IMPLEMENTATION OF DEEP DREAM & NEURAL STYLE TRANSFER ALGORITHM USING PYTHON

Project report submitted in partial fulfillment of the of the requirement for the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

By
PRAKHAR (191031)

CHANDRAMAULISHWAR SINGH (191037)

UNDER THE SUPERVISION OF

DR. ALOK KUMAR

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
WAKNAGHAT
MAY 2023

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1 -

CHAPTER 2 -

CHAPTER 3 -

CHAPTER 4 -

LIST OF TABLES

CHAPTER 1 -

DECLARATION

We hereby declare that the work reported in the B.Tech Project Report entitled **"IMPLEMENTATION OF DEEP DREAM & NEURAL STYLE TRANSFER ALGORITHM USING PYTHON"** submitted at Jaypee University of Information Technology, Waknaghat, India is an authentic record of our work carried out under the supervision of Dr. Alok Kumar. We have not submitted this work elsewhere for any other degree or diploma.

PRAKHAR

191031

CHANDRAMAULISHWAR SINGH

191037

This is to certify that the above statement made by the candidates is correct to the best of my knowledge

Dr. Alok Kumar

Project Supervisor

Date:

Dr. Shweta Pandit

Project Co-ordinator

ACKNOWLEDGEMENT

It is my team's honor to express my feelings of gratitude and thankfulness towards our project Supervisor **Dr. Alok Kumar**, who with his sincere guidance and knowledge helped us in completing this project report on the topic "**IMPLEMENTATION OF DEEP DREAM & NEURAL STYLE TRANSFER ALGORITHM USING PYTHON**".

Without his motivation, this work would not have been possible. I, along with my partner, am forever indebted for his kind guidance and encouragement.

I would also like to mention the sincere contribution of my partner, who with their hard work and dedication made this project report possible. This study has indeed helped us explore and develop more knowledge avenues related to our project work and I am confident it will help us in the future.

THANK YOU

PLAGIARISM

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ……

Type of Document (Tick): | **PhD Thesis** | **M.Tech Dissertation/ Report** | **B.Tech Project Report** | **Paper** |

Name: _____ __Department: _____ Enrolment No _____

Contact No. _____E-mail. _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ……..(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><th>Copy Received on</th><th>Excluded</th><th>Similarity Index (%)</th><th colspan="2">Generated Plagiarism Report Details (Title, Abstract & Chapters)</th></tr>
<tr><td rowspan="2"></td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td>Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td rowspan="2">Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

……

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ABSTRACT

Deep Dream is an artistic algorithm where a pretrained CNN feds an image and optimizes it to amplify the features it "sees" in the image. Depending on the neural network layer, the features amplified will either be low-level (like edges, certain geometric patterns, etc.) or high-level (like dog snouts, eyes, etc.), which heavily depends on the dataset on which the net was pretrained! As a result, mimicking phenomenological features of altered states without these other more widespread consequences offers an essential experimental tool for research into consciousness and psychiatry. Here, we discuss this device, which we refer to as the hallucination machine. Deep convolutional neural networks (DCNNs) and panoramic footage of natural surroundings, watched immersively through a head-mounted display, make up the innovative combination (panoramic VR). Neural style transfer is a technique in computer vision that enables the creation of artistic images by combining the content of one image with the style of another. The technique is based on deep neural networks, specifically convolutional neural networks (CNNs), which are trained to learn the artistic style of a particular image. By optimizing a cost function that balances the content and style of two images, neural style transfer generates a new image that incorporates the content of one image while applying the style of another. This technique has many practical applications, such as in the creation of personalized artwork, the design of websites, and the development of visual effects in movies and video games. Despite its computational complexity, neural style transfer has proven to be a powerful and versatile tool for creating visually stunning and unique images. Implementation of Deep Dream algorithm has been done using VGG-16 and ResNet 50, implementation of Neural Style Transfer (NST) has been done using VGG-19.

CHAPTER 1

INTRODUCTION

1.1 A DESCRIPTION OF THE TECHNOLOGY

In order to automatically categorize photographs, the program is made to recognize faces and other patterns in images.

This serves as the framework for the Deep Dream concept and may be utilized for graphics to help you better understand the neural network's emergent structure.

In contrast, a current picture may be changed to make it "more cat-like," and the improved image can then be added once again to the process [2].

This application is similar to the practice of searching for skies that may have mammal or some other designs.

Fig 1.1 – Different patterns in the skies

Google developed the tool Deep Dream. Utilizes sophisticated AI techniques and a CNN to identify and improve patterns in photos creating a purposeful hallucinatory effect in the photographs. [4]

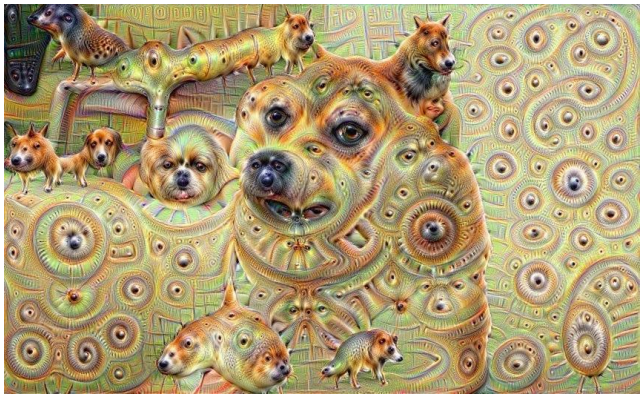


Fig 1.2 – Finding dog like patterns in the above image

When gradient ascent is applied individually to each input pixel, it creates pictures with an excess of high-frequency information due to the lack of relationship between neighboring pixels. By adding a prior or regularizer that favors inputs with natural picture statistics (without a preference for any particular image), or that are just smooth, the resulting images can be enhanced. The discussion of various regularizers continues here [3][4]. More recently, a thorough visual examination of feature visualization and regularization methods was released.

1.1.1 Objective

Basically, deep dream is used to analyze the patterns in different images, just like our brain tries to find patterns in different things or different images.

The concept of dreaming may be used to explore patterns that are not visible in the outcome, allowing examination of the representation and activities of various network elements. Additionally, the input can be optimized to satisfy either a single neuron (this application is occasionally known as "activity maximization") or an entire layer of neurons. [2]

It was only lately suggested that attaching "dreamed" inputs to the training set might shorten training periods for abbreviations in computer science, even though dreaming is almost always employed for seeing networks or creating digital art. [2][3]

It’s been shown that the DeepDream model may be used to create computer arts.

Fig 1.3 - Computer art using deep dream

1.1.2 Future Scope

There is not only one use of deep dream; it can also be used to generate computer art using different images in a dataset by applying it to the original image. It can also be used to create or show people what hallucinations look like. [9]

The concept of dreaming may be used to examine hidden (inner) neurons more than in the output, permitting research of the functioning and representations of numerous network elements.

This serves as the substructure for the DeepDream research and may be utilized for judgments to grasp the arising structure of the neural network. [2][3]

1.2 LITERATURE REVIEW

The DeepDream program was created in 2014 and it was made public in July 2015. It was based on a deep convolutional neural network dubbed "Inception" after the movie of the same name.

Google's DeepDream initiative helped the concept of dreaming and its name become well-known online in 2015. Similar techniques have been used to create synthetic visual textures.

The concept has been around since the early days of neural networks. Several research groups have already pioneered related visualization concepts before Google.[7]

A plethora of equipment in the shape of online services, mobile programs, and desktop programs has been made available on the market to allow individuals to alter their own images since Google revealed their methods and made their code open to the public.[9]

The Hallucination Machine was created by a research team from the University of Sussex in 2017.

It turns a panoramic video into a virtual reality environment that users may explore to mimic the effects of intoxicants and/or psychopathological conditions using the DeepDream algorithm.

They were successful in proving that the subjective impressions caused by the Hallucination Machine were noticeably different from control (non-hallucinogenic) films, despite possessing phenomenology parallels to the psychedelic state (following the administration of psilocybin). [4][5]

The resemblance between DeepDream and real psychedelic experiences was shown in a study that was published in the year 2021 using neuroscientific data.

While viewing a movie clip and its DeepDream-generated equivalent, the writers sat down to record the brain activity of human volunteers.

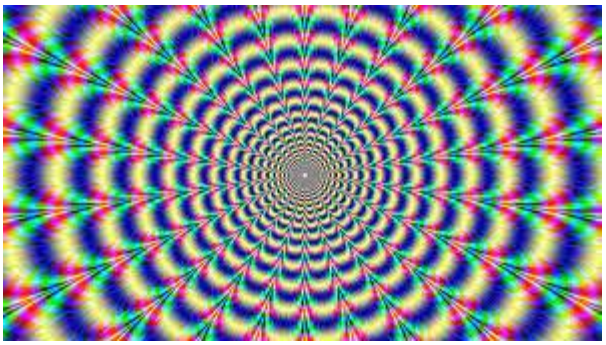


Fig 1.4 - Hallucination image

Two well-known hallmarks of a true psychedelic experience—a strong signal and more functional interaction between different brain areas.

Researchers from the University of Trento will report their findings in the end of 2022, stating that "participants" cognitive flexibility and inventiveness were tested post being exposed to simulated 3D environment panoramic movies and their hallucinatory counterparts created by the DeepDream algorithm.[6]

1.3 PROBLEM DEFINITION

The idea behind this algorithm is that given an input image, the neural network we run on some kind of classification test or something similar on IMAGENET basically sees something in that image, certain features, and whatever the network sees we just tell it to amplify those exact features and by doing that you get the image. The categorization of images and speech recognition have made notable recent advancements thanks to artificial neural networks. Nevertheless, despite the fact that these are really helpful tools based on existing mathematical techniques, we in fact know Unexpectedly little about why certain models are successful and others are not.

Let's examine some basic methods for snooping around these networks. In order to get the classifications we desire from an artificial neural network, we progressively change the network parameters after exposing it to millions of training samples. The artificial neurons are commonly layered in ten to thirty layers in the network. Each visual is supplied to the input layer, which contacts with the following layer, and so on right up until the "output" layer. This top output layer provides the network's "response." Understanding exactly what happens at each layer is one of the difficulties with neural networks [5]. We are aware that during training, each layer gradually extracts more complex information from the picture so that the last layer effectively decides what the image shows. [1]

Intermediate layers evaluate the fundamental characteristics to search for elements, such as a door or a leaf. These neurons light up in reaction to extremely complex objects, like entire houses. Next few layers combine those into comprehensive interpretations. Using the network and asking it to improve an input image in a way that encourages a specific interpretation can help you see what's going on. There's a surprise: neural networks that were taught to pick out various image kinds also possess a significant amount of the knowledge required to create images. [1][2] Check out some further samples from various classes: We may also let the network decide which characteristic to amplify rather than giving it specific instructions. In this case, we just give the network a random picture or snapshot and let it process it. Next, we choose a layer and instruct the network to improve anything it discovers. The complexity of the features we create depends on the layer of the network we choose to improve, as each layer deals with features at a different degree of abstraction.

Because they are sensitive to fundamental properties like edges and their orientations, lower layers often create strokes or simple ornamental patterns.

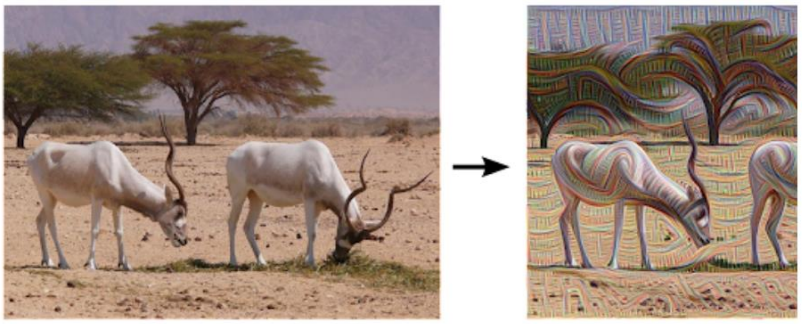


Fig 1.5 - Original photo and processed

Complex characteristics or even entire things often appear when we select higher-level layers, which can recognize more complicated aspects in photos. Again, we just provide our neural network with an existing image as a starting point. We say to the network, "I want more of whatever you see there!" This results in a feedback loop: if a cloud resembles a bird just a little, the network will intensify that similarity [3]. The network will then recognize the bird much more clearly on subsequent passes as a result, and so on, until a highly detailed bird suddenly appears.

1.4 REQUIREMENTS

1.4.1 Software Requirements

Sr. No	Software Required
1.	Jupyter Notebook
2.	VS Code

1.4.2 Libraries Requirements

Sr. No	Library Name
1.	PyTorch
2.	Open CV
3.	Matplotlib

CHAPTER 2

OVERVIEW OF THE PROJECT

2.1 UNDERSTANDING THE PROGRAMMING LANGUAGE

2.1.1 Python

Python is a general-purpose, high-level programming language. Its design philosophy prioritises code readability and often employs indentation. [10][11]

Python has dynamic typing and garbage collection. It supports a variety of various programming paradigms, including structured, object-oriented, procedure, and functional programming (especially this). Because of its enormous standard library, it is frequently alluded to as a "batteries included" language. [11]

In the late 1980s, Guido van Rossum created Python to rival the ABC programming language. In 1991, Python 0.9.0 was made accessible.

