GENERATE ARTISTIC IMAGE USING DEEP LEARNING

BY

MD. MONOARUL ISLAM BHUIYAN

Roll: 1407019

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

KHULNA UNIVERSITY OF ENGINEERING & TECHNOLOGY

KHULNA 9203, BANGLADESH

February, 2019

CERTIFICATION

The project titled "**Generate Artistic Image Using Deep Learning**" submitted by Md. Monoarul Islam Bhuiyan, Roll No: 1407019, Academic Year: 2017-18, for partial fulfillment of the requirements for the degree of "Bachelor of Science in Computer Science and Engineering".

Supervisor

Dr. Muhammad Aminul Haque Akhand

Professor

Department of Computer Science and Engineering

Khulna University of Engineering and Technology

APPROVAL

The project titled "**Generate Artistic Image Using Deep Learning**" submitted by Md. Monoarul Islam Bhuiyan, Roll No: 1407019, Academic Year: 2017-18, for partial fulfillment of the requirements for the degree of "Bachelor of Science in Computer Science and Engineering".

External

Dr. Sk. Md. Masudul Ahsan

Professor

Department of Computer Science and Engineering

Khulna University of Engineering and Technology

ACKNOWLEDGEMENTS

First and foremost, I must sense grateful to and wish to acknowledge my insightful indebtedness to Dr. Muhammad Aminul Haque Akhand, Professor, Department of Computer Science and Engineering and the supervisor of the project. Her unfathomable knowledge in the field of Deep Learning influenced me to carry out this project up to this point. His endless endurance, Scholarly guidance, continual encouragement, constant and lively supervision, constructive criticism, priceless suggestion made it possible to come up to this phase. Without her inspiration, enthusiasm and encouragement, this work could not be completed.

Last, but by no means least, I thank Allah for the abilities I was given that made it possible to undertake this project.

ABSTRACT

In fine art, especially painting, humans have mastered the skill to create unique visual experiences through composing a complex interplay between the content and style of an image. Thus far the algorithmic basis of this process is unknown and there exists no artificial system with similar capabilities. However, in other key areas of visual perception such as object and face recognition near-human performance was recently demonstrated by a class of biologically inspired vision models called Deep Neural Networks. Here I implement the artificial system based on a Deep Neural Network that creates artistic images of high perceptual quality. The system uses neural representations to separate and recombine content and style of arbitrary images on the basis of a neural algorithm for the creation of artistic images.

ABBREVIATIONS

VGG	Visual Geometry Group
CNN	Convolutional Neural Network
MSE	Mean Square Error
RoI	Region of Interest
ReLU	Rectified Linear Unit
FC	Fully Connected
MLP	Multi-layer Perceptron neural network

CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

Neural Transfer Style is one of the most amazing applications of Artificial Intelligence in a creative context. In this Project i transfer an art painting style to a chosen image, creating stunning results. In fine art, especially painting, humans have mastered the skill to create unique visual experiences through composing a complex interplay between the content and style of an image. Thus far the algorithmic basis of this process is unknown and there exists no artificial system with similar capabilities. However, in other key areas of visual perception such as object and face recognition near-human performance was recently demonstrated by a class of biologically inspired vision models called Deep Neural Networks. Here I implement the artificial system based on a Deep Neural Network that creates artistic images of high perceptual quality. The system uses neural representations to separate and recombine content and style of arbitrary images on the basis of a neural algorithm for the creation of artistic images.

1.1 Background

In fine art, especially painting, humans have mastered the skill to create unique visual experiences through composing a complex interplay between the content and style of an image. Thus far the algorithmic basis of this process is unknown and there exists no artificial system with similar capabilities. However, in other key areas of visual perception such as object and face recognition near-human performance was recently demonstrated by a class of biologically inspired vision models called Deep Neural Networks. With the help of an artificial system based on a Deep Neural Network that creates artistic images of high perceptual quality. The system uses neural representations to separate and recombine content and style of arbitrary images, providing a neural algorithm for the creation of artistic images. Moreover, in light of the striking similarities between performance-optimised artificial neural networks and

biological vision it offers a path forward to an algorithmic understanding of how humans create and perceive artistic imagery.

