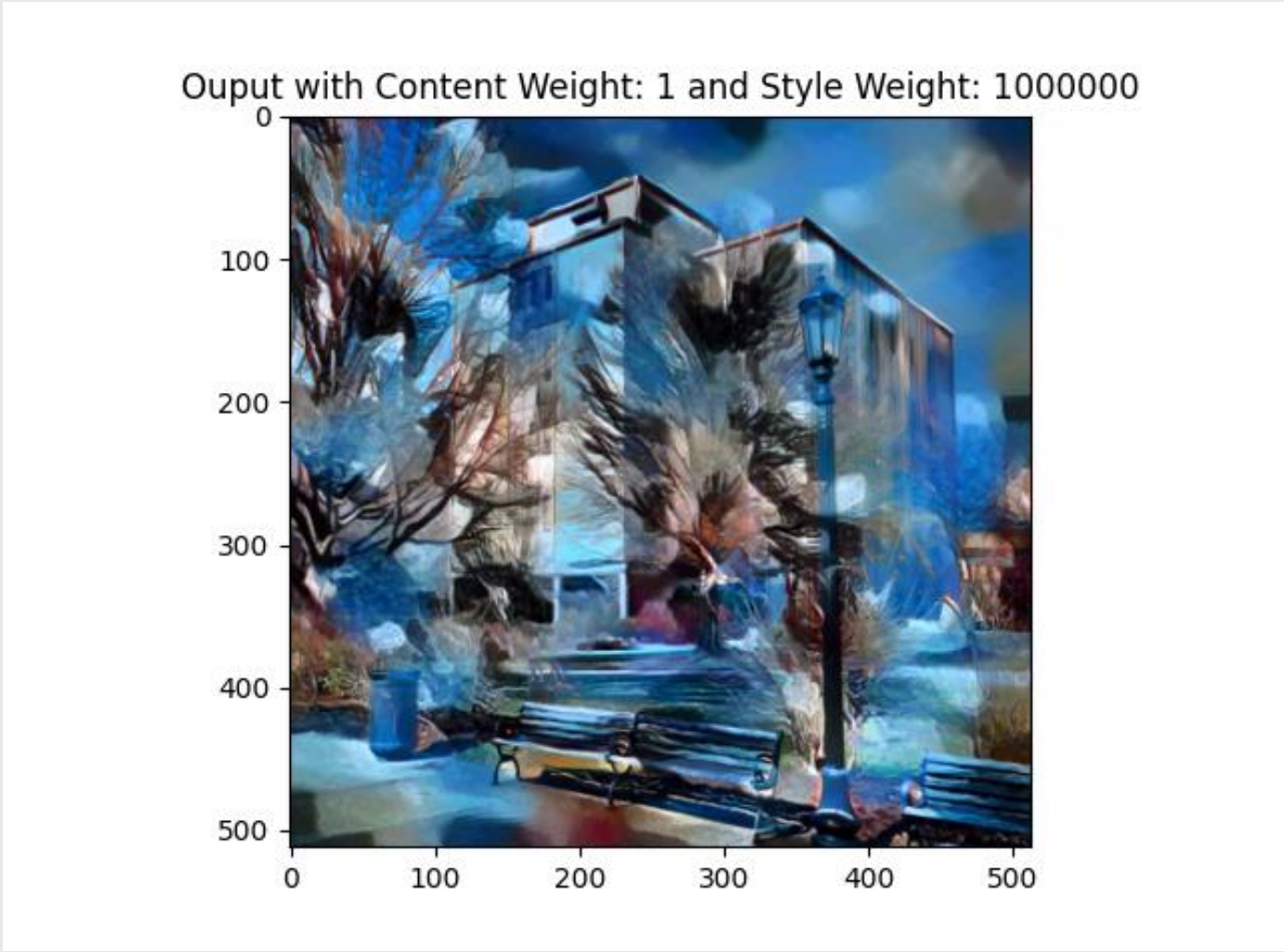
Neural Style Transfer

Jacob Mazurkiewicz

Algorithmic Art

- *Content Image*

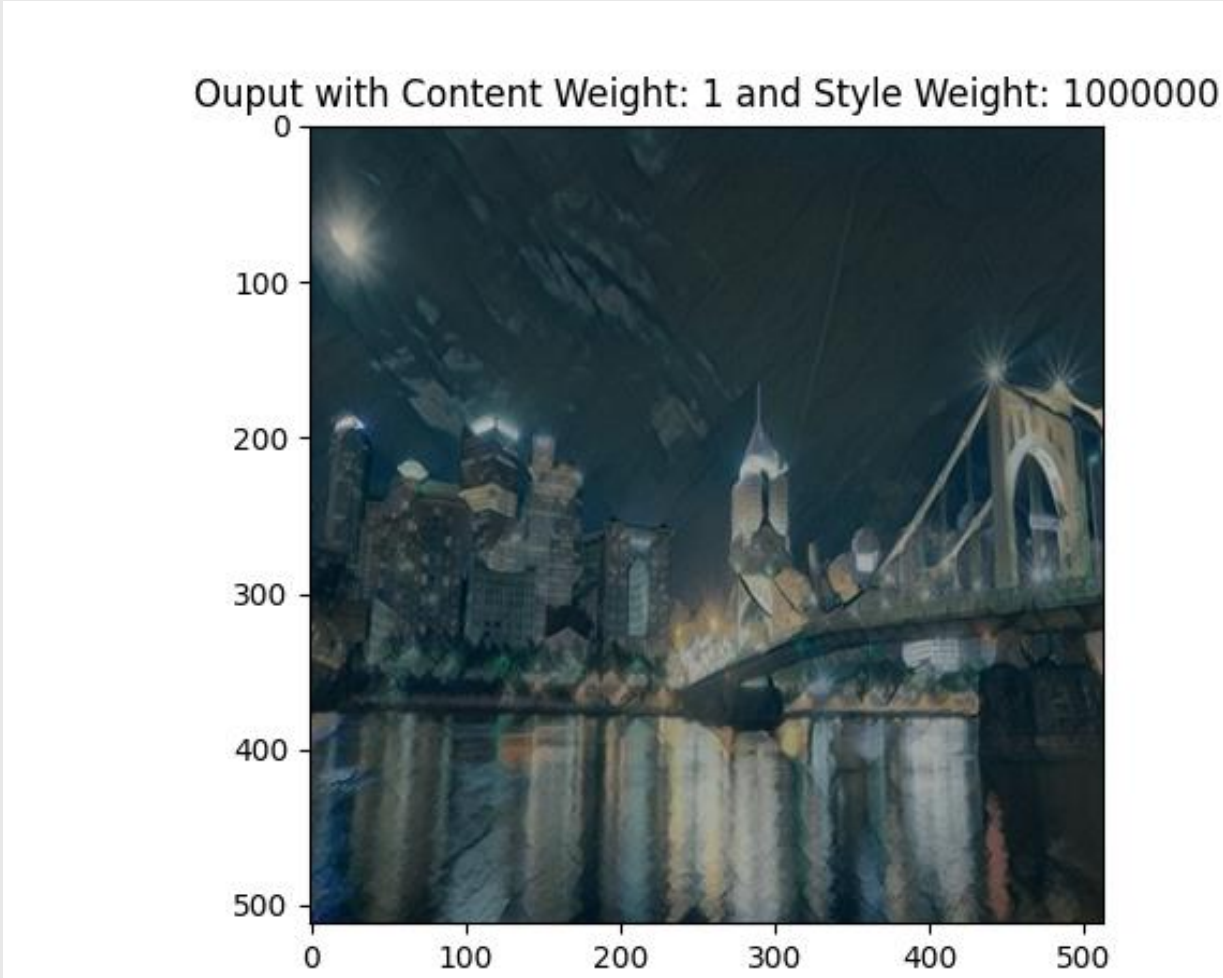
