June 6th 2019

MRP – Week#2 Report

I spent this week implementing the code for neural style transfer. Particularly I looked at image processing and VGG network architecture selection

---

Libraries and API's

- **PIL:** allows for opening, manipulating, and saving many different image file formats.
- **Numpy:** allows for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- **Keras:** an open-source neural-network library written in Python. It is capable of running on top of TensorFlow. We will be accessing our VGG-16 network from this library
- **Scipy:** used for scientific computing and technical computing

Image Preprocessing

- Assert fixed dimensions on the length and width of the content and style images. This will make concatenating the images much easier.
- We need to convert the images into arrays for numerical processing
- The images occur in RGB pixel channel format creating the 3 dimensions in addition to length and width (length x width x 3 dimensions)
- We add an additional dimension of 1 to allow us to concatenate the style and content images to generate the final result (1 x length x width x 3 dimensions)
- In their paper *VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION,* Karen Simonyan and Andrew Zisserman suggest that the only preprocessing we do is subtract the mean RGB value, computed on the training set, from each pixel.
- We will be using the publicly available ImageNet training set which has a mean RGB value of (103.939, 116.779, 123.68)
- Lastly, the array values of the images can be converted to keras backend variable for later use
- A placeholder variable was also created for the final generated image

VGG-16 Network

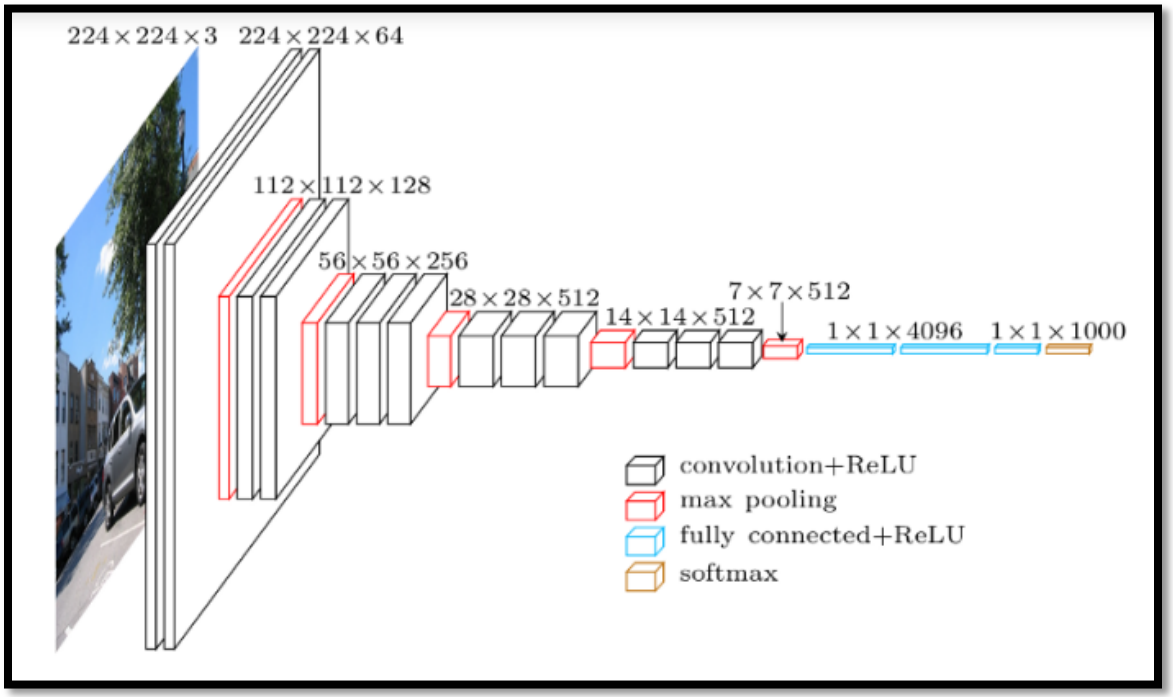
- Keras' allows us to directly call on the VGG-16 model in Python with the following of code

```
model = VGG16(input_tensor=input_tensor, weights='imagenet', include_top=False)
```

June 6th 2019

- **ImageNet** is a large visual database designed for use in visual object recognition software research. More than 14 million images have been hand-annotated by the project to indicate what objects are pictured and in at least one million of the images, bounding boxes are also provided. ImageNet contains more than 20,000 categories with a typical category, such as "balloon" or "strawberry", consisting of several hundred images
- We will be using a VGG-16 network pre-trained on the ImageNet dataset to save us the time and computational stress of “re-inventing the wheel”
- **The key finding of Gatys’ neural style transfer is that a CNN, pre-trained for image classification already know how to encode perceptual and semantic information about images**