The class of Deep Neural Networks that are most powerful in image processing tasks are called Convolutional Neural Networks. Convolutional Neural Networks consist of layers of small computational units that process visual information hierarchically in a feed-forward manner. Each layer of units can be understood as a collection of image filters, each of which extracts a certain feature from the input image. Thus, the output of a given layer consists of so-called feature maps: differently filtered versions of the input image. When Convolutional Neural Networks are trained on object recognition, they develop a representation of the image that makes object information increasingly explicit along the processing hierarchy. Therefore, along the processing hierarchy of the network, the input image is transformed into representations that increasingly care about the actual content of the image compared to its detailed pixel values. We can directly visualise the information each layer contains about the input image by reconstructing the image only from the feature maps in that layer. Higher layers in the network capture the high-level content in terms of objects and their arrangement in the input image but do not constrain the exact pixel values of the reconstruction. In contrast, reconstructions from the lower layers simply reproduce the exact pixel values of the original image We therefore refer to the feature responses in higher layers of the network as the content representation. To obtain a representation of the style of an input image, we use a feature space originally designed to capture texture information.8 This feature space is built on top of the filter responses in each layer of the network. It consists of the correlations between the different filter responses over the spatial extent of the feature maps By including the feature correlations of multiple layers, we obtain a stationary, multi-scale representation of the input image, which captures its texture information but not the global arrangement.

1.2 Objective of the Project

Neural Style Transfer algorithm uses two images to create an output piece: the content image, and the style image. As is implied by their labels, the style of the style image is woven into the content of the content image. This is carried out using a clever technique which allows the computer to distinguish between the content and style of a picture, using a convolutional neural network (conv net or CNN). The output image (initially a random set of pixels) is compared to the specified content and style, and is incrementally tweaked until it mathematically resembles the desired result.

The study will be carried out with the following specific objectives:

- Review the Neural art method for better artistic images.
- Apply Neural Style algorithm in images.
- Generate artistic images by Deep Learning.
- Study the VGG16 and VGG19 conv net.

1.3 Organization of the Project.

The main attraction of this project is to present aartistic image using Deep Neural Network. The project has five chapters. An introduction to Neural style Transfer has been given in Chapter 1. Chapter wise overviews of rest of the thesis are as follows: Chapter 2 is for literature review that includes a brief description of Deep Learning and Convolutional Neural Network. Chapter 3 explains the VGG Architecture and the VGG 19 network. Chapter 4 explains the working principles of Transfer of Neural Style. Chapter 5 is for the conclusions of this project.

CHAPTER 2

Literature Review

Neural Transfer Style is one of the most amazing applications of Artificial Intelligence in a creative context. In this Project, we'll see how to transfer an art painting style to a chosen image, creating stunning results. Leon A. Gatys et al. [10] conceived the concept of Neural Style Transfer in their paper A Neural Algorithm of Artistic Style in 2015. After that, many researchers applied and improved the methodology, adding elements to the loss, trying different optimizers and experimenting different neural networks used for the purpose.Still, the original paper remains the best source to understand this concept, and the VGG16 and VGG19 networks are the most used models in this context. This choice, which is unusual considering that both have been outperformed by most recent networks, is proved by the highest performance achieved in style transfer.

2.1 Transfer learning

Transfer Learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. For example, knowledge gained while learning to recognize cars could apply when trying to recognize trucks. This area of research bears some relation to the long history of psychological literature on transfer of learning, although formal ties between the two fields are limited.

2.2 Deep Learning

Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised. [1][2][3]

Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases superior to human experts. [4][5][6]

Deep learning models are vaguely inspired by information processing and communication patterns in biological nervous systems yet have various differences from the structural and functional properties of biological brains (especially human brains), which make them incompatible with neuroscience evidences. [7][8][9]

2.3 Convolutional neural network

CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. [2] A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, RELU layer named as activation function, pooling layers, fully connected layers and normalization layers. [3] They are also known as shift variant or space invariant artificial neural networks, based on their shared-weights architecture and translation invariance characteristics. [4][5]

Description of the process as a convolution in neural networks is by convention. Mathematically it is a cross-correlation rather than a convolution (although cross-correlation is a related operation). This only has significance for the indices in the matrix, and thus which weights are placed at which index.