- *Style Image*



Algorithmic Art

- *Content Image*

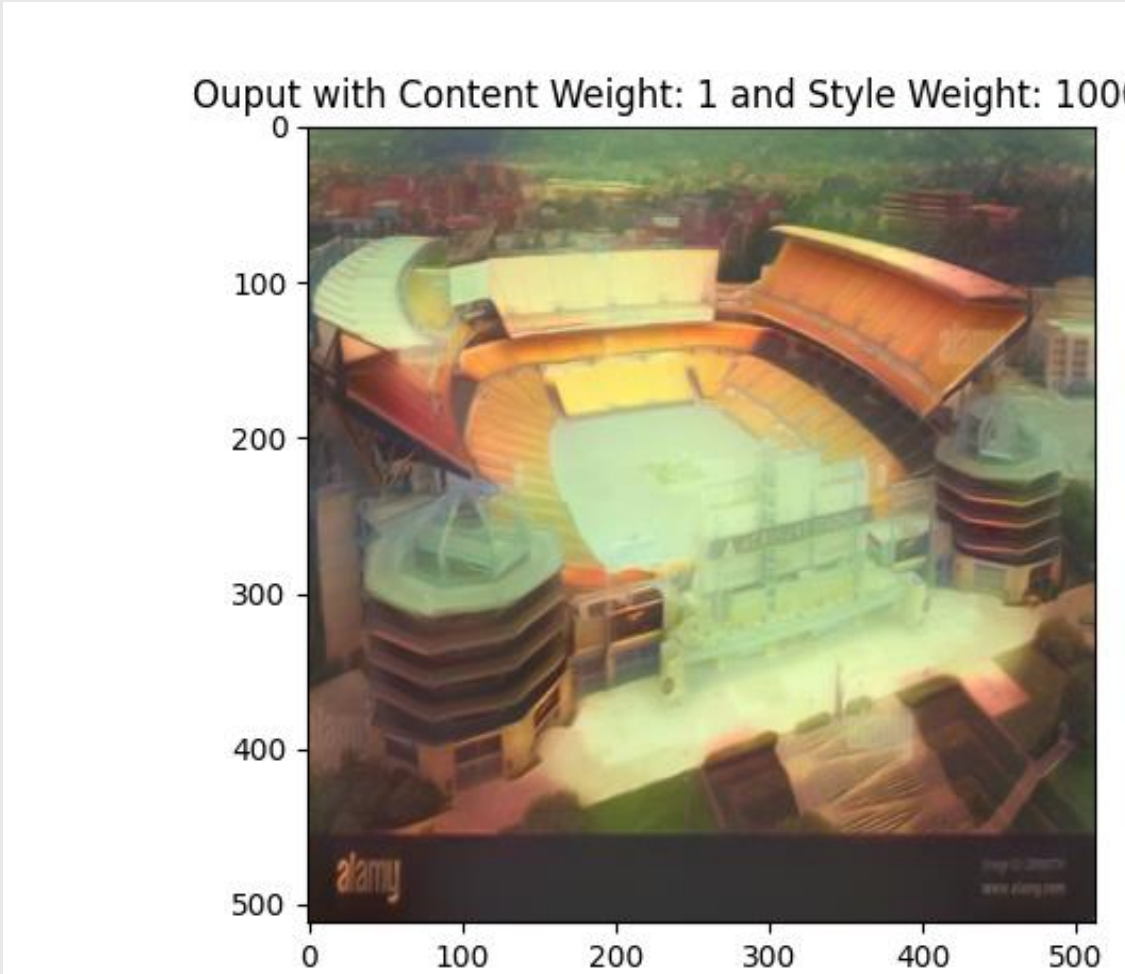
- *Style Image*



Algorithmic Art