In Python 2.0, that was made accessible in 2000, new features were added, including list comprehensions, cycle-detecting garbage collection, reference counting, and support for Unicode.[10]

Python 3.0, which was released in 2008, was a substantial change that was not totally backward compatible with earlier versions. Python 2 was discontinued in version 2.7.18. [10][11]

Fig 2.1 - Python

Python is intended to be an easy-to-read language. It has a neat structure and commonly uses English phrases in place of punctuation. In contrast to different languages, it do not utilize curly brackets to divide bricks, and although semicolons are allowed following remarks, they are once in a while used. It has fewer special cases and syntactic exceptions instances than C [11]

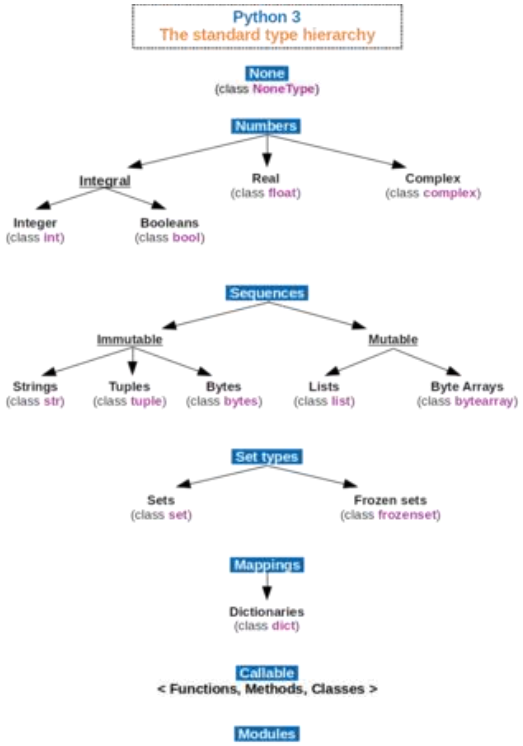


Fig 2.2 - Standard type hierarchy in python

2.2 UNDERSTANDING THE SOFTWARE REQUIRED

2.2.1 JUPYTER NOTEBOOK

Using the free IDE Jupyter Notebook, Jupyter documents may be created and shared while being actively coded. Additionally, it is a web-based interactive computing environment. Python, Julia, Scala, R, and other popular data science languages may all be used in the Jupyter notebook. [12]

A Jupyter Notebook was the first web tool for creating and sharing computational documents. It offers a simple, effective, document-focused user interface. [12][13]

Fig 2.3 - File upload interface of Jupyter Notebook

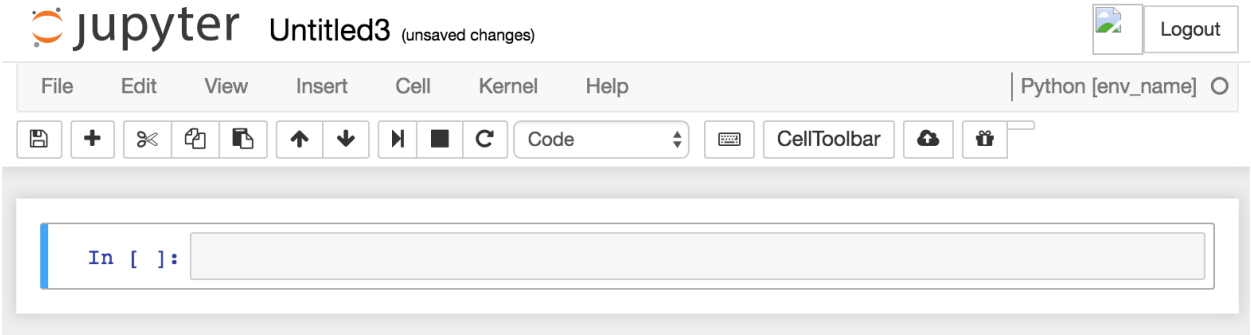


Fig 2.4 - Coding Interface of Jupyter Notebook

A Jupyter Notebook document is a JSON file that adheres to a versioned format and often ends in ".ipynb." The metadata, notebook format, and cell list are the three main components of Jupyter notebooks. To set up and present the notebook, metadata works as a data dictionary of definitions. The software's version number is Notebook Format. [14]

2.2.2 VS Code

With the help of the free code editor Visual Studio Code, you can start coding right now. We can use it to code in any computer language without switching editors. Visual Studio Code provides support for a wide range of languages, including Python, Java, C++, and JavaScript. [15]

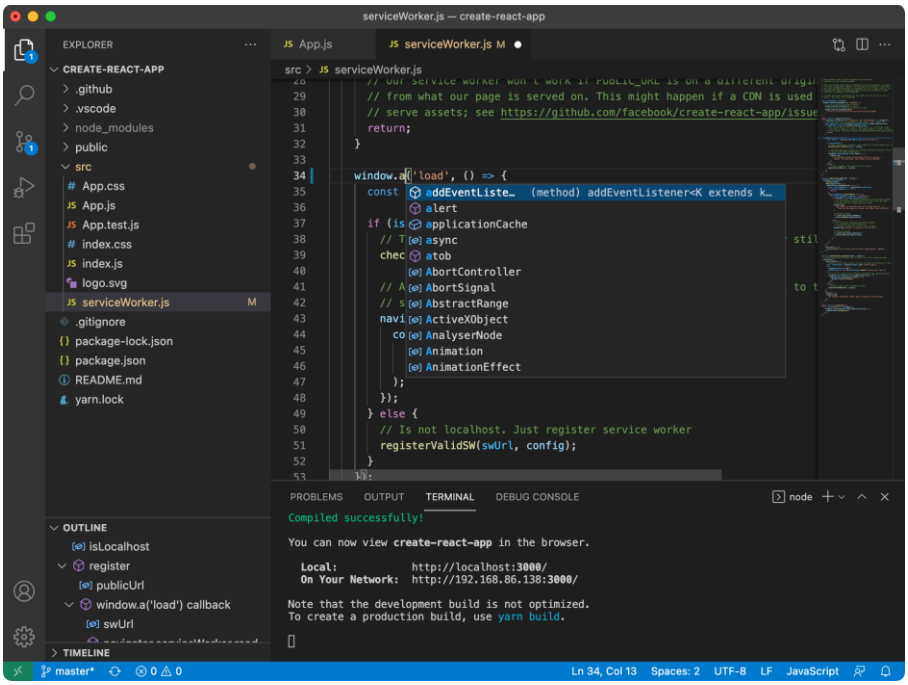


Fig 2.5 - Coding environment of VS Code

2.3 UNDERSTANDING THE LIBRARIES

2.3.1 PyTorch

PyTorch is a machine learning framework based on the Torch library that is used for applications like computer vision and natural language processing. PyTorch was originally developed by Meta AI and is now a member of the Linux Foundation [16]. It is free software that is open-source and distributed with a modified BSD license. Even though the Python interface is more sophisticated and the major focus of development, PyTorch has a C++ interface.

Fig 2.6 - PyTorch

The framework combines Torch's GPU-accelerated backend modules, which are effective and adaptable, with Python's logical frontend, which emphasizes quick prototyping, understandable code, and support for the greatest range of deep learning models. Pytorch enables programmers to use the well-known imperative programming paradigm while still producing graphs. [16][17] It was made available as open source in 2017, and because of its Python heritage, machine learning developers love it. [16]

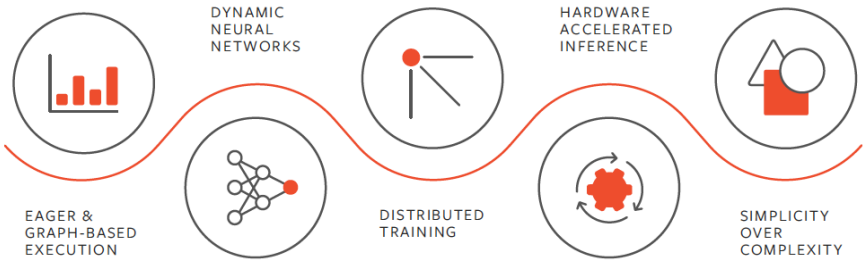


Fig 2.7 - Features of PyTorch

2.3.2 OPEN CV

A collection of functions with a focus on real-time computer vision is known as OpenCV (Open Source Computer Vision Library). It was first created by Intel and then sponsored by Willow Garage and Itseez. [18]

The library is freely available and cross-platform thanks to the Apache 2 License for Open-Source Software. As of 2011, It now provides GPU acceleration for real-time operations.

Fig 2.8 - OpenCV

OpenCV still has a more constrained but still reliable older C interface, while being created in C++ and using C++ as its primary interface. All of the most recent advancements and algorithms are present in the C++ interface. There are Java, Python, and MATLAB/OCTAVE bindings. The API for these interfaces is available in the online documentation. [18]

Wrappers have been developed for a variety of computer languages to encourage adoption by a wider audience. In version 3.4, JavaScript bindings for a certain subset of OpenCV functions were made accessible as OpenCV.js for use with web-based systems.[18]

Applications of OPEN CV:

- toolkits for 2D and 3D features
- estimate of egotism
- Face recognition technology
- gesture identification
- Computer-human interaction (HCI)
- Robotics on wheels
- knowledge of motion
- Detecting objects
- Identification and segmentation
- stereopsis stereo vision: two cameras' combined depth perception
- Building from motion (SFM)
- motion monitoring
- combined reality

2.3.3 MATPLOTLIB

For the Python and NumPy mathematics extension, Matplotlib is a graphing toolkit. It gives programs utilizing all-purpose GUI toolkits.

It is not advised to use the procedural "pylab" confluence, which was designed to resemble in detail the MATLAB confluence and is based on a state machine (just like OpenGL). In SciPy, Matplotlib is utilized. [19][20]

The basic idea of matplotlib is that it is used to plot graphs i.e... bar graphs, pie charts, histograms etc.

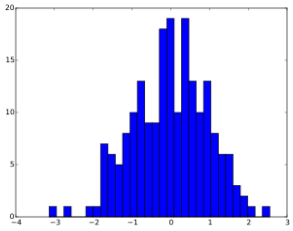


Fig 2.9 - Histogram

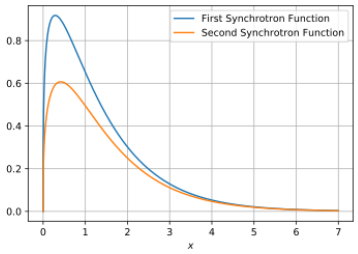


Fig 2.10 - Line Graph

2.3.4 Tensor Flow Hub

For the purpose of finding, using, and sharing pre-trained machine learning models in TensorFlow, there is a platform and library called TensorFlow Hub. It offers an easy-to-use API to integrate these models into new applications and streamlines the process of looking for pre-trained models for a range of tasks[22]. By enabling fine-tuning on tiny data sets, the platform also permits the dissemination of customised models, which addresses the issue of data scarcity[22]. By giving academics and developers access to a centralised collection of models that can be shared and reused to progress the field, TensorFlow Hub is a potent tool for expediting the development of machine learning applications.

Fig 2.11 - Tensor Flow Hub

CHAPTER 3

WORKING

3.1 METHODOLOGY

This is accomplished by sending a picture through the network, followed by the calculation of the gradient of the portrayal with regard to the arising of a specific layer. The portrayal is then altered to boost these arisings, strengthening the marking noticed by the mesh and creating a visual that resembles a dream. This method was known as "Inceptionism".

3.2 ALGORITHM USED

We have used gradient ascent on Mean Squared Error (MSE) loss. With one exception, gradient ascending behaves similarly to gradient fall. Its aim is to maximize some function rather than minimize it.

The distinction arises from the fact that, periodically, we may desire to maximize a function instead of minimizing it; for illustration, if we're maximizing the distance between separation hyperplanes and observations.

Gradient ascent, on the other hand, denotes a movement toward the nearest maximum while gradient decline denotes an iterative movement toward the closest minimum. In this sense, there exists a symmetric function -f on any function f to which gradient descent is applied to which gradient ascent could be employed. This indicates that if we merely project a problem over onto axis of the independent variable, we can analyze it by using gradient ascent and also gradient descent. The same function is illustrated in this graph, except that it is mirrored down the x-axis:

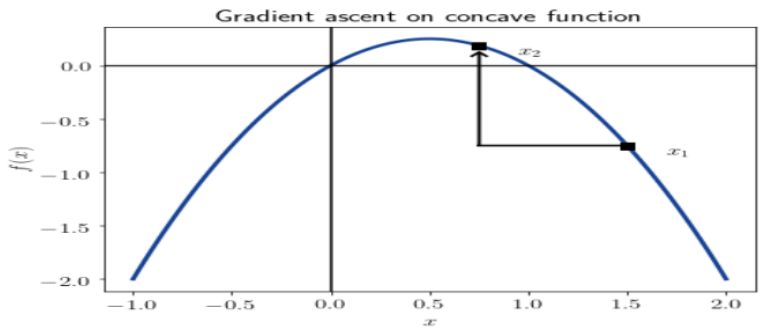


Fig 3.1 - Gradient Ascent

The objective function is concave if we adopt a positive log-likelihood, for which case we must employ gradient ascent. Basically, in, gradient ascent, the only difference is you just change the sign "plus" when you do the update for a single pixel. [21]

You don't do the minus where you use the learning rate and the gradients; you just switch it to "plus."