Convolutional networks were inspired by biological processes [6] in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

2.3.1 Convolutional

Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli.

Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of

neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10000 weights for *each* neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters.[9] For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters. In this way, it resolves the vanishing or exploding gradients problem in training traditional multi-layer neural networks with many layers by using backpropagation.

2.3.2 Pooling

Convolutional networks may include local or global pooling layers, which combine the outputs of neuron clusters at one layer into a single neuron in the next layer. For example, *max pooling* uses the maximum value from each of a cluster of neurons at the prior layer. Another example is average pooling, which uses the average value from each of a cluster of neurons at the prior layer.

2.3.3 Fully connected

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP).

2.3.4 Receptive field

In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from *every* element of the previous layer. In a convolutional layer, neurons receive input from only a restricted subarea of the previous layer. Typically the subarea is of a square shape (e.g., size 5 by 5). The input area of a neuron is called its *receptive field*. So, in a fully connected layer, the receptive field is the entire previous layer. In a convolutional layer, the receptive area is smaller than the entire previous layer.

2.3.5 Weights

Each neuron in a neural network computes an output value by applying some function to the input values coming from the receptive field in the previous layer. The function that is applied to the input values is specified by a vector of weights and a bias (typically real numbers). Learning in a neural network progresses by making incremental adjustments to the biases and weights. The vector of weights and the bias are called a *filter* and represents some feature of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons share the same filter. This reduces memory footprint because a single bias and a single vector of weights is used across all receptive fields sharing that filter, rather than each receptive field having its own bias and vector of weights.

2.4 Multilayer Perceptron

A multilayer perceptron (MLP) is a class of feedforward artificial neural network. An MLP consists of, at least, three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

Multilayer perceptrons are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer.

2.5 Feedforward Neural Network

A feedforward neural network is an artificial neural network wherein connections between the nodes do not form a cycle. [7] As such, it is different from recurrent neural networks.

The feedforward neural network was the first and simplest type of artificial neural network devised [8]. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.