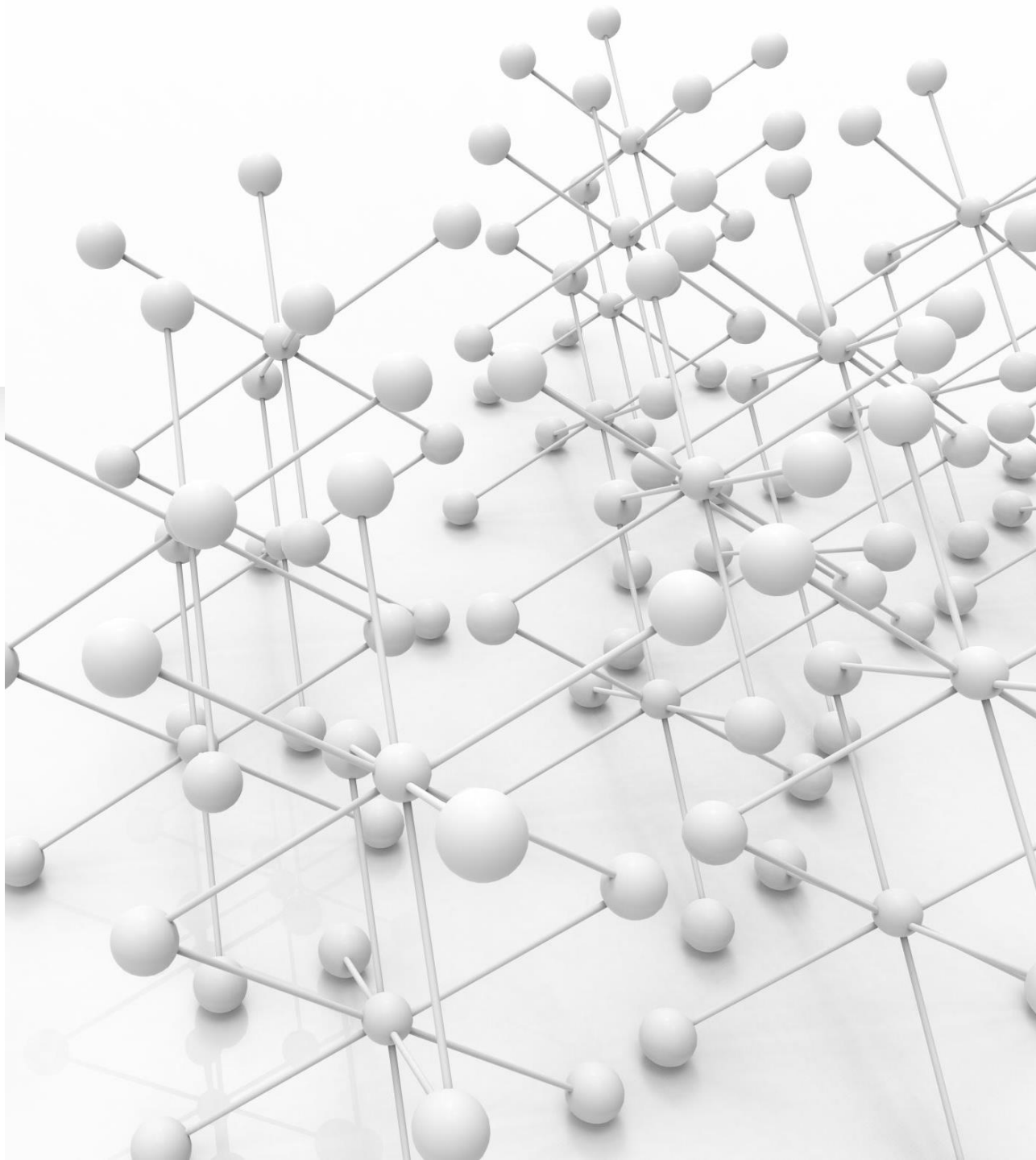
- *Content Image*

- *Style Image*

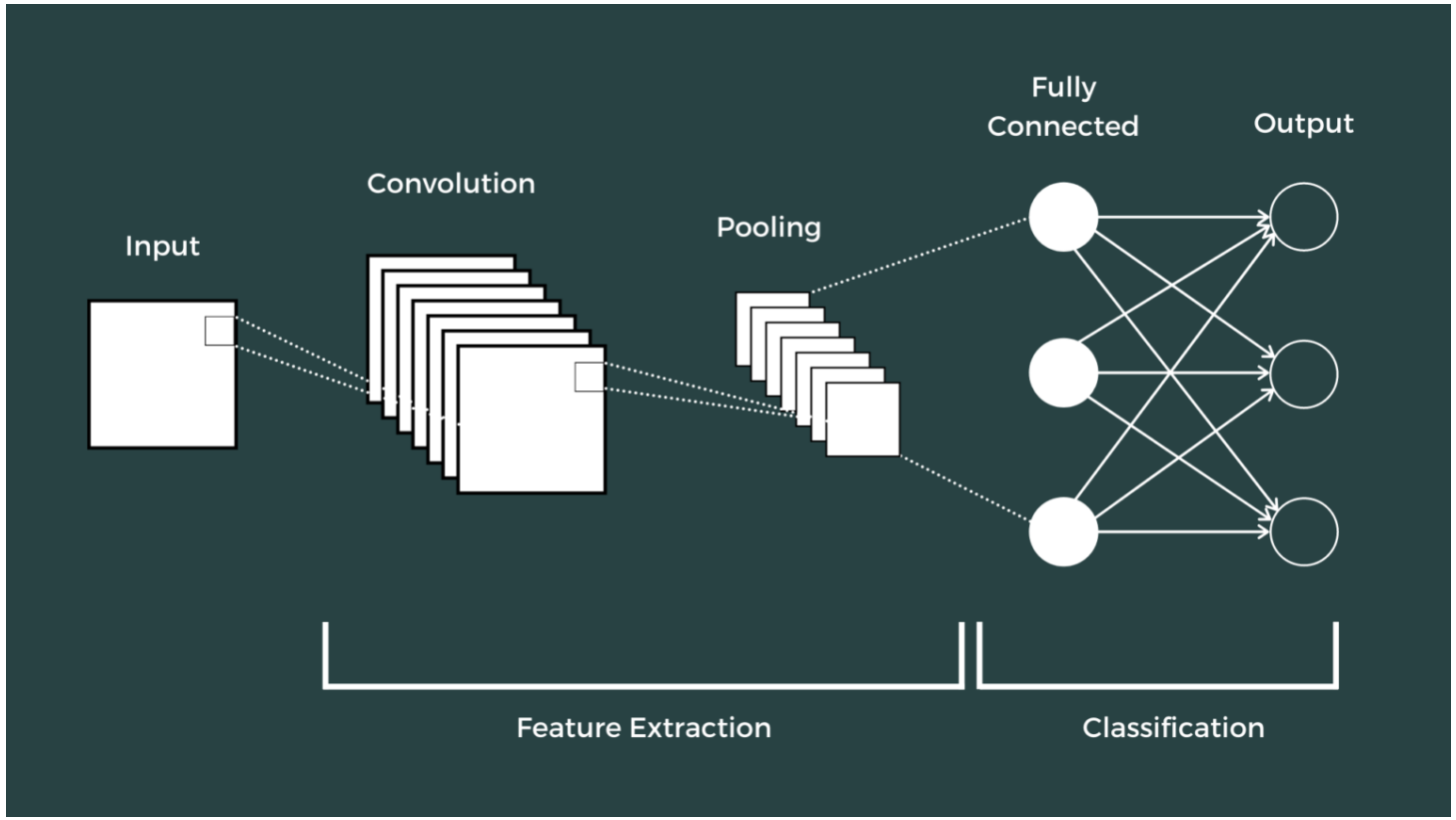


Neural-Style Transfer Algorithm

- Uses convolutional deep neural nets to minimize loss (difference) between a style and content image
- Creates a new image that is a blend of the two