3.2.1 Cascade Gaussian Smoothing

The Gaussian smoothing operator, a 2-D convolution operator, is being used to "blur" visuals and remove outliers and detail. In this view, it is identical to the mean filter, but it uses a different kernel to mimic a Gaussian (or "bell-shaped") hump. This kernel has certain unique characteristics.

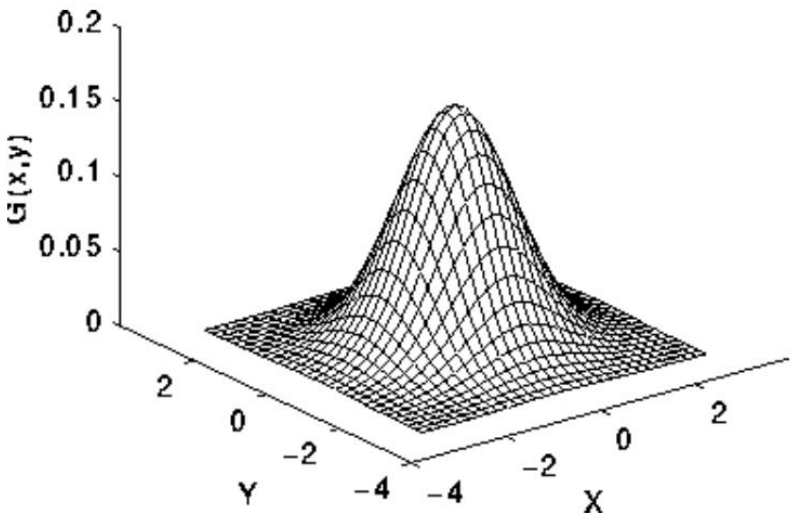


Fig 3.2 - Cascade Gaussian Smoothing

3.2.2 Image Pyramid

We'll be feeding the image in various resolutions into the CNN. By doing that, the network will see different things each time, and that will in return give us a richer output.

Note: that happens because the ratio between the receptive field of the CNN and the input image changes and thus sometimes the net will see the entire image and thus can create global features and in other cases it will see a small portion of the image and it can focus more on the texture.

In order to feed in multiple resolutions, we'll need to define something called an image pyramid.

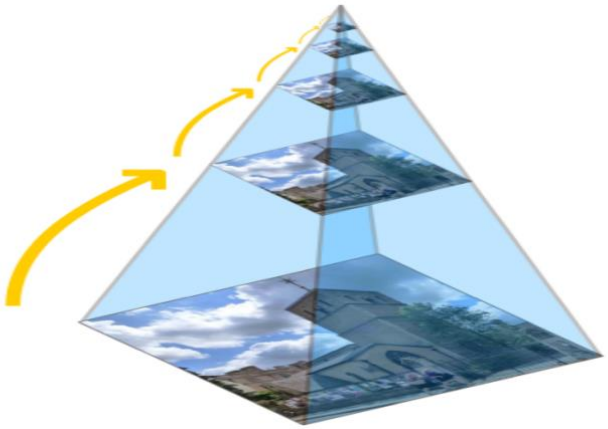


Fig 3.3 - Image Pyramid

3.3 INTRODUCTION OF VGG-16

We utilised a single CNN model called the VGG16. The VGG16 architecture, a variant of the VGG model with 16 convolution layers, has been widely explored. The 16 convolutional layers of the incredibly beautiful VGGNet-16 have a very uniform architecture. Similar to AlexNet, it only uses 3x3 convolutions but has a number of filters. It might be taught for two to three weeks on four GPUs.

The community currently views it as the best method for extracting attributes from snapshots. The weight configuration of the VGGNet is open to the public and is applied as a common feature extractor in many applications and challenges. It could be difficult to manage the 138 million parameters in VGGNet, though.[21]

VGG may be achieved using transfer learning, which entails retraining the model on a dataset and adjusting the parameters for increased accuracy. Afterward, the parameter values can be applied.

3.3.1 Application of VGG-16

There are several computer vision applications that make use of the VGG-16 architecture, including:

1. Image Classification: The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset, where it has been utilised as a benchmark model, is where the VGG-16 model has excelled in picture classification tasks.
2. Object Detection: The region-based CNN (R-CNN) and the single shot detector (SSD), two more models that may be used with VGG-16 to recognise objects in pictures, have also been employed in object identification tasks.
3. Face Recognition: In order to properly identify people and extract information from facial photographs, face recognition algorithms have employed the VGG-16 architecture.
4. Medical Imaging: To categorise different medical pictures, including X-rays and MRI scans, and to spot abnormalities, VGG-16 has been employed in medical imaging applications.
5. Autonomous Vehicles: Autonomous vehicle systems have employed the VGG-16 architecture to identify and categorise things in real-time, such as other cars, pedestrians, and traffic signals.

Overall, the VGG-16 architecture has proven to perform exceptionally well in a variety of computer vision applications and continues to be a common paradigm in the study and creation of new applications.

3.3.2 Architecture of VGG-16

As the name implies, the VGG-16 architecture is a convolutional neural network (CNN) model with 16 layers. The Visual Geometry Group (VGG) at the University of Oxford initially announced it in 2014, and it has since been widely used to a variety of computer vision problems.

This is a thorough description of the VGG-16 architecture:

Input Layer: The input layer receives the picture data and converts it into a 224x224x3 array of RGB channel values.

1. Convolutional Layers: The 13 convolutional layers of the VGG-16 network each have a 3x3 filter size, a 1pixel stride, and zero padding to maintain the spatial dimensions. An activation function called a rectified linear unit (ReLU) follows every convolutional layer.
2. Max Pooling Layers: A max pooling layer is added with a 2x2 filter size and a stride of 2 pixels after every two convolutional layers. By reducing the spatial dimensions of the feature maps while maintaining crucial characteristics, this process helps.
3. Fully Connected Layers: The network's last three levels, which each include 4096 units, are completely linked layers. Following each of these layers is a ReLU activation function.
4. Output Layer: A softmax activation function follows the output layer, which contains 1000 units (corresponding to 1000 ImageNet classes), and normalises the output into probability scores for each class.

The VGG-16 architecture, in summary, is a deep CNN model with 13 convolutional layers, 3 fully connected layers, and max pooling layers. It performs well on a variety of computer vision tasks because it uses tiny filters and deep layers to learn intricate characteristics from the input pictures.[21]

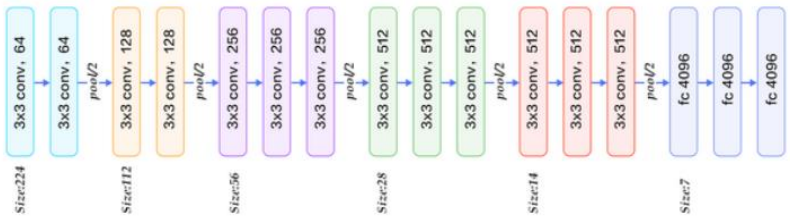


Fig 3.4 - VGG16 Layers

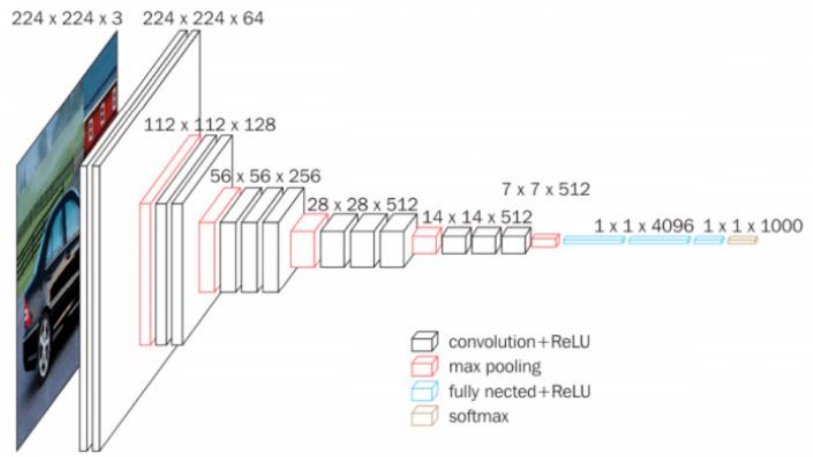


Fig 3.5 - VGG16 Architecture

3.4 INTRODUCTION OF ResNet-50

Convolutional neural network architecture Resnet 50 was initially presented by Microsoft Research in 2015. The Residual Network (ResNet) architecture, which is renowned for its capacity to train extremely deep neural networks, is the source of Resnet 50.[23] Convolutional, pooling, and fully linked layers are among the 50 layers in Resnet 50.

The use of residual connections, which enable the network to learn residual functions rather than attempting to learn the actual mapping function, is one of Resnet 50's significant advances.[23] This contributes to resolving the issue of disappearing gradients, which may occur in deep neural networks and cause the gradients to shrink significantly as they go through the network.

Many computer vision tasks, including image classification, object identification, and image segmentation, have made extensive use of Resnet 50. On a number of benchmark datasets, including ImageNet, COCO, and PASCAL VOC, it has achieved state-of-the-art performance.

3.4.1 Application of RESNET-50

Here are a few typical uses for ResNet-50 :

1. Image Classification: In order to classify photos into multiple categories, such as animals, automobiles, and humans, ResNet-50 may be trained on huge image datasets like ImageNet.
2. Facial Recognition: Based on face traits, ResNet-50 can identify and match individuals. The network can accurately recognise various people by being trained on a vast collection of faces.
3. Medical Imaging: To identify and categorise many kinds of anomalies in medical pictures, such as tumours or other abnormalities in X-rays, MRIs, or CT scans, ResNet-50 can be employed.
4. Autonomous Driving: Building autonomous driving systems requires the ability to recognise and monitor a variety of things on the road, including vehicles, pedestrians, and traffic signals. ResNet-50 can do this.[23]

3.4.2 Architecture of RESNET-50

Microsoft Research introduced ResNet-50 (Residual Network-50), an image categorization convolutional neural network architecture, in 2015. The network has 50 layers, including fully connected, pooling, and convolutional layers.

The convolutional block, the identification block, the down-sampling block, and the classification block are the four basic building components that make up ResNet-50's design.[23]

Convolutional Block:

1. Following batch normalisation and ReLU activation, the convolutional block consists of a convolutional layer with a filter size of 7x7 and a stride of 2. The features from the input picture are extracted using this block.

Identity Block:

2. The spatial resolution of the feature maps is maintained using the identity block. Following batch normalisation and ReLU activation, it has three convolutional layers with filter sizes of 3x3 and a stride of 1 each [23]. To create the residual connection, the input of the block is combined with the output of the final convolutional layer.

Down-Sampling Block:

3. The spatial dimensionality of the feature maps is decreased using the downsampling block. Following batch normalisation and ReLU activation, it has two convolutional layers with a filter size of 3x3 and a stride of 2 each. While the second convolutional layer adds more filters, the first convolutional layer decreases the spatial resolution of the feature maps.

Classification Block:

4. A global average pooling layer, a fully linked layer with 1000 units, and a softmax activation function make up the classification block. In order to provide a fixed-size feature vector that can be utilised for classification, the global average pooling layer averages the feature maps over all spatial dimensions.

Additionally, ResNet-50 has skip connections, also known as residual connections, that aid in solving the vanishing gradient issue. Deeper networks may be trained more easily because of the remaining connections, which enable gradients to pass straight through the network.[23]

ResNet-50 is a potent convolutional neural network design that, in general, has produced cutting-edge outcomes on a variety of image classification tasks, including the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2015.

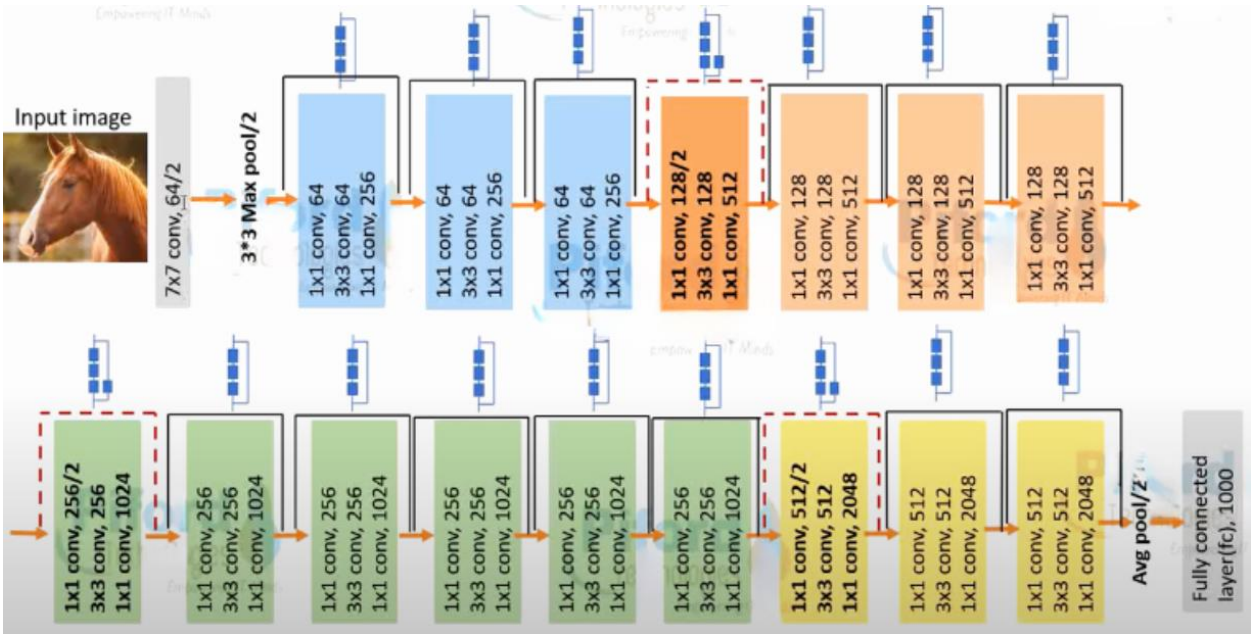


Fig 3.6 - Architecture of ResNet-50

3.5 DIFFERENCE BETWEEN ResNet-50 & VGG-16

Popular deep learning models for image categorization include ResNet50 and VGG16. But there are some significant variations between the two:

1. Architecture: ResNet50's architecture is more complex than VGG16's. ResNet50 contains 50 layers, whereas VGG16 has 16 layers.