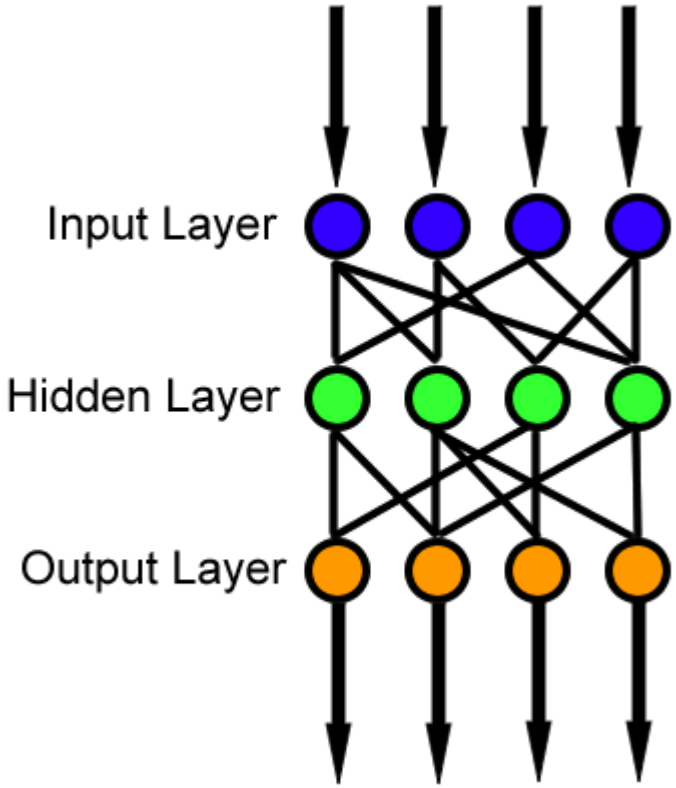

Fig 2.1: Feedforword Network

2.6 Rectifier

where *x* is the input to a neuron. This is also known as a ramp function and is analogous to half-wave rectification in electrical engineering. This activation function was first introduced to a dynamical network by Hahnloser et al. in 2000 with strong biological motivations and mathematical justifications. It has been demonstrated for the first time in 2011 to enable better training of deeper networks,[3] compared to the widely-used activation functions prior to 2011, e.g., the logistic sigmoid (which is inspired by probability theory; see logistic regression) and its more practical[4] counterpart, the hyperbolic tangent. The rectifier is, as of 2018, the most popular activation function for deep neural networks.

A unit employing the rectifier is also called a rectified linear unit (ReLU). [10]

2.7 CNN Building Blocks

A CNN architecture is formed by a stack of distinct layers that transform the input volume into an output volume (e.g. holding the class scores) through a differentiable function. A few distinct types of layers are commonly used. These are further discussed below.

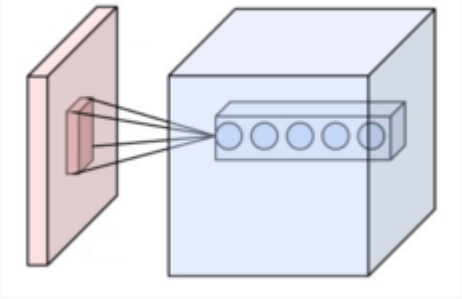

Fig 2.2: Neurons of a convolutional layer (blue), connected to their receptive field (red)

Convolutional layer

The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.

Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map.

Local connectivity

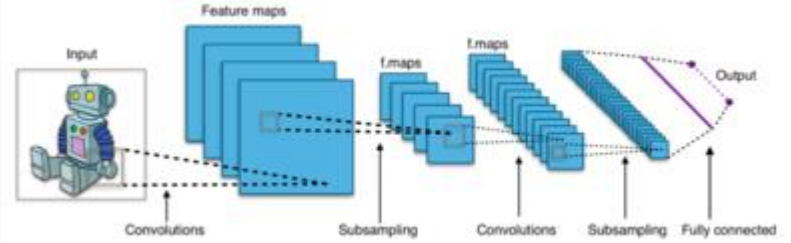

Fig 2.3: Typical CNN architecture

When dealing with high-dimensional inputs such as images, it is impractical to connect neurons to all neurons in the previous volume because such a network architecture does not take the spatial structure of the data into account. Convolutional networks exploit spatially local correlation by enforcing a sparse local connectivity pattern between neurons of adjacent layers: each neuron is connected to only a small region of the input volume.

The extent of this connectivity is a hyper parameter called the receptive field of the neuron. The connections are local in space (along width and height), but always extend along the entire depth of the input volume. Such an architecture ensures that the learnt filters produce the strongest response to a spatially local input pattern.

Pooling layer

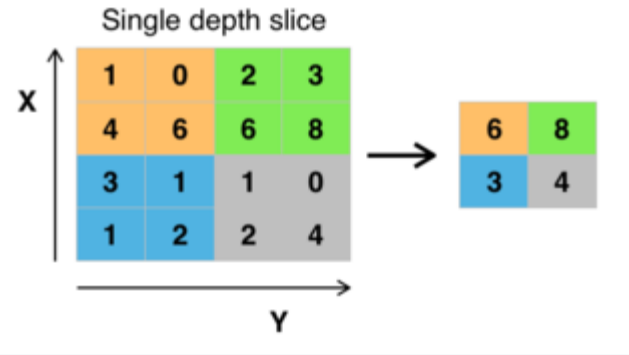

Fig 2.4: Max pooling with a 2x2 filter and stride = 2

Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling among which *max pooling* is the most common. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum.

