May 31st 2019

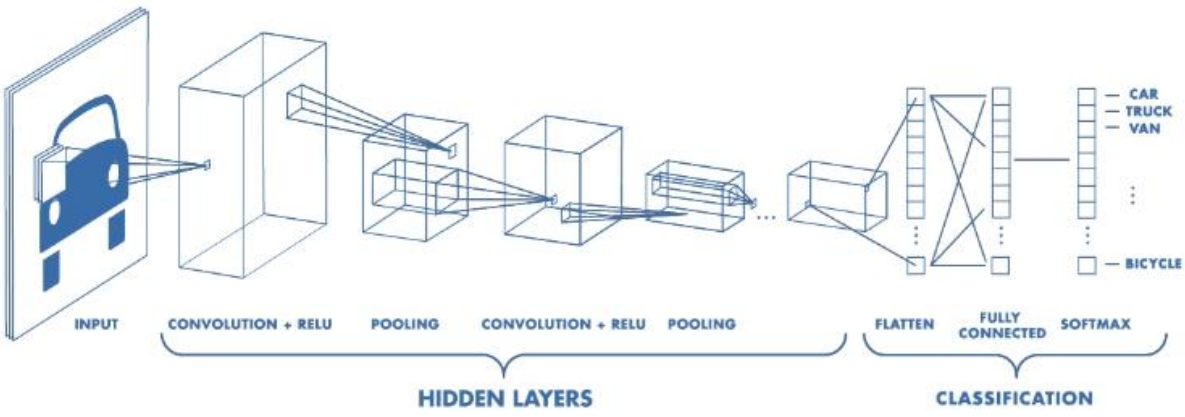
# MRP – Week#1 Report

## I spent this week understanding the background fundamentals of Neural Style Transfer and how it can be implemented in Python.

---

Convolutional Neural Networks (CNNs)

- Great for working with images
- Computers perceive images as 2D arrays of numbers (a.k.a pixels)
- **Regular neural networks:** transform an input by putting it through a series of hidden layers. Every layer is made up of a **set of neurons**, where each layer is fully connected to all neurons in the layer before. Finally, there is a last fully-connected layer — the output layer — that represent the predictions
- **CNNs**: the layers are organised in 3 dimensions: width, height and depth. Further, the neurons in one layer do not connect to all the neurons in the next layer but only to a small region of it. Lastly, the final output will be reduced to a single vector of probability scores, organized along the depth dimension
- **2 Main components**
    - Feature Extraction/Hidden layers – Convolution and Pooling operations to extract features
    - Classification – probability assigned to what the image possibly contains




- **Convolution Layer:** apply a filter as a sliding window over the image
- **Activation Layer:** nonlinear activation functions like ReLu and tanh for machine learning
- **Pooling:** down-sampling of features

Neural Style Transfer

- The CNN extracts information at each layer as a feature map.
- Deeper layers provide more content information
- To extract the style of an image (i.e. the stroke and colors), we calculate the correlation between the outputs of higher level layers. This calculation captures texture information in the image.

May 31st 2019

Python Implementation

- Tensors allow us to store discriminative feature in an array
- Python offers a TensorFlow flow package, ideal for implementing CNNs
- Flow refers to the operations we perform on our arrays such as non-linear transformations
- When you're working with TensorFlow, constants, variables and placeholders come handy to define the input data, class labels, weights and biases.
- We can use an existing VGG-16 network, pre-trained on over 14 million images. (Saves us the pain!)
- The downloads are available on : https://www.cs.toronto.edu/~frossard/post/vgg16/