2. Skip connections: Skip connections, often referred to as residual connections, are used by ResNet50 to solve the gradient disappearing issue. This indicates that the output of a later layer in the network is added to the input of a certain layer. There are no skip links in VGG16.
3. Computational efficiency: Because skip connections are used, ResNet50 is more computationally efficient than VGG16. This eliminates the issue of vanishing gradients and enables the training of deeper networks.
4. Accuracy: When it comes to accuracy, ResNet50 frequently surpasses VGG16 on a range of picture categorization tasks. However, VGG16 continues to deliver outstanding performance and has been employed in several productive applications.

In conclusion, ResNet50 is a deeper and more effective network from a computational standpoint, and it often achieves greater accuracy than VGG16. VGG16, a less complex model, can yet be useful in some situations.

CHAPTER 4

IMPLEMENTATION OF DEEP DREAM ALGORITHM

4.1 FEATURE DATASET

We have used ImageNet for VGG-16 and MIT places 365 for ResNet-50 which both are open source large stock image database.

4.1.1 ImageNet

With over 14 million pictures and 21,000 categories, ImageNet is a sizable image dataset that serves as a standard for assessing computer vision algorithms. Convolutional neural networks and deep learning for image recognition have benefited from its development.[21] An annual contest called the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) measures how well models perform using the dataset. Machine learning and computer vision have greatly benefited from ImageNet.[21]

4.1.2 MIT PLACES 365

With over 1.8 million photographs of both indoor and outdoor locations labelled with one of 365 scene types, MIT Places 365 is a sizable image collection. It was developed by MIT researchers to offer a standardised dataset for developing and testing scene identification computer vision algorithms. Modern deep learning models for scene identification were trained using the dataset, which contains a wide range of scenes with both global and local labels.

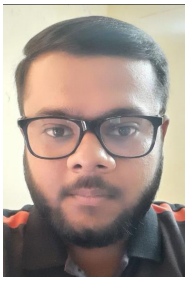
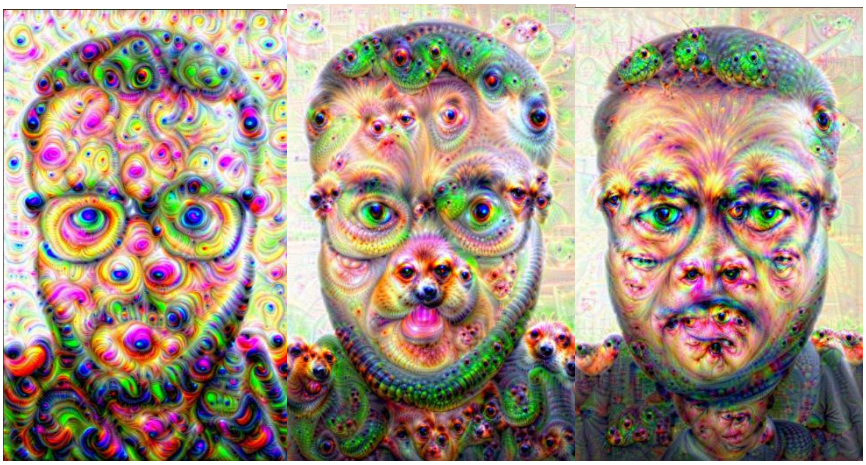


Fig 4.1 – Test Image

4.2 EXPERIMENTAL RESULT USING VGG-16

4.2.1 EXPERIMENTING WITH DIFFERENT LAYERS OF THE VGG 16 MODEL:



1 2 3

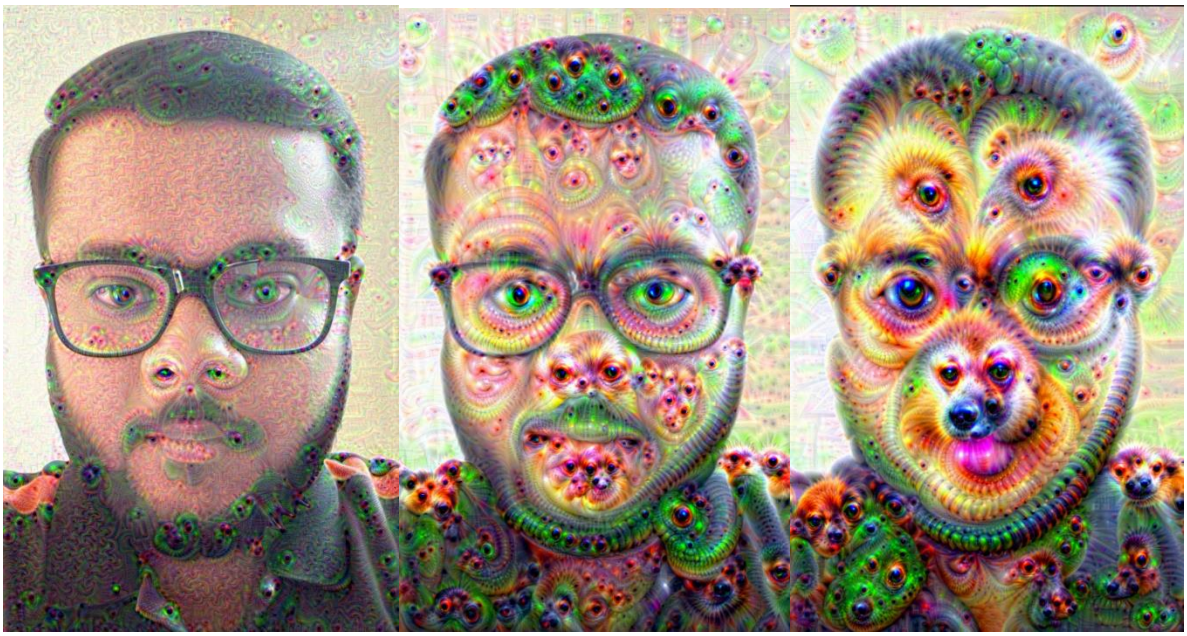
Fig 4.2 - Output of Test Image in different layers of VGG 16

1.VGG16_EXPERIMENTAL_IMAGENET_relu3_3_pyrsize_4_pyrratio_1.8_iter_10_.

2.VGG16_EXPERIMENTAL_IMAGENET_relu4_3_pyrsize_4_pyrratio_1.8_iter_10_

3.VGG16_EXPERIMENTAL_IMAGENET_relu5_1_pyrsize_4_pyrratio_1.8_iter_10_

4.2.2 EXPERIMENTING WITH DIFFERENT PYRAMID SIZE OF THE VGG 16 MODEL:



4 5 6

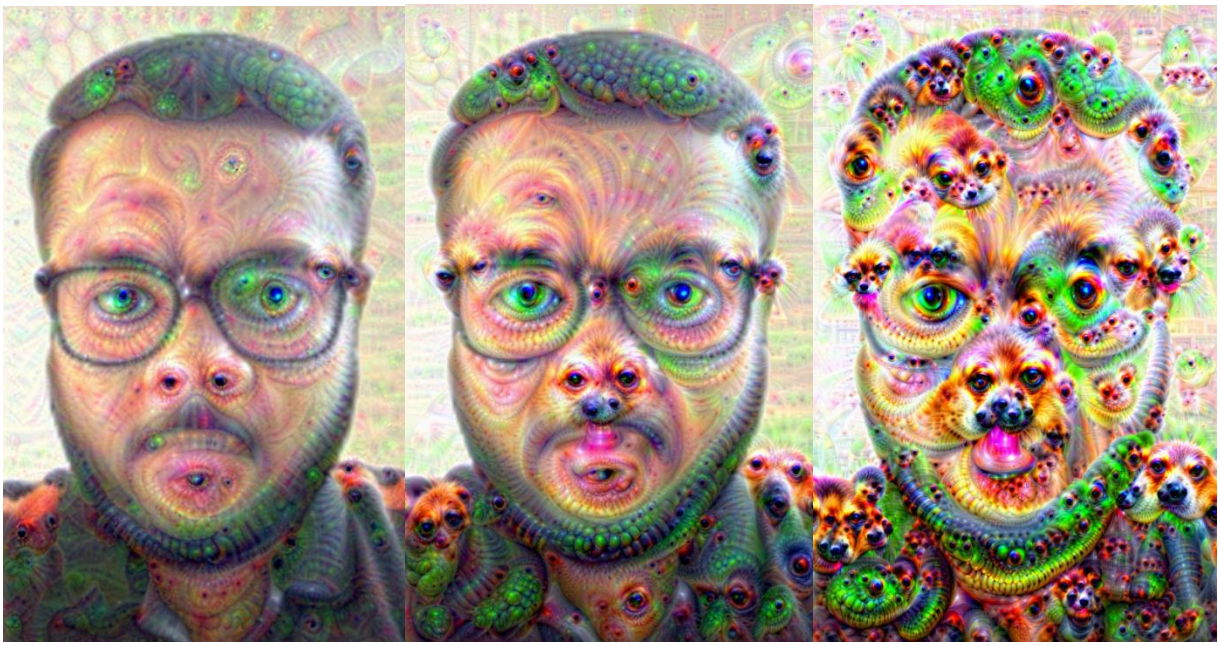
Fig 4.3 – Output of test image with different pyramid size of VGG 16

4. VGG16_EXPERIMENTAL_IMAGENET_relu4_3_pyrsize_1_pyrratio_1.8_iter_10

5. VGG16_EXPERIMENTAL_IMAGENET_relu4_3_pyrsize_3_pyrratio_1.8_iter_10

6. VGG16_EXPERIMENTAL_IMAGENET_relu4_3_pyrsize_5_pyrratio_1.8_iter_10

4.2.3 EXPERIMENTING WITH DIFFERENT ITERATIONS OF THE VGG 16 MODEL:



7 8 9

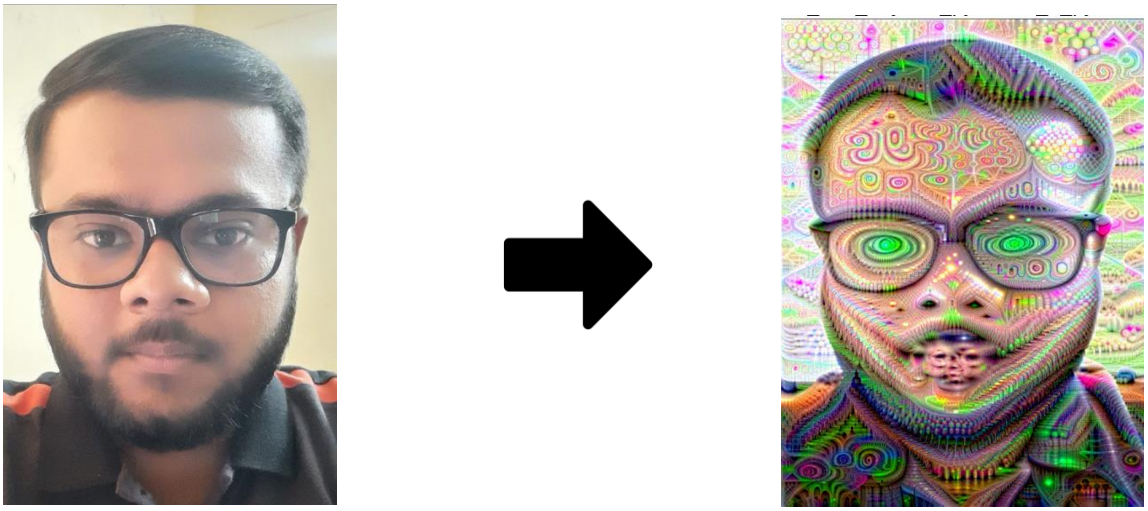
Fig 4.4 - Output of Test Image in different Iterations of VGG 16