Intuitively, the exact location of a feature is less important than its rough location relative to other features. This is the idea behind the use of pooling in convolutional neural networks. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters, memory footprint and amount of computation in the network, and hence to also control overfitting. It is common to periodically insert a pooling layer between successive convolutional layers in a CNN architecture. The pooling operation provides another form of translation invariance.

The pooling layer operates independently on every depth slice of the input and resizes it spatially. The most common form is a pooling layer with filters of size 2×2 applied with a stride of 2 downsamples at every depth slice in the input by 2 along both width and height, discarding 75% of the activations. In this case, every max operation is over 4 numbers. The depth dimension remains unchanged.

ReLU layer

ReLU is the abbreviation of rectified linear unit, which applies the non-saturating activation function. Effectively, it removes negative values from an activation map by setting them to zero. It increases the nonlinear properties of

the decision function and of the overall network without affecting the receptive fields of the convolution layer.

Other functions are also used to increase nonlinearity, for example the saturating hyperbolic tangent and the sigmoid function. ReLU is often preferred to other functions because it trains the neural network several times faster without a significant penalty to generalization accuracy.

Fully connected layer

Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular (non-convolutional) artificial neural networks. Their activations can thus be computed as an affine transformation, with matrix multiplication followed by a bias offset (vector addition of a learned or fixed bias term).

Loss layer

The loss layer specifies how training penalizes the deviation between the predicted (output) and true labels and is normally the final layer of a neural network. Various loss functions appropriate for different tasks may be used. Softmax loss is used for predicting a single class of K mutually exclusive classes Sigmoid cross-entropy loss is used for predicting *K* independent probability values in [0,1]. Euclidean loss is used for regressing to real-valued labels .

CHAPTER 3

VGG architecture And Network

3.1 VGG Network

A Convolutional Neural Network (CNN, or ConvNet) are a special kind of multi-layer neural networks, designed to recognize visual patterns directly from pixel images with minimal preprocessing.The ImageNet project is a large visual database designed for use in visual object recognition software research. The ImageNet project runs an annual software contest, the ImageNet Large Scale Visual Recognition Challenge where software programs compete to correctly classify and detect objects and scenes.

VGG[11] network is characterized by its simplicity, using only *3×3* convolutional layers stacked on top of each other in increasing depth. Reducing volume size is handled by max pooling. Two fully-connected layers, each with 4,096 nodes are then followed by a softmax classifier.

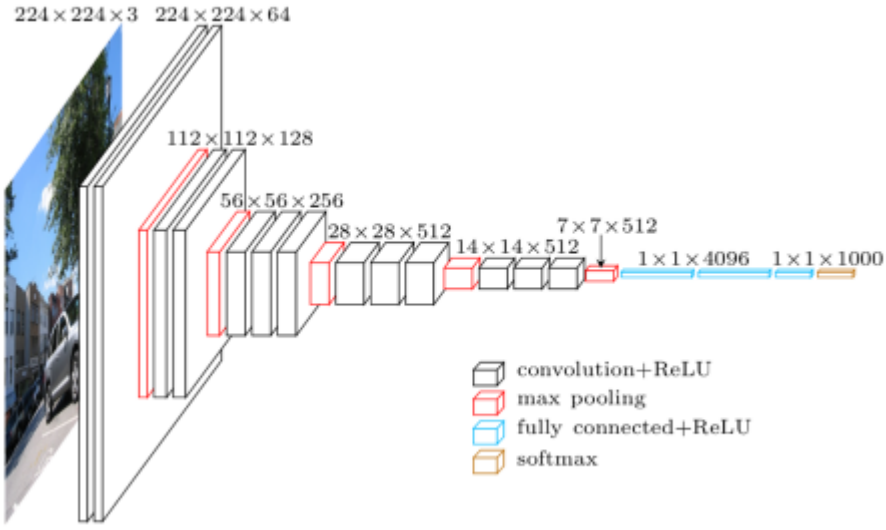

Fig 3.1 : Typical VGG Network

3.2Visualization of VGG 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					

3.2.1 Configuration of VGG

VGG Net consists of 16 convolutional layers and is very appealing because of its very uniform architecture. It only performs 3×33\times 33×3 convolutions and 2×22\times 22×2 pooling all the way through. It is currently the most preferred choice in the community for extracting features from images. However, VGG Net consists of 140 million parameters, which can be a bit challenging to handle.