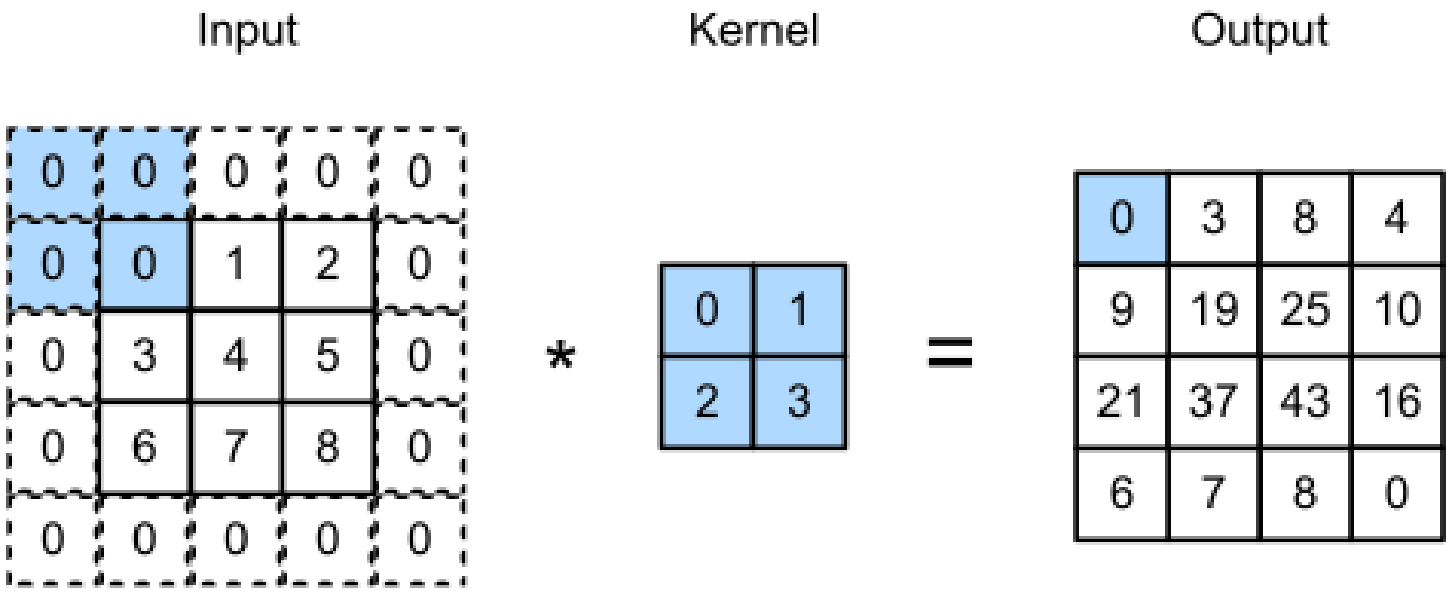


Convolutional Neural Nets



- Neural nets are universal function approximators
- Convolutional layers (introduced by Kunihiko Fukushima)
- Feedforward layers