VGG-16 Architecture



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

- The chart above breaks down the layers at each stage in the figure above
- The 1st convolution layer of vgg16 uses a 3x3 kernel and trains 64 feature maps to generate an image representation of 224x224x64 by taking a 3 channel image of size 224x224 as input. (see breakdown below)
- All layers follow the same method in different dimensions
- Each layer learns to detect a particular pattern
- For example, a particular layer maybe activated by seeing a circular shape such as a wheel or ring or ball

- Hence, as we go from shallow layers to deeper layers the hidden units become capable to detect more and more complex features from a given image.
- Since we're not interested in the classification problem, we don't need the fully connected layers or the final softmax classifier. We only need the part of the model marked in green in the table above

Breakdown on Layer 1

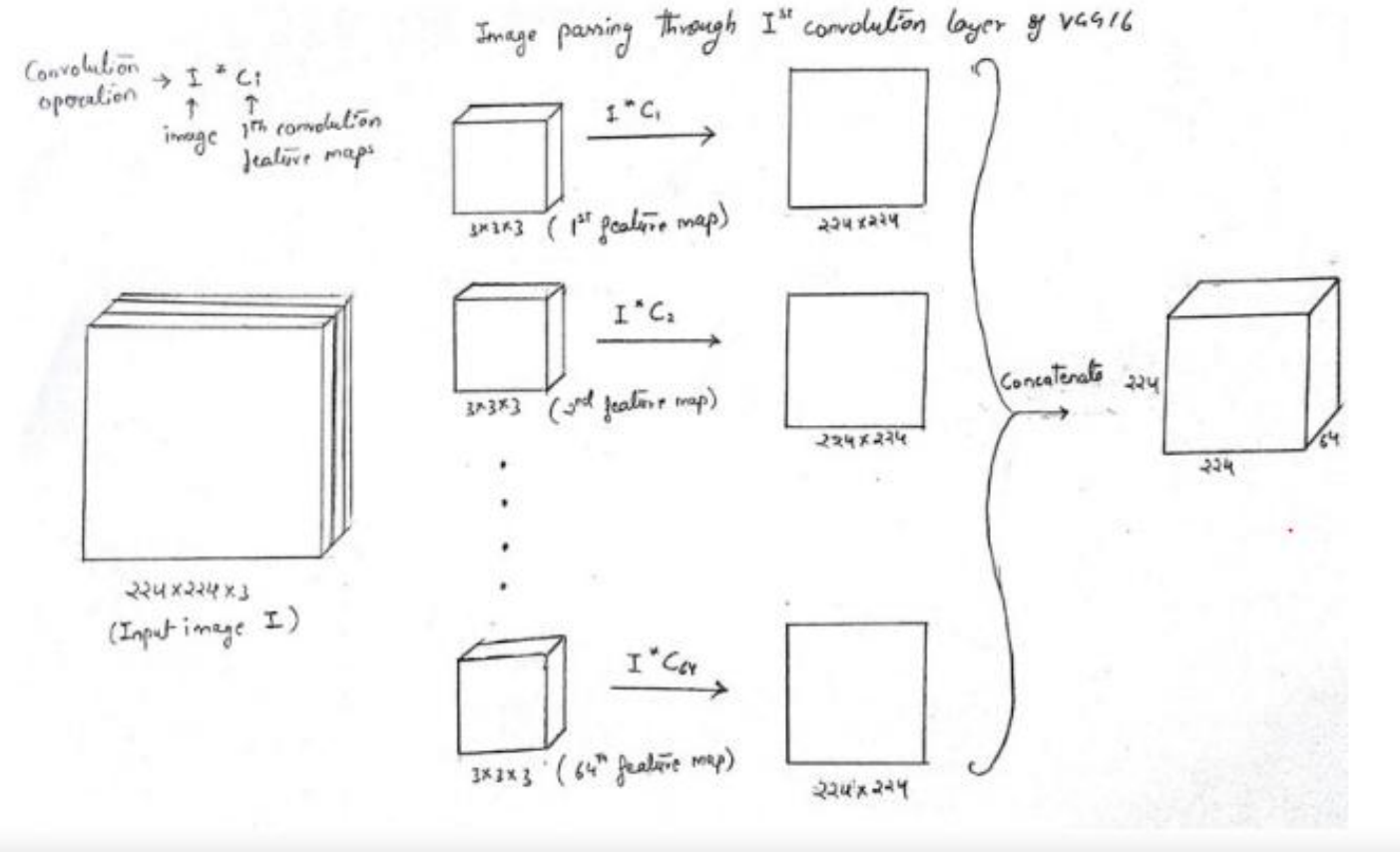


Image source: https://towardsdatascience.com/neural-style-transfer-tutorial-part-1-f5cd3315fa7f