3.3 VGG 19 Network

The neural style research paper works with the use of a pre-trained, very competent CNN called VGG 16. VGG 16 includes a classification section, but only its convolutional components are used for N.A.A.S. The network's architecture is as follows where after every convolution listed, the ReLU activation function is applied. All filters in the network are of size 3x3. Pooling in the network is of size 2x2.

Convolutional block 1:

Conv1_1: 64 filters, each with depth 3.

Conv1_2: 64 filters, each with depth 64.

Max Pooling 1

Convolutional block 2:

Conv2_1: 128 filters, each with depth 64.

Conv2_2: 128 filters, each with depth 128.

Max Pooling 2

Convolutional block 3:

Conv3_1: 256 filters, each with depth 128.

Conv3_2: 256 filters, each with depth 256.

Conv3_3: 256 filters, each with depth 256.

Conv3_4: 256 filters, each with depth 256.

Max Pooling 3

Convolutional block 4:

Conv4_1: 512 filters, each with depth 256.

Conv4_2: 512 filters, each with depth 512.

Conv4_3: 512 filters, each with depth 512.

Conv4_4: 512 filters, each with depth 512.

Max Pooling 4

Convolutional block 5:

Conv5_1: 512 filters, each with depth 512.

Conv5_2: 512 filters, each with depth 512.

Conv5_3: 512 filters, each with depth 512.

Conv5_4: 512 filters, each with depth 512.

Max Pooling 5

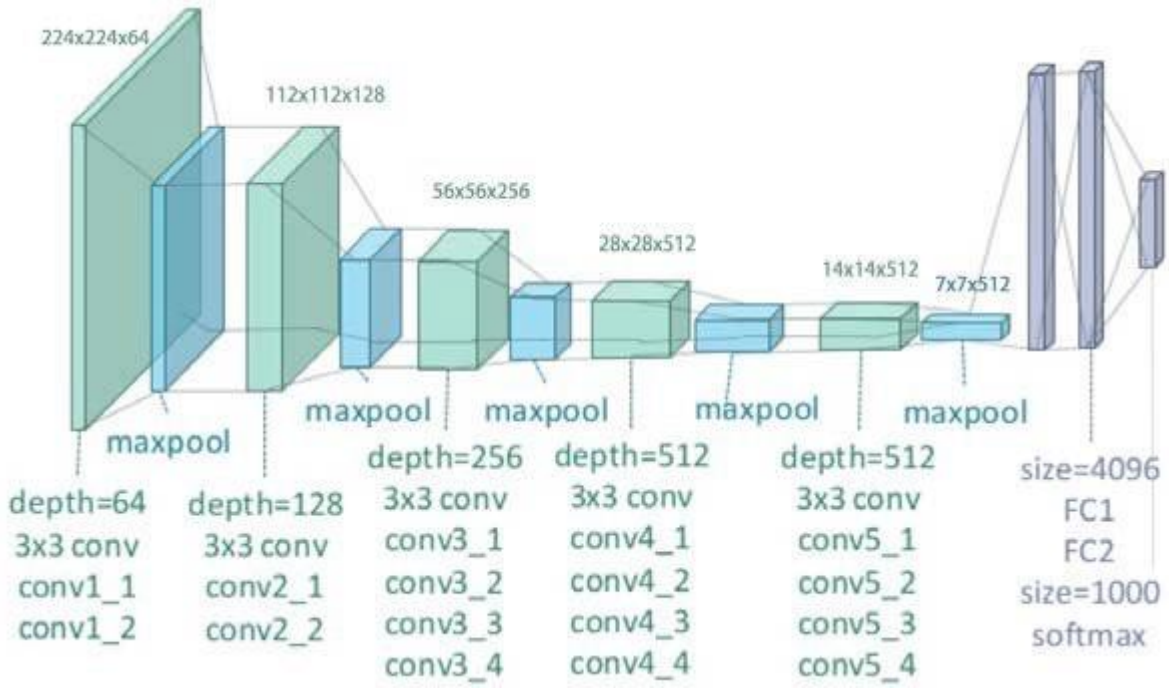

Fig:3.2 VGG19 Network, including FC layers

CHAPTER 4

Neural Style Transfer Working Principles

4.1 Basic Idea

The goal of this technique is to apply the style of an image, which we will call "style image", to a target image, conserving the content of the latter. Defining these two terms as below:

- **Style** is textures and visual patterns in an image. An example of this is the brush strokes of an artist.
- **Content** is the macrostructure of an image. People, buildings, objects are examples of the content of an image.

4.2 High-level steps to generate artistic images :

- Choose the image to style
- Choose the style reference image. Usually, this is a painting with a peculiar and well recognizable style.
- Initialize a pre-trained Deep neural network mainly the VGG 16, and obtain the feature representations of intermediate layers. This step is done to achieve the representations of both the content image and the style image. In the content image, the best option is to obtain the feature representations of the highest layers, since they contain information on the image macrostructure. For the style reference image, feature representations are obtained from multiple layers at different scales.
- Define the loss function to minimize as the sum of the content loss, the style loss and the variation loss. Each iteration, the optimizer generated an image. The content loss is the difference (l2 normalization) between the generated image and the content image, while the style loss between the generated image and the style. We'll see later how these variables are defined mathematically.
- Re-iterate the minimization of the loss

4.3 Processing and Deprocessing Images

First of all, formatting our images to be used by our network. The CNN that we are going to use is the pre-trained VGG19 convnet. As processing the image into a compatible array, we also need to de-process the resulting image, switching from a BGR to an RGB format.

4.4 Content Representation and Loss

The content loss preserves the content of the main input image to style. Since higher layers of a Convolutional Neural Network contain information of the image macrostructure, we calculate the content loss as the difference (l2 normalization) between the output of the highest layer of the input image, and the same layer of the generated image.

The content loss is defined as:

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

Content loss

In the equation, *F* is the feature representation of the content image (what the network outputs when we run our input image through), and *P* the one of the generated image, at a specific hidden layer *l*.

The content loss generated at layer l of a conv net, with regards to a content image p and an optimization image x, is the average square difference between each content representation, or activation, of p and x at layer l.

Here, F denotes the representation of the optimization image at layer l of a conv net. F(i, j) is a single neuron from F at layer l. P and P(i, j) denote the content image representation at layer l, and an activation thereof. So, for every value pair of i and j, add the squared difference between activation F(i, j) and P(i, j) to a running sum. Upon completion of this process, divide the sum by the total number of iterations in the summation (equal to the number of neurons in F or P). Divide again by 2.

The objective of the entire style transfer process is to minimize the N.A.A.S. cost function, the content loss making up one half of the function. In order to minimize the loss, the optimization image must be changed such that its content representation F is

similar to the content image's representation P. This would transfer the content of p onto x, preserving all the important details of the specified image.

4.5 Style Representation and Loss

The most inventive half of Neural style transfer alogorithm is the style representation. To differentiate style from content is to use gram matrices to compute the correlation between features.

To compute a gram matrix, we take the content representation at a layer ℓ and convert the data into a two-dimensional matrix. Each row of the matrix is the entire output of a filter which analyzed layer ℓ –1. Seeing as every filter is looking to pick up on a specific feature, one could also say that every row in the matrix is a representation of a given feature at different positions on the input image.

Now that the values have been properly structured, the matrix can be computed using the following equation:

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l}$$

Gram matrix computation

This equation explicitly states that element (i, j) of the gram matrix is equal to the sum of the product of elements from two distinct rows in the content matrix.

To break it down, element 1 of row i is multiplied by element 1 of row j. Element 2 times element 2 and so on, up until element k. Finally, these products are summed to produce a single element of the gram matrix.