- Only use convolutional sections in neural-style transfer

Convolutional Layers

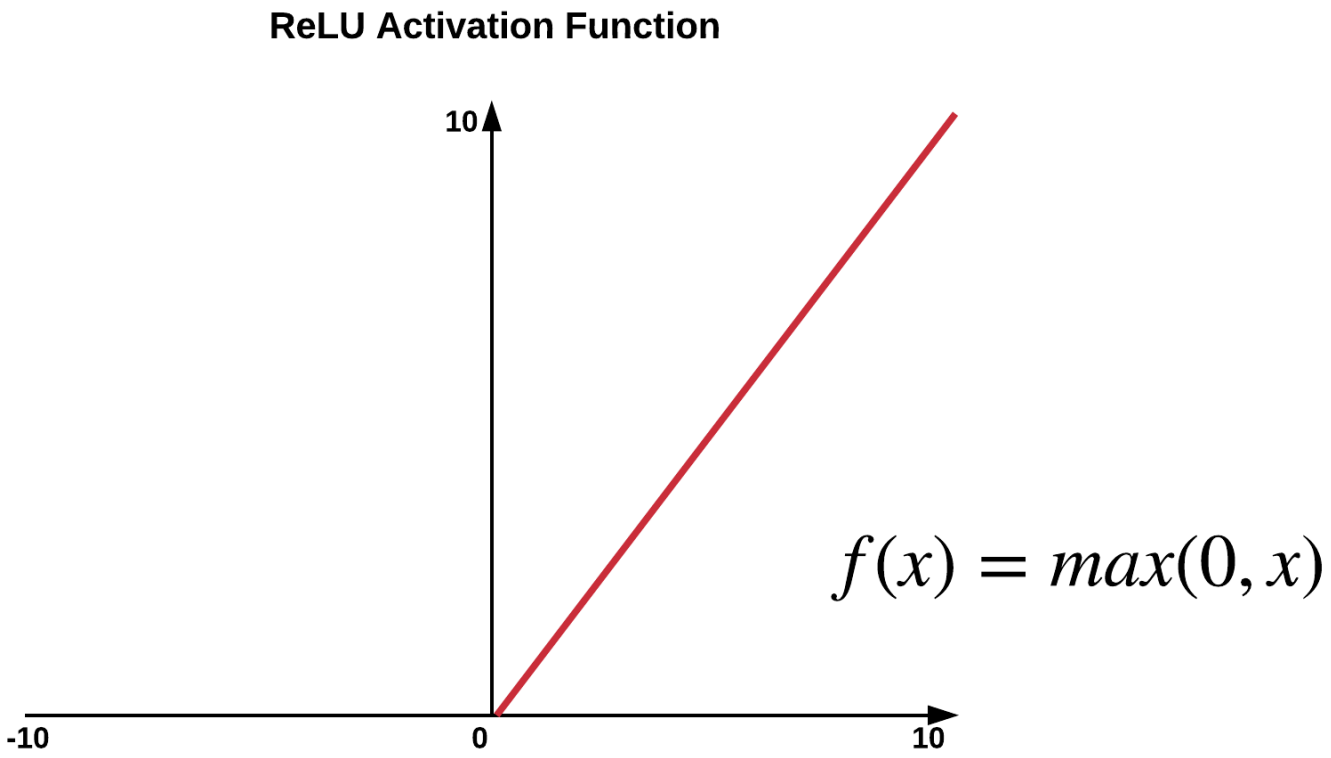


Kernels (Filters) in convolutional neural network (CNN), Let's talk about them. | by Rahul Kadam | CodeX | Medium