7. VGG16_EXPERIMENTAL_IMAGENET_relu4_3_pyrsize_4_pyrratio_1.8_iter_2

8. VGG16_EXPERIMENTAL_IMAGENET_relu4_3_pyrsize_4_pyrratio_1.8_iter_5

9. VGG16_EXPERIMENTAL_IMAGENET_relu4_3_pyrsize_4_pyrratio_1.8_iter_20

4.3 EXPERIMENTAL RESULT USING ResNet-50

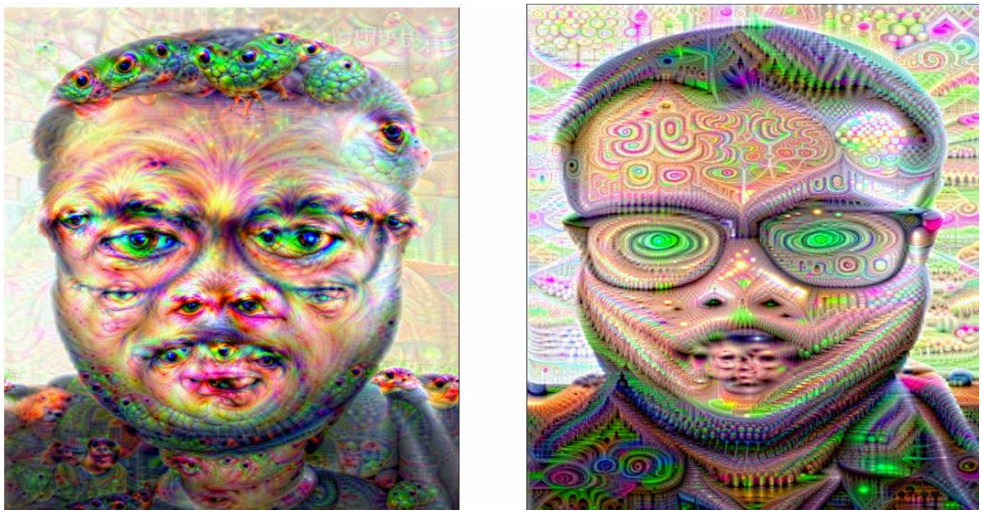


(10)

Fig 4.5 - Output using ResNet-50

10. model_RESNET50_PLACES_365_layer3_pyrsize_4_pyrratio_1.8_iter_10

4.4 COMPARISON OF RESULTS



(11) (12)

Fig 4.6 - Comparison of output

11. VGG16_EXPERIMENTAL_IMAGENET_relu4_3_pyrsize_4_pyrratio_1.8_iter_20

12. model_RESNET50_PLACES_365_layer3_pyrsize_4_pyrratio_1.8_iter_10

4.5 COMPARISON OF BOTH MODELS

4.5.1 Training & Validation

In machine learning, a model is trained by providing it with input data and expected output data so that it can recognise patterns in the data and change its internal parameters as necessary. The end goal of training is to create a model that can correctly forecast results for novel, unforeseen input data.[24]

Instead of only memorising the input-output pairings from the training data, it is crucial to make sure that the trained model generalises successfully to new data. The idea of validation enters the picture at this point.

Examining the trained model's performance on a different dataset, referred to as the validation set, is the process of validation. The data distribution that the model will face in the actual world should be reflected in this collection. We can predict how well the model will perform on brand-new, untested data by measuring its performance on this validation set.

The following phases are often included in the training and validation process:

1. Dividing the available data into training and validation sets.
2. Training the model on the training set.
3. Evaluating the model's performance on the validation set.
4. Adjusting the model's hyperparameters and/or architecture based on the validation results.
5. Repeating steps 2-4 until satisfactory performance is achieved.[24]

In order to prevent overfitting, which occurs when a model performs well on training data but badly on fresh data, a separate validation set is used to evaluate the model's performance. It makes that the model adapts adequately to fresh inputs.[24]

4.5.2 Training Accuracy & Training Loss

Two important measures used in machine learning to assess a model's performance throughout the training process are training accuracy and training loss.

Training accuracy, which is often stated as a percentage, shows how effectively the model predicts the right results for the data it was trained on. By modifying the model's internal parameters to minimise mistakes on the training data, the goal of training is to enhance the model's training accuracy.[25]

Contrarily, training loss quantifies the discrepancy or mismatch between each training example's actual results and the model's expected results. The average of the mistakes across all training samples is displayed as a single value.[25] In order to improve the model's fit to the training data, the purpose of training is to reduce the training loss.

High training accuracy and minimal training loss are desirable, but they do not ensure that the model will function effectively when presented with fresh, untested data. To make sure that the model can generalise successfully to new data, it is important to assess its performance on a different validation set.[25]

In general, training accuracy and training loss offer crucial information about how effectively the model is recognising the underlying trends in the training data and modifying its internal parameters to produce reliable predictions.

4.5.3 Validation Accuracy & Validation Loss

In machine learning, validation accuracy and validation loss are two crucial measures used to assess how well a model performs on brand-new, untried data during training.[25]

The model's ability to correctly predict the outcomes for validation data, which consists of fresh information that it was not exposed to during training, is measured by validation accuracy. Typically, a percentage is used to indicate it.[25]

On the other side, validation loss gauges the discrepancy or mismatch between each validation example's actual output and the model's anticipated output. The average of the mistakes across all of the validation samples is shown as a single value.

Monitoring validation accuracy and validation loss has as its main objective the prevention of overfitting, which is when a model performs well on training data but badly on fresh, untested data.[25] We may assess if the model is overfitting and modify its internal parameters by measuring the model's performance on a different validation set during the training phase.

The model should ideally have high validation accuracy and minimal validation loss, demonstrating good generalizability to new data.[25] Although it might be difficult to achieve high validation accuracy and low validation loss, doing so needs the model to understand the underlying patterns in the data and guard against overfitting.

In conclusion, validation accuracy and validation loss are crucial measures for assessing a model's ability to generalise throughout the training phase.[25] We can prevent overfitting by keeping an eye on these indicators and ensuring that the model generalises properly to fresh, untested data.

4.5.4 Training & Validation Datasets

Fig 4.7 - First dataset for training & validation

Fig 4.8 - Second dataset for training & validation

4.5.5 Output: Training & Validation

```
Epoch 1/8
100/100 [==============================] - 922s 9s/step - loss: 0.1117 - accuracy: 0.9620 - val_loss: 0.0481 - val_accuracy: 0.9840
Epoch 2/8
100/100 [==============================] - 871s 9s/step - loss: 0.0381 - accuracy: 0.9845 - val_loss: 0.0511 - val_accuracy: 0.9810
Epoch 3/8
100/100 [==============================] - 899s 9s/step - loss: 0.0187 - accuracy: 0.9940 - val_loss: 0.0552 - val_accuracy: 0.9810
Epoch 4/8
100/100 [==============================] - 837s 8s/step - loss: 0.0105 - accuracy: 0.9975 - val_loss: 0.0476 - val_accuracy: 0.9790
Epoch 5/8
100/100 [==============================] - 822s 8s/step - loss: 0.0065 - accuracy: 0.9980 - val_loss: 0.0541 - val_accuracy: 0.9840
Epoch 6/8
100/100 [==============================] - 823s 8s/step - loss: 0.0026 - accuracy: 1.0000 - val_loss: 0.0513 - val_accuracy: 0.9850
Epoch 7/8
100/100 [==============================] - 817s 8s/step - loss: 0.0021 - accuracy: 0.9995 - val_loss: 0.0579 - val_accuracy: 0.9820
Epoch 8/8
100/100 [==============================] - 818s 8s/step - loss: 7.3018e-04 - accuracy: 1.0000 - val_loss: 0.0578 - val_accuracy: 0.9850
100/100 [==============================] - 285s 3s/step - loss: 0.0578 - accuracy: 0.9850
[INFO] loss=0.0578, accuracy: 98.5000%
```

Fig 4.9 - Training & Validation: Accuracy & Loss (VGG-16)

```
Epoch 1/15
58/58 [==============================] - 289s 5s/step - loss: 0.1986 - accuracy: 0.9380 - val_loss: 0.0690 - val_accuracy: 0.9830
Epoch 2/15
58/58 [==============================] - 283s 5s/step - loss: 0.0557 - accuracy: 0.9875 - val_loss: 0.0480 - val_accuracy: 0.9870
Epoch 3/15
58/58 [==============================] - 282s 5s/step - loss: 0.0385 - accuracy: 0.9915 - val_loss: 0.0415 - val_accuracy: 0.9870
Epoch 4/15
58/58 [==============================] - 283s 5s/step - loss: 0.0286 - accuracy: 0.9955 - val_loss: 0.0366 - val_accuracy: 0.9880
Epoch 5/15
58/58 [==============================] - 282s 5s/step - loss: 0.0229 - accuracy: 0.9970 - val_loss: 0.0342 - val_accuracy: 0.9880
Epoch 6/15
58/58 [==============================] - 283s 5s/step - loss: 0.0188 - accuracy: 0.9985 - val_loss: 0.0321 - val_accuracy: 0.9900
Epoch 7/15
58/58 [==============================] - 282s 5s/step - loss: 0.0157 - accuracy: 0.9990 - val_loss: 0.0317 - val_accuracy: 0.9880
Epoch 8/15
58/58 [==============================] - 282s 5s/step - loss: 0.0136 - accuracy: 0.9980 - val_loss: 0.0315 - val_accuracy: 0.9870
Epoch 9/15
58/58 [==============================] - 282s 5s/step - loss: 0.0114 - accuracy: 0.9995 - val_loss: 0.0328 - val_accuracy: 0.9870
Epoch 10/15
58/58 [==============================] - 283s 5s/step - loss: 0.0095 - accuracy: 1.0000 - val_loss: 0.0302 - val_accuracy: 0.9890
Epoch 11/15
58/58 [==============================] - 283s 5s/step - loss: 0.0080 - accuracy: 1.0000 - val_loss: 0.0296 - val_accuracy: 0.9880
Epoch 12/15
58/58 [==============================] - 282s 5s/step - loss: 0.0072 - accuracy: 1.0000 - val_loss: 0.0289 - val_accuracy: 0.9890
Epoch 13/15
...
Epoch 15/15
58/58 [==============================] - 283s 5s/step - loss: 0.0052 - accuracy: 1.0000 - val_loss: 0.0285 - val_accuracy: 0.9900
100/100 [==============================] - 102s 1s/step - loss: 0.0285 - accuracy: 0.9900
[INFO] loss=0.0285, accuracy: 99.0000%
```

Fig 4.10 - Training & Validation : Accuracy & Loss (ResNet-50)

4.5.6 Visualizing: Training & Validation

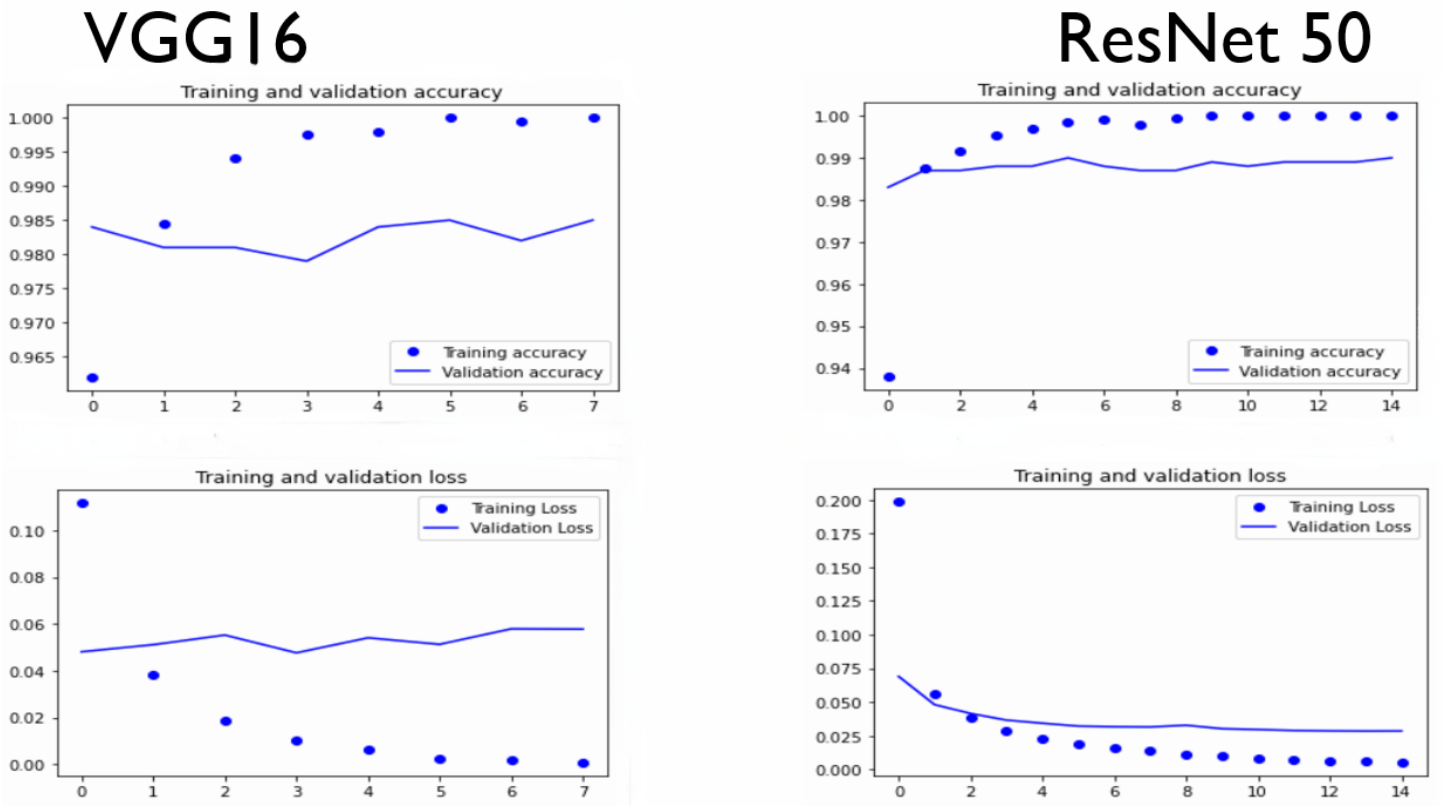


Fig 4.11 - Visualization of Training & Validation

CHAPTER 5

NEURAL STYLE TRANSFER

5.1 INTRODUCTION

A deep learning approach called neural style transfer enables the creation of images that mix the content of one image with the style of another image. Utilising a convolutional neural network (CNN) that has already been trained to recognise objects and other elements in pictures forms the basis of the approach.[26] To create a picture that matches the content of one image and the style of another, we may change the CNN's input and alter the network's output.

