer should learn the tone of the image, the second layer learns texture etc. These common tune and texture properties are assumed transferable)

Three common approaches to use pre-trained models

1. **Swapping the last layer**
   - Replace the last fully-connected layer from pre-trained model with the intended new layer, then train the weights of neurons in this replaced layer (all other weights kept intact)
2. **Fine-tuning weights**
   - Replace the last layer, but also allow back-propagation to fine-tune all neuronal weights in the network. This is essentially an initialization method.
3. **Use pre-trained "checkpoint" model**
   - Similar to fine-tuning, but using checkpoint or intermediate models as starting points

Discussion

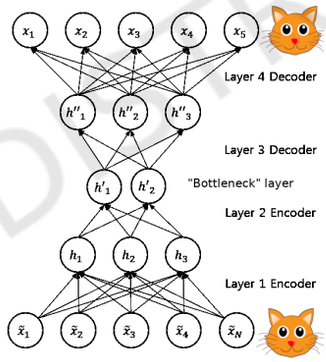
What is the extent of transferabilty of the network learning ?

DO NOT DISTRIBUTE

kptan86@gmail.com

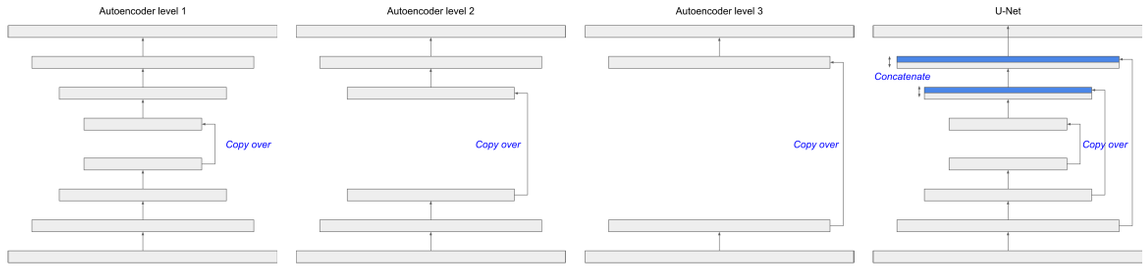
AutoEncoder

- Setting target output equals to input
- The "bottleneck" layer encodes the "minimal representation" of data
- Uses: denoising, sparse encoding, dimensional reduction
  - (e.g. replaces PCA in non-linear space)



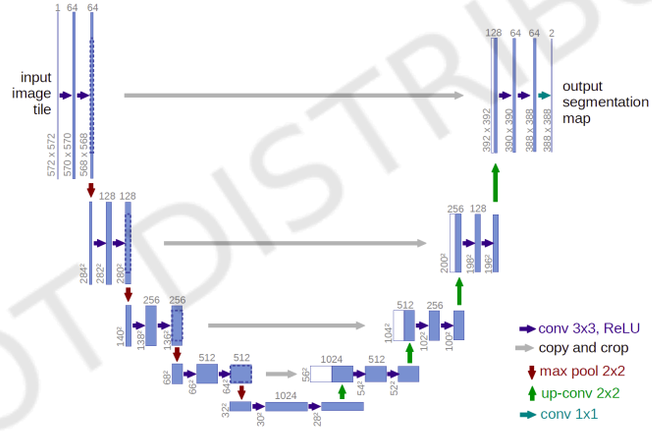
U-net

- The encoding capability of Autoencoder is limited by bottleneck layer
- U-net "enhances" the encoding by combining multiple levels of bottleneck



U-net

- The encoding capability of Autoencoder is limited by bottleneck layer
- U-net "enhances" the encoding by utilizing multiple level of bottleneck



DO NOT DISTRIBUTE

kptan86@gmail.com

Neural style transfer

- An interesting niche application of transferring style (usually texture style) from one image to another

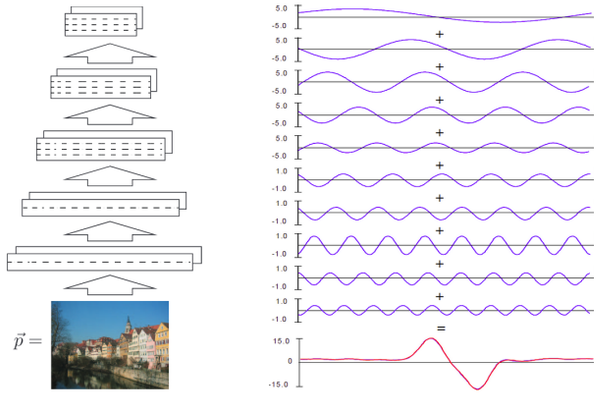
Input Content Image + Input Style Image = Output Image

Gatys et al, 2016

Neural style transfer

- Think of deep CNN as a (hierarchical) framework/basis to "decompose" an image (note: it actually *is*)

Convolutional "decomposition" **Fourier decomposition**



Neural style transfer

Central idea: Construct an image with "content" of one image and "style" of the other under CNN decomposition architecture

In Gatys et al's implementation:

- Content: element-wise integrity (at every level (CNN layer))
- Style: correlations (inner-product) among feature maps (at every level (CNN layer))

kptan86@gmail.com

Neural style transfer

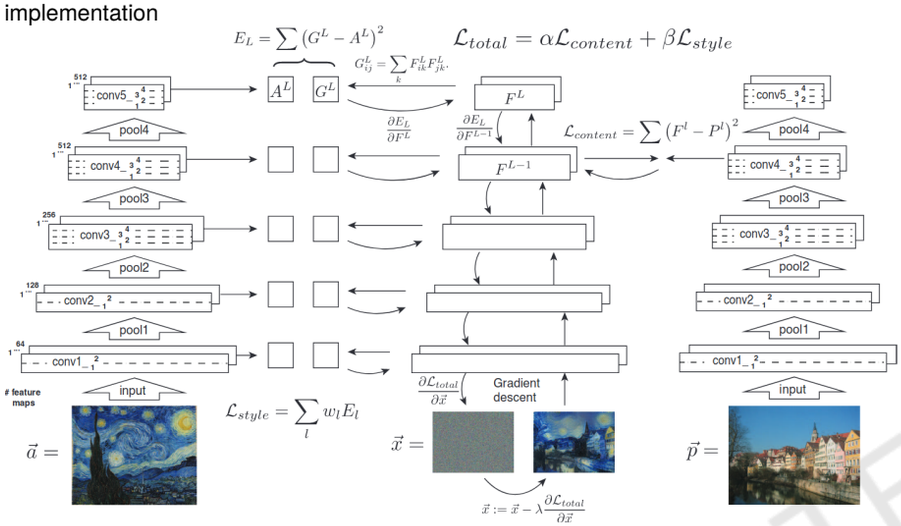
Main steps:

- Choose a CNN architecture (e.g. VGG-19)
- "Decompose" input content image and input style image into layers using the chosen CNN
- Repeat until satisfied:
  1. Generate an output image (e.g. using a generative neural network)
  2. Decompose the image using the same CNN architecture
  3. For every layer*, compute the content loss and style loss
     A. content loss: against input content image
     B. style loss: against input style image
  4. Compute the total loss = weighted sum of content loss and style loss from all layers
  5. (Gradient descent or other optimization technique)

*(*or some layer(s), depending on the implementation)*

Neural style transfer

Gatys et al's implementation



DO NOT DISTRIBUTE