- Filters (kernels) perform dot product operations on an image, sliding across in "strides"
- Filters detect features like edges and contours
- Produces a "feature map"

Non-Linear and Pooling Layers

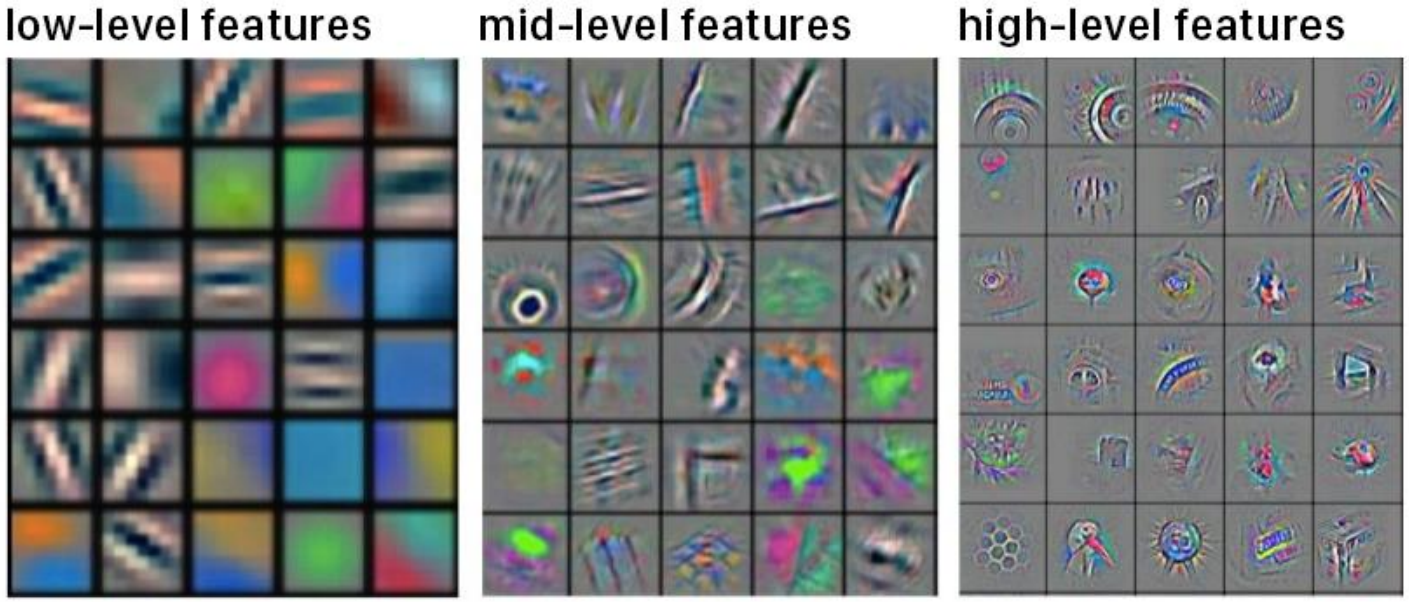
- Feature maps passed through non-linear "activation" functions to maintain complexity
- Output of activation layers fed through a pooling operation for downsampling



Rectified Linear Unit Formula at David Price blog (asktheman.xyz)

Properties of Convolutional Layers

- Higher layers detect more complex, abstract features like objects and arrangements
- Lower layers detect simpler features like colors and textures
- Content captured in high level features, style in low level



Convolutional Neural Networks Explained (twopointseven.github.io)

Neural-Style Transfer Loss

Algorithm takes advantage of convolutional layer properties by considering losses at specific points in network.

It computes the differences in feature maps between a blank input image and the style/content image at certain depths

Neural-Style Transfer Loss

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^l - P_{ij}^l\right)^2 \ .$$

Optimize Pixel Values

- The pixel values are altered in the input image to minimize the loss
- Optimization algorithms like gradient descent do this
- I use the LBFGS Algorithm. LBFGS accounts for "curvature" of loss function to produce dynamic step sizes that minimize loss