Fig 5.1 - Sample Result of NST

We start with a content image and a style image before doing Neural Style Transfer. The content image is often a picture or image whose content we want to keep. On the other hand, the style picture is a representation of a specific aesthetic style that we wish to apply to the content image.[26]

The characteristics of the content and style pictures are then extracted using a pre-trained CNN, such as VGG19 or ResNet50. Then, we can create a new image by matching the content of the original image and the style of the style image using these feature representations.

Numerous applications, like as picture and video editing, creative rendering, and even fashion design, have exploited neural style transfer [26]. In order to create realistic photographs of objects and settings for use in virtual and augmented reality applications, the technology has also been modified.

5.1.1 Application of Neural Style Transfer (NST)

Neural Style Transfer has several uses in many different industries, such as:

1. Image and Video Editing: Adding artistic styles to photos and videos with Neural Style Transfer may give them a distinctive and imaginative appearance. In pictures and movies, it may also be used to change or eliminate the backdrop.[26]
2. Artistic Rendering: By fusing the style of one image with the content of another, Neural Style Transfer may create fresh, original pieces of art. This has uses in the advertising, graphic design, and fashion design industries.
3. Virtual and Augmented Reality: In virtual and augmented reality applications, Neural Style Transfer may be used to create realistic pictures of items and settings. This may be applied to provide people more realistic and immersive experiences.
4. Medical Imaging: Medical pictures like X-rays and CT scans can be improved through neural style transfer to make them simpler to comprehend and analyse. For training and instructional reasons, it may also be utilised to provide realistic representations of inside organs and tissues.
5. Natural Language Processing: Text may be created via neural style transfer that mimics the writing style of a certain author or genre. Applications like sentiment analysis, automatic content production, and chatbots may all make advantage of this.[26]

Overall, Neural Style Transfer is a strong tool for creating innovative and original material in a variety of sectors and has numerous possible uses.

CHAPTER 6

WORKING

6.1 WORKING OF NST

The working of Neural Style Transfer typically involves the following steps:

1. Define the Content and Style Images: The choice of the content and style pictures to be used for the style transfer is the first stage. The content image is the one whose content we want to keep, but the style image is the one whose style we wish to borrow.[26]
2. Define a Pre-Trained CNN: The following step is to create a pre-trained convolutional neural network (CNN), such as VGG19 or ResNet50 (we chose VGG19), that has been trained on a sizable dataset. The content and stylistic elements from the input photographs will be extracted using this CNN.
3. Extract Content and Style Features: The content and style characteristics are extracted from the content and style photos using CNN. The style features are collected from the network's intermediate levels, whereas the content features are extracted from one of the network's later layers.[27]
4. Define a Loss Function: The output picture will be optimised using a loss function, which must be defined next. The loss function typically consists of two parts: a content loss that calculates the difference between the input image's content features and the output image's content features, and a style loss that computes the difference between the input image's style features and the output image's style features.
5. Optimize the Output Image: The generated picture must then be optimised using a loss function-minimizing technique, such as gradient descent.[26] The optimisation technique modifies the output image's pixel values to reduce content and style loss.

In general, Neural Style Transfer's design entails utilising a pre-trained CNN to extract content and style features from input photos, creating a loss function to calculate the difference between input and output images, and then optimising the output image to reduce the loss function.

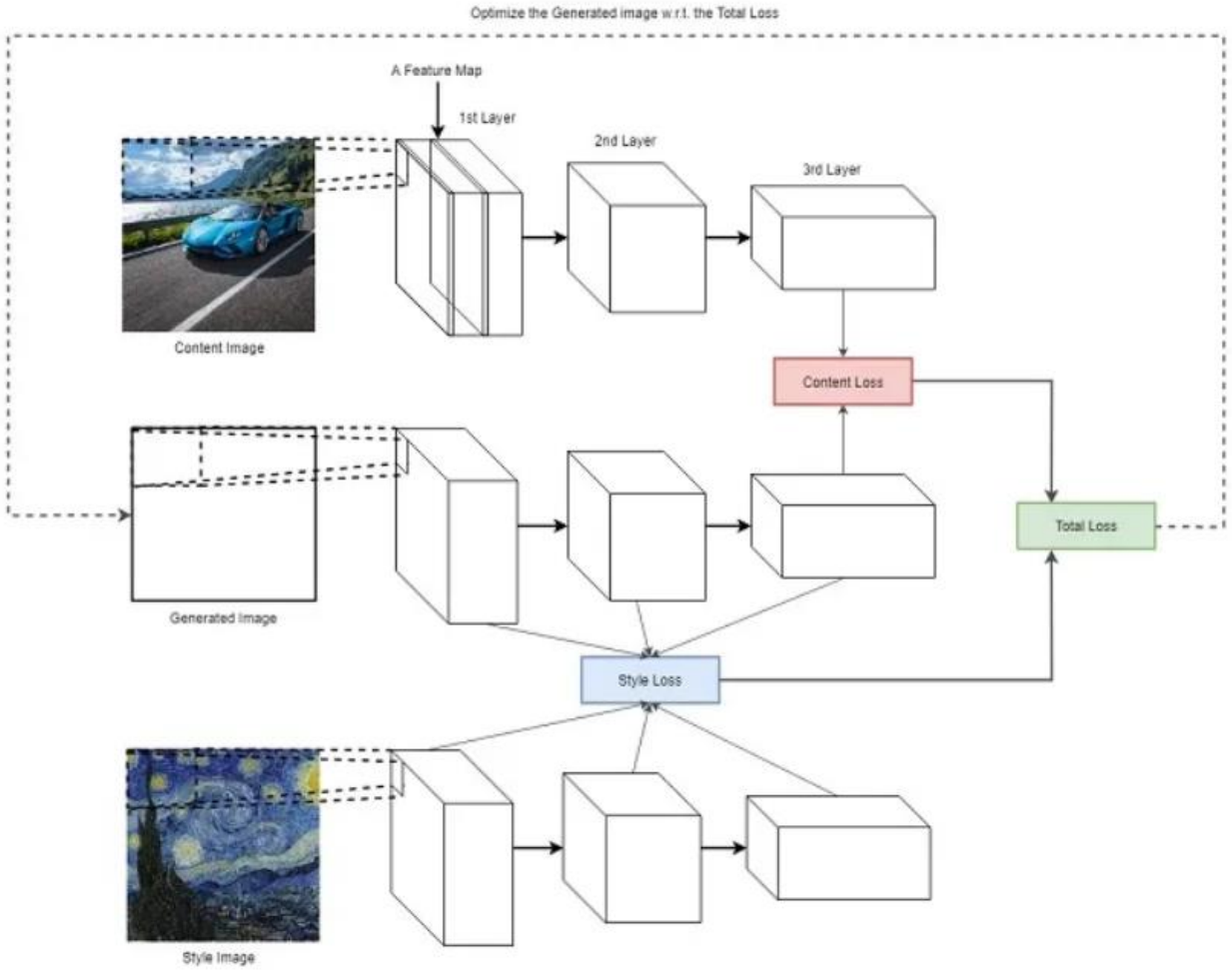


Fig 6.1 – Architecture of NST

6.1.1 Magenta Arbitrary Image Stylization

A deep neural network-based machine learning model called the Magenta Arbitrary Image Stylization model, developed by Google and published on TensorFlow Hub, is used to artistically

stylize photos. The model can generalise and apply a broad variety of creative styles to new photos because it was trained on a big dataset of images and artistic styles.[28]

It is a versatile tool for creative picture stylization since it is simple to integrate into other applications and may be improved utilising transfer learning techniques.[28] The model is helpful for designers, developers, and artists who want to employ deep learning to produce stylised graphics.

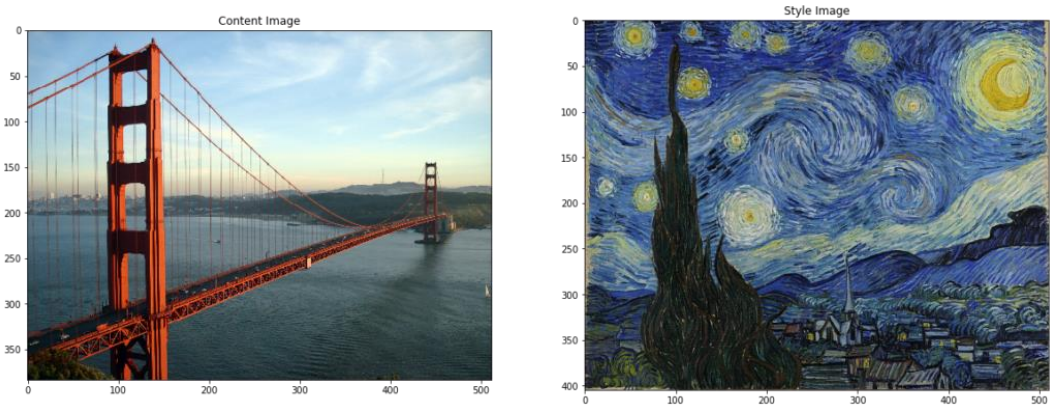


Fig 6.2 - Result from Magenta Arbitrary Image Stylization

6.1.2 Content & Style Features

In neural style transfer, technique for artistically stylizing pictures, there are two unique categories of characteristics: content features and style features. Content features, which are representations of an image's content, are often derived from a convolutional neural network (CNN) that has already been trained.[26] These qualities don't pay attention to minute aspects like colour and texture, instead focusing on an image's main elements like its objects, forms, and general structure.

The texture and patterns of an artistic style, however, are captured by style features. They are often derived from a pre-trained CNN by examining the statistics of the network's various layer activations.[26] Style elements, such as brush strokes, colour schemes, and other visual components that distinguish one style from another, are the distinctive textures and patterns of an artistic style.

The main goal of neural style transfer is to combine the style features of a reference style picture with the content features of an input image to produce a new stylized image that retains the reference image's style while maintaining the input image's content.[27] This is accomplished by fine-tuning a loss function that strikes a compromise between maintaining the content of the input image and incorporating the characteristics of the style image.

In conclusion, content and style features are important for neural style transfer because they enable the fusion of the content and style of two separate pictures to produce distinctive and new styled images.

6.1.3 Loss Function

The discrepancy between the content and style elements of two images is calculated using a mathematical calculation called the loss function, which is employed in neural style transfer.[27] This loss function aims to minimise the difference between the stylized picture's content features and those of the input image while simultaneously minimising the difference between the stylized image's style characteristics and those of the style reference image.

The content loss and the style loss are the two fundamental parts of the loss function. The difference between the input image's and the styled image's content characteristics is calculated using the concept of content loss.[27] The difference between the style attributes of the stylized picture and the style reference image is measured by the style loss, on the other hand.

The mean squared error (MSE) function, which gauges the separation between the input picture's feature maps and the styled image, is used to calculate the content loss. A lower MSE value

indicates a better match between the content features of the input and styled pictures. The MSE function calculates the average difference between the two sets of feature maps.[27]

The Gramme matrix, which gauges the correlations between the activations of various filters in a particular layer of a convolutional neural network (CNN), is used to calculate the style loss.[27] Then, it calculates the distance between the Gramme matrices of the styled picture and the style reference image, with a smaller difference indicating a better similarity between the two images' stylistic attributes.

The weights used to calculate the total loss function determine the proportional relevance of each component. The total loss function is defined as the weighted sum of the content loss and the style loss.[26] Using an iterative approach like gradient descent, the stylized image's pixel values are adjusted as part of the optimization process to reduce the total loss function.