So, to compute the gram matrix in its entirety, this is computation is carried out for every combination of rows in the matrix.

As every row of the matrix was an unrolled feature map, containing spatial information regarding the presence of features. By multiplying elements together to check if these elements overlap at their location in the image. This is done for every location in the image and with every possible pairing of features. Summation is then performed,

discarding all of the information's spatial relevance and this overlap or correlation between elements, regardless of their whereabouts, is exactly what style is.

And that's the representation of style.

Now, the MSE Loss is put to use once again.

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

MSE Loss for style

This is iterating over the gram matrix of the optimization image (G) and the gram matrix of the style image (A). summing the square difference between all elements of these two matrices. Here, N represents the number of feature maps in layer l and M represents the number of elements in these N feature maps. The sum is then scaled by 4 times $N^2 * M^2$ for a final style loss for layer l.

The total style loss is therefore:

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

Complete style loss

Where w(l) represents a weight for the loss at layer l, and E(l) is the loss at layer l.

4.6 Total loss

Finally, the total loss is calculated taking into account all these contributions. First, we need to extract the output of the specific layers we choose. Then, we compute the loss calling the functions previously code. Each component is multiplied by specific weights, that we can tune to give intense or lighter effects.

The loss function we minimise is

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

Seeing as the object of the algorithm is to minimize the cost function, the optimization image's style will slowly change to resemble that of the style image. This will increase the similarity of their gram matrices, ultimately decreasing the cost.

4.7 Project Outcome

The Generation of images after each iteration with the help of content and style images are shown as below and the current loss value are decreasing after each iteration.

Fig 4.1 Content Image(Sample)

Fig 4.2 Style Image(Sample)

Fig 4.3 Generated Image of 20 iterations

CHAPTER 5

Conclusion

Neural Transfer Style is a machine learning technique for combining the artistic style of one image with the content of another image. The basic idea is to take the feature representations learned by a pre-trained deep convolutional neural network (typically trained for image classification or object detection) to obtain separate representations for the style and content of any image. Once these representations are found, we can then try to optimize a generated image to recombine the content and style of different target images.

REFERENCES

[1] West, Jeremy; Ventura, Dan; Warnick, Sean (2007). "Spring Research Presentation: A Theoretical Foundation for Inductive Transfer". Brigham Young University, College of Physical and Mathematical Sciences. Archived from the original on 2007-08-01. Retrieved 2007-08-05.E.

[2] LeCun, Yann. "LeNet-5, convolutional neural networks". Retrieved 16 November 2013.

[3] "CS231n Convolutional Neural Networks for Visual Recognition". cs231n.github.io. Retrieved 2018-12-13

[4] Zhang, Wei (1988). "Shift-invariant pattern recognition neural network and its optical architecture". Proceedings of Annual Conference of the Japan Society of Applied Physics.

[5] Zhang, Wei (1990). "Parallel distributed processing model with local space-invariant interconnections and its optical architecture". Applied Optics. **29** (32): 4790–7. Bibcode:1990 ApOpt..29.4790Z.

[6] Matusugu, Masakazu; Katsuhiko Mori; Yusuke Mitari; Yuji Kaneda (2003). "Subject independent facial expression recognition with robust face detection using a convolutional neural network" (PDF). *Neural Networks*. **16** (5): 555–559. Retrieved 17 November 2013.

[7] Zell, Andreas (1994). Simulation Neuronaler Netze [Simulation of Neural Networks] (in German) (1st ed.). Addison-Wesley. p. 73. ISBN 3-89319-554-8.

[8] "Deep learning in neural networks: An overview". *Neural Networks. **61**: 85–117. 2015-01-01. .*

[9] Vinod Nair and Geoffrey Hinton (2010). Rectified Linear Units Improve Restricted Boltzmann Machines

[10] A Neural Algorithm of Artistic Style Leon A. Gatys, Alexander S. Ecker, Matthias Bethge (Submitted on 26 Aug 2015 (v1), last revised 2 Sep 2015 (this version, v2))

[11] VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION Karen Simonyan & Andrew Zisserman