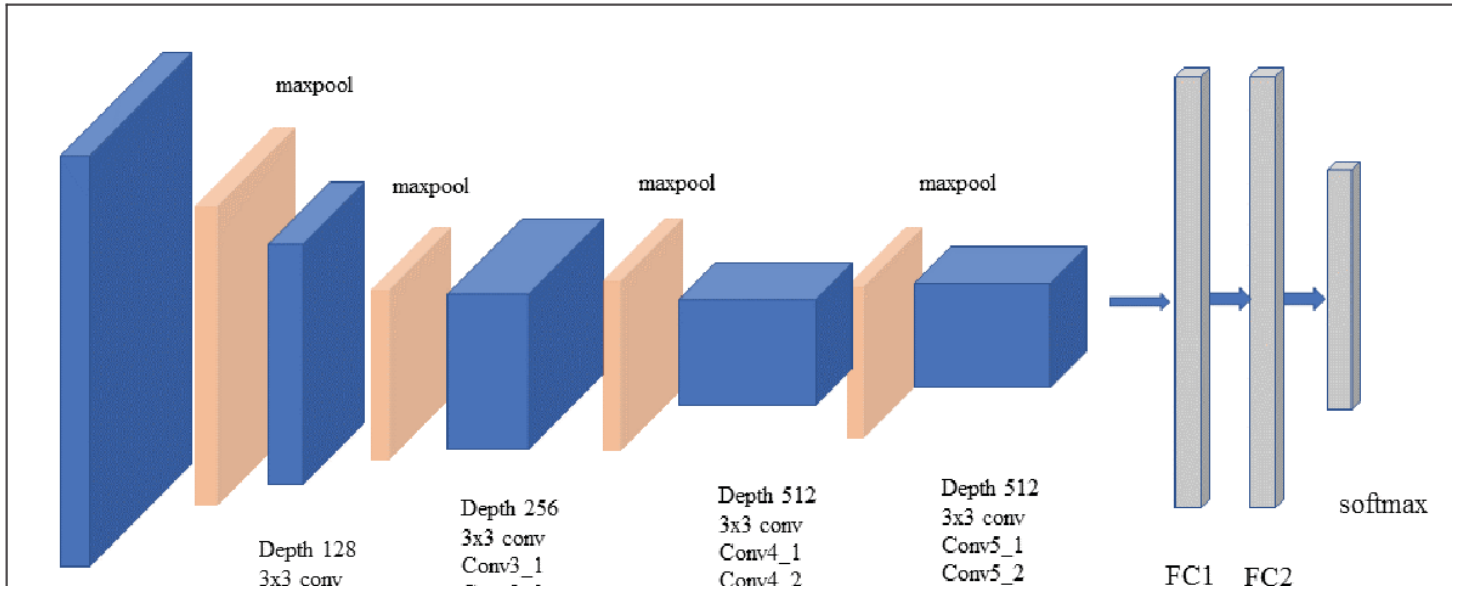
Input Image

- Over repeated iterations of this optimization, the pixel values of the input image begin to resemble the content of content image and style of style image.

Gatys, L., Ecker, A., & Bethge, M. (2016a). A neural algorithm of artistic style. *Journal of Vision*, *16*(12), 326. https://doi.org/10.1167/16.12.326

Implementation

- PyTorch
- Uses pre-trained VGG-19 network on image classification (since kernel values are not being learned)
- Create Loss and Normalization classes that inherit nn.Module, insert as layers into VGG-19 Network



Vgg19 – Towards Data Science

Pick out Specific Layers

```
for layer in convolutional_network.named_children():
    if isinstance(layer[-1], nn.Conv2d):
        name = str(layer)

    elif isinstance(layer[-1], nn.ReLU):
        name = str(layer)

        layer = nn.ReLU(inplace=False)

    elif isinstance(layer[-1], nn.MaxPool2d):
        name = str(layer)
```

```
elif isinstance(layer[-1], nn.BatchNorm2d):
        name = str(layer)
    if type(layer) == tuple:
        if isinstance(layer[-1], nn.Conv2d):
            layer_add = layer[-1]
    else:
        layer_add = layer
    model.add_module(name, layer_add)
```

```
if name in content_layers_default:

content_target_filters = model(image_one).detach()

        content_loss = content_image_loss(content_target_filters)

        model.add_module(f'content_loss_{layer[0]}', content_loss)


    content_loss_list.append(content_loss)
```

Convolutional Layers used in Innovative ways

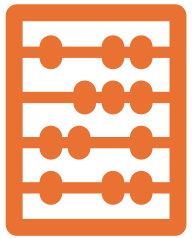
Not a standard optimization task

No learned Kernel values

Creatively crafted loss function

Produces extraordinary results!

Conclusions



Properties of CNNs can be utilized for alternative purposes

Neural Networks can give insight into biological functioning and human intuition about art

AI is pretty cool!