In conclusion, the optimization of the stylized picture to strike a balance between content preservation and style adoption depends on the loss function in neural style transfer, which is essential for determining the difference between the content and style elements of two images.[28]

$$L_{content} = \frac{1}{2} \sum_{i,j} (A_{ij}^{l}(g) - A_{ij}^{l}(c))^2$$

$$L_{style} = \sum_{l} w^{l} L_{style}^{l} \text{ where,}$$

$$L_{style}^{l} = \frac{1}{M^{l}} \sum_{ij} (G_{ij}^{l}(s) - G_{ij}^{l} g)^2 \text{ where,}$$

$$G_{ij}^{l}(I) = \sum_{k} A_{ik}^{l}(I) A_{jk}^{l}(I).$$

$$L = \alpha L_{content} + \beta L_{style}$$

6.1.4 Gradient Descent

In machine learning, gradient descent is a popular optimization approach. Through an iterative process of updating the model parameters based on the gradient of the loss function with respect to the parameters, it is used to minimize the loss function of a model.[29] Until the loss is minimized or converges, the method adjusts the parameters in the opposite direction of the loss function's slope.[29]

The methods used in gradient descent come in a variety of forms, including batch, stochastic, and mini-batch algorithms.[30] While stochastic gradient descent computes the gradient for each individual example in the training set, batch gradient descent computes the gradient for the whole training set. A middle ground between the two is mini-batch gradient descent, in which just a tiny portion of the training set is used to compute the gradient.[30] In order to train neural networks and other machine learning models, gradient descent is frequently utilized. It is a crucial optimization procedure that iteratively changes the model's parameters to increase the model's correctness.[30]

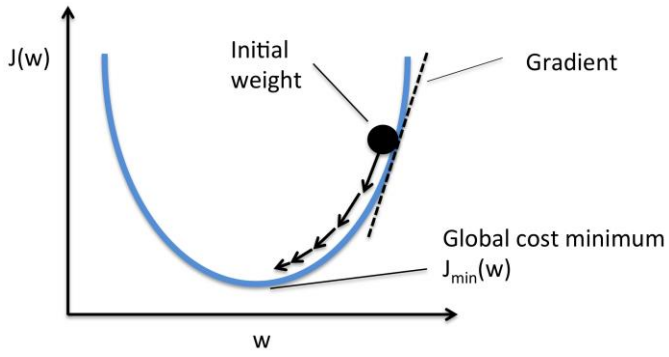


Fig 6.3 - Gradient Descent Graph

6.1.5 Content & Style Reconstruction

In Neural Style Transfer (NST), content and style reconstruction entails creating a new picture by combining the content of one image with the style of another. The underlying items and their spatial arrangement are referred to as an image's content, whilst the texture, color, and other visual elements that contribute to its distinctive look are referred to as its style. In NST, a neural network is trained to extract these content and style attributes from two input photos and merge them into a new image while retaining the content and style of each individual image.

A convolutional neural network (CNN) is utilized in the style transfer process to extract the underlying content characteristics from the content picture. The style picture is similarly processed by the same CNN, but its attributes are utilized to determine the image's style. The content characteristics are then modified by these style elements to produce a new picture that combines the style of the style image with the original image's content.

NST comes in a variety of forms that perform content and style reconstruction using various CNN architectures, optimization methods, and loss functions. Neural style transfer developed by Gatys et al., quick neural style transfer by Johnson et al., and texture networks developed by Ulyanov et al. are noteworthy methods. These techniques have been used for a variety of purposes, including picture manipulation, creative rendering, and visual effects in movies and television.

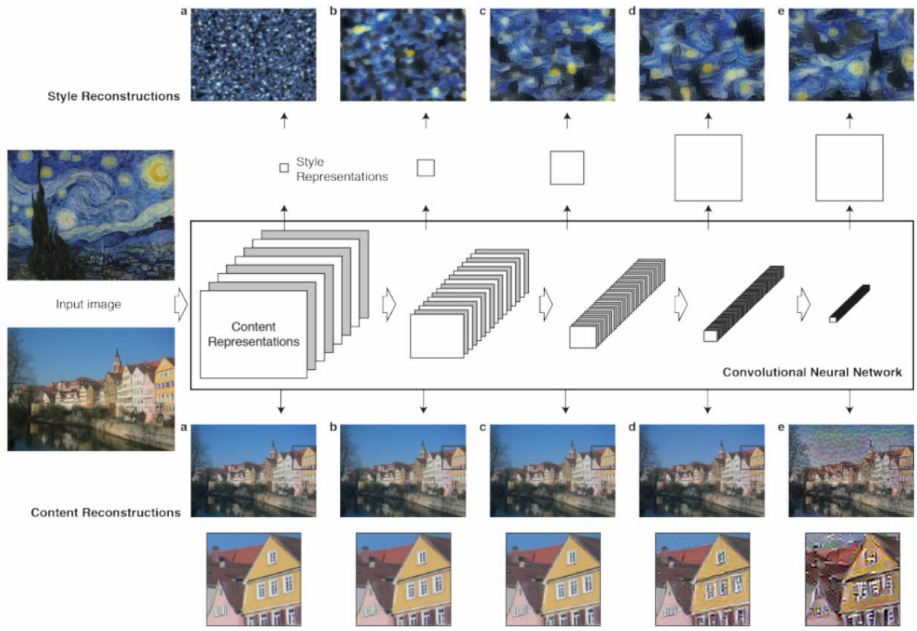


Fig 6.4 - Content and Style reconstruction

6.2 INTRODUCTION VGG 19

The Visual Geometry Group at the University of Oxford built the deep convolutional neural network VGG19. The VGGNet family of neural networks, which has been widely applied in computer vision applications, forms the foundation of the architecture of VGG19.[31]

VGG19 has 19 layers, comprising 3 fully connected layers, 5 max-pooling layers, and 16 convolutional layers. The network accepts RGB pictures with an input size of 224x224 pixels.[31] The spatial information of the input pictures is preserved through the use of tiny 3x3 filters with a stride of 1 in the convolutional layers of VGG19.

The ImageNet dataset, which includes more than 1.2 million photos divided into 1000 distinct categories, served as the basis for training VGG19.[31] Using supervised learning, the network was trained, and its weights were changed to reduce the cross-entropy loss between the true and predicted labels of the training set's pictures.[31]

Among other computer vision tasks, VGG19 has achieved state-of-the-art performance in picture classification, object identification, and semantic segmentation.[31] In several applications of transfer learning, the network has also been utilised as a feature extractor, where the features obtained are sent into another neural network to perform a particular job.[31]

Overall, VGG19 is a powerful neural network architecture that has been widely used in computer vision applications due to its strong performance and simple architecture.

6.2.1 Application of VGG 19

VGG19 has many applications in computer vision, including:

1. Image Classification: For image classification tasks like locating objects in pictures, VGG19 is frequently utilised. On a number of picture classification benchmarks, notably the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), the network has demonstrated cutting-edge performance.[31]
2. Object Detection: VGG19 has also been employed for object identification, which entails spotting and locating items in a picture. In several object identification pipelines, the network has been employed as a feature extractor, with the features collected from the network being used as input to another neural network for object detection.
3. Semantic Segmentation:Semantic segmentation, which involves providing each pixel in an image a label, has also been employed using VGG19.[31]In several semantic segmentation pipelines, the network has been employed as a feature extractor, with the features collected from the network being used as input to another neural network for semantic segmentation.
4. Transfer Learning: Transfer learning, in which the weights of the network are utilised as a starting point for training another neural network on a new task or dataset, has also been extensively used to VGG19.[31] Even when the dataset being utilised is substantially less than the initial dataset used to train VGG19, the features produced from the network can be used to enhance the performance of other computer vision tasks.

VGG19 is a potent neural network design that may be used for a wide range of computer vision tasks, such as semantic segmentation, transfer learning, object identification, and picture categorization.

6.3 ARCHITECTURE OF VGG 19

The architecture of VGG19 consists of 19 layers, including 16 convolutional layers, 3 fully connected layers, and 5 max-pooling layers. The input to the network is a 224x224 RGB image.

Here's a breakdown of the layers in VGG19:

1. Convolutional layer with 64 filters, kernel size 3x3, and stride 1
2. Convolutional layer with 64 filters, kernel size 3x3, and stride 1
3. Max pooling layer with pool size 2x2 and stride 2
4. Convolutional layer with 128 filters, kernel size 3x3, and stride 1
5. Convolutional layer with 128 filters, kernel size 3x3, and stride 1
6. Max pooling layer with pool size 2x2 and stride 2
7. Convolutional layer with 256 filters, kernel size 3x3, and stride 1
8. Convolutional layer with 256 filters, kernel size 3x3, and stride 1
9. Convolutional layer with 256 filters, kernel size 3x3, and stride 1
10. Convolutional layer with 256 filters, kernel size 3x3, and stride 1
11. Max pooling layer with pool size 2x2 and stride 2
12. Convolutional layer with 512 filters, kernel size 3x3, and stride 1
13. Convolutional layer with 512 filters, kernel size 3x3, and stride 1
14. Convolutional layer with 512 filters, kernel size 3x3, and stride 1
15. Convolutional layer with 512 filters, kernel size 3x3, and stride 1
16. Max pooling layer with pool size 2x2 and stride 2
17. Fully connected layer with 4096 units
18. Fully connected layer with 4096 units
19. Fully connected layer with 1000 units

The spatial information of the input pictures is preserved through the use of tiny 3x3 filters with a stride of 1 in the convolutional layers in VGG19. The spatial resolution of the feature maps is decreased by the usage of the max pooling layers, which employ a pool size of 2x2 and a stride of 2.

With the exception of the last layer, which has 1000 units to match to the number of classes in the ImageNet dataset, each of the fully connected layers in VGG19 has 4096 units. Using supervised learning, the VGG19 network was trained on the ImageNet dataset, with the weights of the network being modified to reduce the cross-entropy loss between the predicted and actual labels of the training set's pictures.[31]

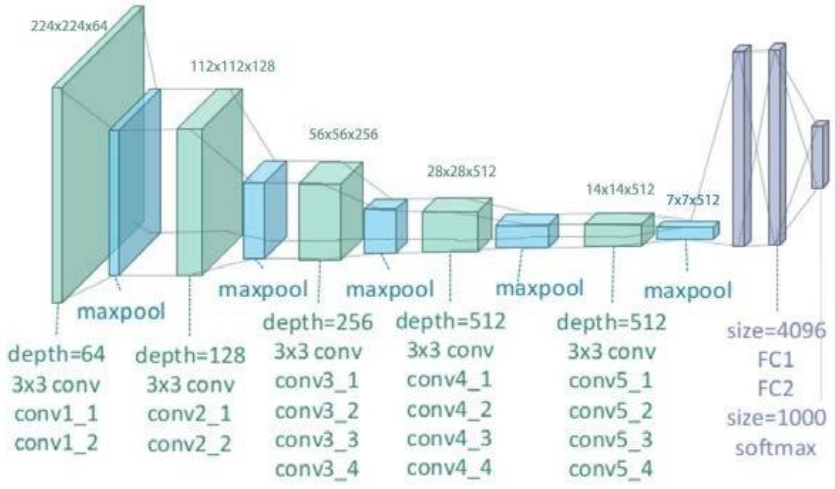


Fig 6.5 - VGG-19's architectural design

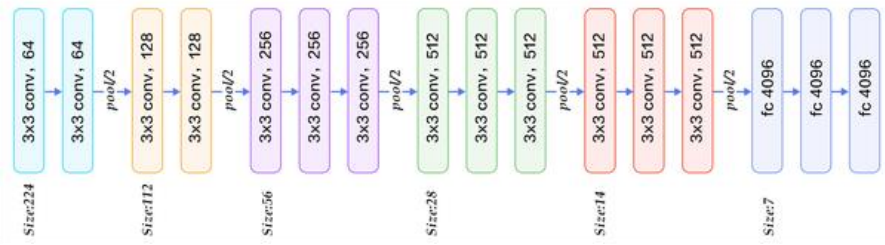


Fig 6.6 - Different layers of VGG-19

CHAPTER 7

IMPLEMENTATION OF NEURAL STYLE TRANSFER ALGORITHM

7.1 IMPLEMENTATION USING MAGENTA ARBITRARY IMAGE STYLIZATION

Above mentioned library is published by google on tensor flow hub.

Fig 7.1 - First result

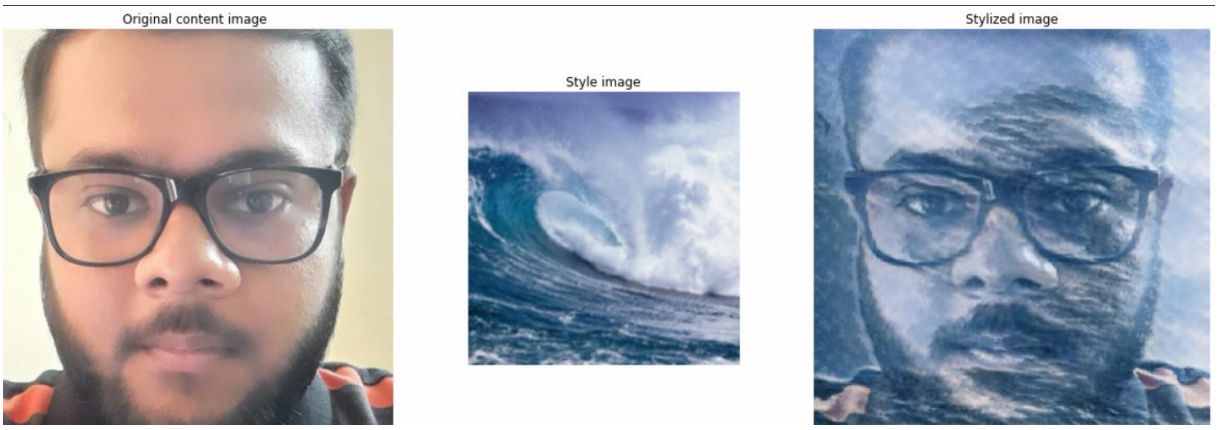


Fig 7.2 - Second Result

7.2 IMPLEMENTATION USING VGG-19

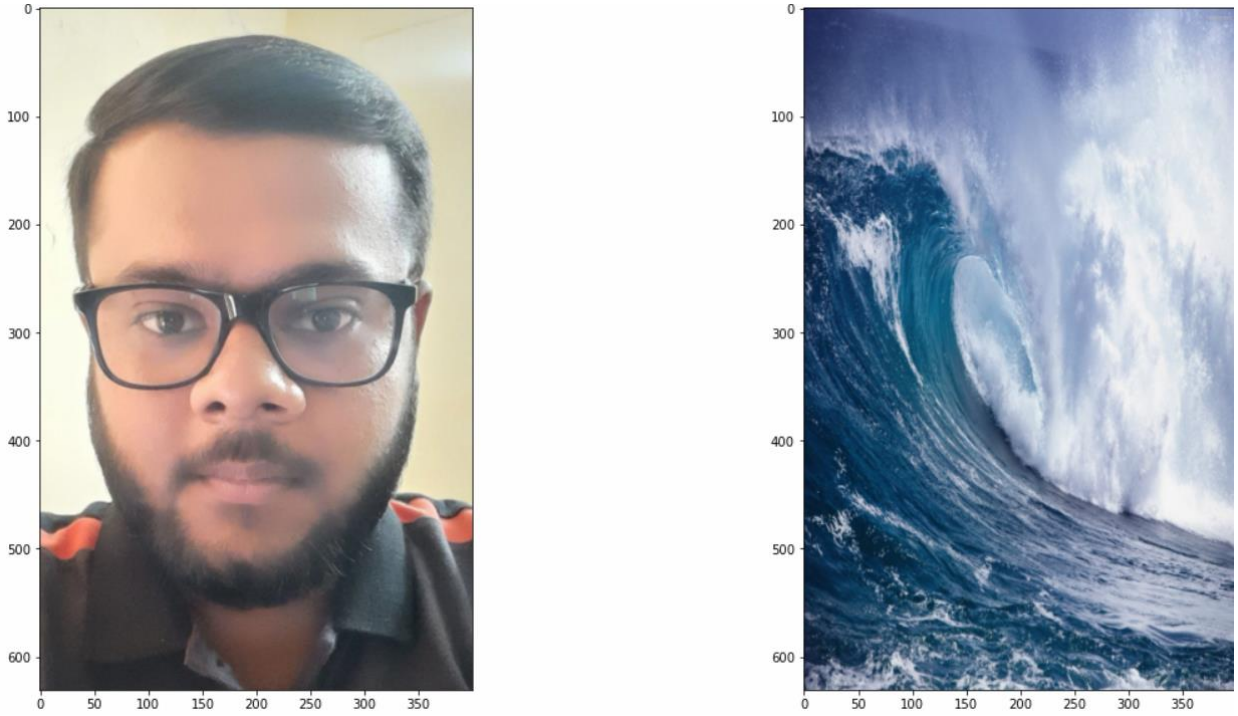


Fig 7.3 - Content image and Style image

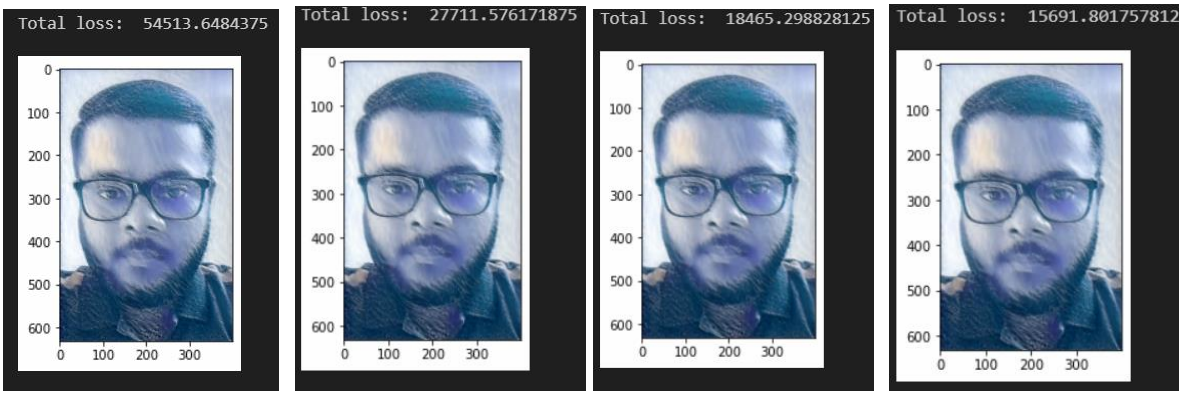


Fig 7.4 - Total loss decreasing after each passing iteration

7.2.1 Final Output

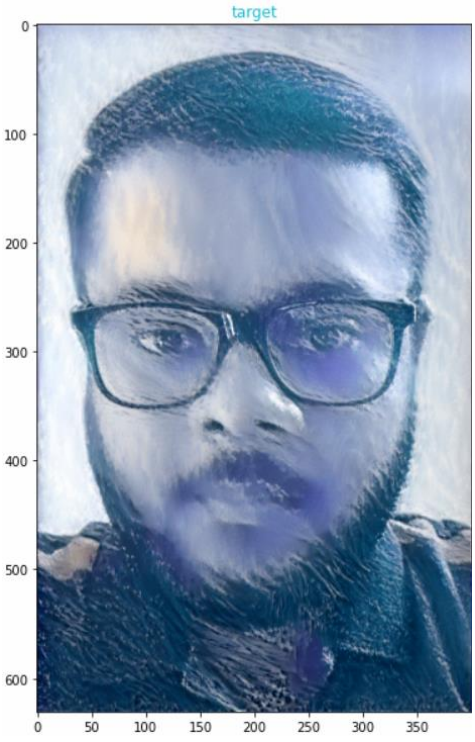


Fig 7.5 - Final result

CONCLUSION

The concept of dreaming may be used to explore hidden (internal) neurons other than those in the output, enabling investigation of the functions and representations of various network components. Adding "dreamed" inputs to the training set may shorten training durations for abstractions, even though dreaming is often employed to visualise networks or create computer art. This serves as the foundation for the DeepDream idea and may be used for visualisations to better comprehend the emergent structure of the neural network.

The method of neural style transfer has attracted a lot of interest lately. In order to create a new image that incorporates both styles, deep neural networks are used to transfer the style of one image to the content of another. This approach generates original and eye-catching graphics using convolutional neural networks and optimization techniques, making it a handy tool for designers and artists. Several applications, such as picture augmentation, image creation, and video stylization, have made use of it. Neural style transfer has advantages, but it also has drawbacks and difficulties. For example, the procedure could take a while, and there's a chance of overfitting. The method is still being developed and has the potential to revolutionise industries including computer vision, art, and design.Overall, neural style transfer is a strong and imaginative method that has drawn the attention of both academics and artists. It is a promising field for future investigation due to its capacity for further innovation.

FUTURE SCOPE

Deep Dream and Neural Style Transfer may have important uses in a variety of sectors in the future. They may be used to provide breathtaking visual effects for entertainment, improve creative expression, alter photos and videos for marketing and advertising, and support medical image analysis. Additionally, these methods might be useful in augmented and virtual reality, enabling users to customise their virtual settings. Deep Dream and Neural Style Transfer may develop into indispensable tools for the creative sector, academic study, and even daily living since they provide novel approaches to visual perception and manipulation.

REFERENCES

[1] K. Suzuki, W. Roseboom, D. J. Schwartzman et al., "A Deep-Dream Virtual Reality Platform for Studying Altered Perceptual Phenomenology," IEEE Access, vol. 7, pp. 15982-15990, 2019.

[2] A. Mordvintsev, C. Olah, and M. Tyka, "DeepDream - a code example for visualizing Neural Networks," Google Research, 2015 .

[3] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going Deeper into Neural Networks," Google Research, 2015 .

[4] D. Culpan, "These Google "Deep Dream" Images Are Weirdly Mesmerising," Wired, 03-Jul-2015 .

[5] A. Greco, G. Gallitto, M. D'Alessandro, and C. Rastelli, "Increased Entropic Brain Dynamics during DeepDream-Induced Altered Perceptual Phenomenology," Entropy, vol. 23, no. 7, pp. 839, Jul. 2021. doi: 10.3390/e23070839.

[6] C. Rastelli, A. Greco, Y. Kennett, C. Finocchiaro, and N. De Pisapia, "Simulated visual hallucinations in virtual reality enhance cognitive flexibility," Sci Rep, vol. 12, no. 1, pp. 4027, Mar. 2022. doi: 10.1038/s41598-022-08047-w.

[7] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going Deeper into Neural Networks," 2015 .

[8] B. Hayes, "Computer Vision and Computer Hallucinations," American Scientist, vol. 103, no. 6, pp. 380, Nov. 2015. doi: 10.1511/2015.117.380.

[9] https://slate.com/technology/2015/07/google-deepdream-its-dazzling-creepy-and-tells-us-a-lot-about-the-future-of-a-i.html

[10] "General Python FAQ — Python 3.9.2 documentation". docs.python.org. Archived from the original on 24 October 2012. Retrieved 28 March 2021.

[11] "Python 0.9.1 part 01/21". alt.sources archives. Archived from the original on 11 August 2021. Retrieved 11 August 2021.

[12] Vu, Linda (June 14, 2021). "Project Jupyter: A Computer Code that Transformed Science". Berkeley Lab Computing Sciences. Retrieved August 15, 2022.

[13] J. M. Perkel, "Why Jupyter is data scientists' computational notebook of choice," Nature, vol. 563, no. 7729, pp. 145-146, Oct. 2018. doi: 10.1038/d41586-018-07196-1.

[14] J. M. Perkel, "Ten computer codes that transformed science," Nature, vol. 589, no. 7841, pp. 344-348, Jan. 2021. doi: 10.1038/d41586-021-00075-2.

[15] "Visual Studio Code Display Language (Locale)". code.visualstudio.com. Microsoft. Retrieved 2021-03-19.

[16] T. Claburn, (12 September 2022). "PyTorch gets lit under The Linux Foundation". The Register.

[17] Patel, Mo (2017-12-07). "When two trends fuse: PyTorch and recommender systems". O'Reilly Media. Retrieved 2017-12-18.

[18] G. Bradski; A. Kaehler (2008). Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media, Inc. p. 6.

[19] Whitaker, Jeffrey. "The Matplotlib Basemap Toolkit User's Guide (v. 1.0.5)". Matplotlib Basemap Toolkit documentation. Retrieved 24 April 2013.

[20] "Matplotlib: Python plotting — Matplotlib 3.2.0 documentation". matplotlib.org. Retrieved 2020-03-14.

[21] O. Russakovsky, J. Deng, Su, H., et al. “ImageNet Large Scale Visual Recognition Challenge.” International Journal of Computer Vision (IJCV). Vol 115, Issue 3, 2015, pp. 211–252

[22] M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," in Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16), 2016, pp. 265-283. doi: 10.5555/3026877.3026899.

[23] H. Kaiming; Z. Xiangyu; R. Shaoqing; S. Jian (2015). *Deep Residual Learning for Image Recognition*

[24] "Machine learning - Is there a rule-of-thumb for how to divide a dataset into training and validation sets?". *Stack Overflow*. Retrieved 2021-08-12.

[25] James, Gareth (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer. p. 176. ISBN 978-1461471370.

[26] Gatys, Leon A.; Ecker, Alexander S.; Bethge, Matthias (26 August 2015). "A Neural Algorithm of Artistic Style". arXiv:1508.06576

[27] J. Justin; A. Alexandre; L. Fei-Fei (2016). "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". arXiv:1603.08155

[28] https://www.tensorflow.org/hub/tutorials/tf2_arbitrary_image_stylization

[29] C. Lemaréchal (2012). "Cauchy and the Gradient Method" (PDF). *Doc Math Extra*: 251–254.

[30] R. Courant, (1943). "Variational methods for the solution of problems of equilibrium and vibrations". *Bulletin of the American Mathematical Society*. **49** (1): 1–23. doi:10.1090/S0002-9904-1943-07818-4

[31] https://in.mathworks.com/help/deeplearning/ref/vgg19.html

ORIGINALITY REPORT

18%	7%	10%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to SASTRA University Student Paper	1%
2	Submitted to Babes-Bolyai University Student Paper	1%
3	Submitted to Liverpool John Moores University Student Paper	1%
4	library.samdu.uz Internet Source	1%
5	Charu C. Aggarwal. "Neural Networks and Deep Learning", Springer Science and Business Media LLC, 2018 Publication	1%