A Project Report

on

Deepfake in Picture using Autoencoder

Submitted in partial fulfillment of the

requirement for the award of the degree

of

Bachelor of Technology in Computer Science and Engineering

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Mr. Padmanabhan P.
Assistant Professor

Department of Computer Science and Engineering

Submitted By

20SCSE1010776- ANIKET GUPTA

20SCSE1010777- RISHABH CHAUDHARY

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GALGOTIAS UNIVERSITY, GREATER NOIDA, INDIA DECEMBER - 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled **"Deepfake in Picture using Autoencoder"** in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater Noida, is an original work carried out during the period of **JULY-2021 to DECEMBER-2021**, under the supervision of **Mr. Padmanabhan P., Assistant Professor**, **Department of Computer Science and Engineering** of School of Computing Science and Engineering, Galgotias University, Greater Noida

The matter presented in the project has not been submitted by me/us for the award of any other degree of this or any other places.

20SCSE1010776- ANIKET GUPTA

20SCSE1010777 – RISHABH CHAUDHARY

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

Supervisor

(**Mr. Padmanabhan P**, Assistant

Professor)

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of **20SCSE1010776 – ANIKET GUPTA, 20SCSE1010777 – RISHABH CHAUDHARY** has been held on ________and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING**.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date:

Place:

ABSTRACT

Generative neural networks are usually the most essential part of deepfake, a technique of image-to-image translation, designed to combine and overlay objects in images or videos creating deceptively realistic counterfeits. In this paper, four leading methods used for deepfake generation: autoencoders, variational autoencoders, variational autoencoders generative adversarial networks and cycle generative adversarial networks, in problem of face-to-face conversion, are analyses. We present results of experiments conducted on face-swapping task, performed on specially preprocessed data from VoxCeleb2 dataset. Due to the lack of numerical methods for deepfake comparison, a descriptive assessment method was proposed, and all obtained results were rated in a visual evaluation process. General conclusions concerning applicability of considered approaches to deepfake generation problem were formulated.

Table of Contents

List of Figures

List of Tables

Acronyms

SVM	Support Vector Maching
ML	Machine Learning
DL	Deep Learning
CNN	Convolution Neural Networks

CHAPTER-1

Introduction

Machine learning has found many different applications in the field of image data processing and computer vision. From picture classification to image denoising and resolution enhancement, deep neural networks has gained the opinion of exceptionally useful tools. But for some time, a new, controversial use-case has been getting more and more attention: so-called **deepfake technology** has opened doors to many new possibilities of picture generation, but also raised many issues of moral and legal matters.

Deepfake is a machine learning technology designed to combine and overlay objects in images or videos, creating deceptively realistic counterfeits. It is usually based on generative neural networks, such as various kinds of autoen- coders (AE), generative adversarial networks (GAN), or CycleGAN networks. Deepfake videos, almost impossible to distinguish with the naked eye, are created for entertainment purposes on many YouTube channels as for example “Ctrl Shift Face”. Many of them are created with one of the most advanced ready to use, open-source tool for deepfake generation “DeepFaceLab” which allows generating exceptionally believable deepfakes without having to acquire extensive knowledge in the field of machine learning. Another, very effective, open-source tool for creating high-quality deepfakes is “FaceSwap” project. Although there are many malicious ways of using deepfake technology, it might also be used for good reasons. Besides, to be able to detect such artificial image modifications it might be vital to understand algorithms and techniques behind it, especially in case of various possible applications of this technology.In this paper, we present results of experiments conducted on basic building blocks of deepfake techniques, namely generative neural networks, used for swapping face images. Four main approaches were examined and compared on a task of human faces replacement. In additional to regular autoencoders, also variational autoencoders (VAE), variational autoencoder generative adversarial networks (VAE-GAN) and CycleGAN were considered. It is, to our knowledge, the first study aiming at comparing these methods of generating deepfakes.

Deepfakes are AI-generated synthetic videos of any person or celebrity that impersonates the actual person and makes them act or say anything they originally never did. The process of creation of deepfakes is technically complex and generally requires a vast amount of data which is then fed to a neural network to train and generate the synthetic video.

Deepfake

Deepfakes are synthetic media in which **a person in an existing image or video is replaced with someone else's likeness.** The act of injecting a fake person in an image is not new. However, recent Deepfakes methods usually leverage the recent advancements of powerful models, aiming at facial manipulation.

In general, facial manipulation is usually conducted with Deepfakes and can be categorized in the following categories:

- **Face synthesis**
- **Face swap**
- **Facial attributes and expression**

Face synthesis

In this category, the objective is to create non-existent realistic faces using GANs. The most popular approach is StyleGAN. Briefly, a new generator architecture learns separation of high-level attributes (e.g., pose and identity when trained on human faces) without supervision and stochastic variation in the generated images (e.g., freckles, hair), and it enables intuitive, scale-specific control of the synthesis. The StyleGAN's generator is shown in Figure 2.

The input is mapped through several fully connected layers to an intermediate representation **w** which is then fed to each convolutional layer through adaptive instance normalization (AdaIN), where each feature map is normalized separately. Gaussian noise is added after each convolution. The benefit of adding noise directly in the feature maps of each layer is that global aspects such as identity and pose are unaffected.

The StyleGAN generator architecture makes it possible to control the image synthesis via scale-specific modifications to the styles. The mapping network and affine transformations are a way to draw samples for each style from a learned distribution, and the synthesis network is a way to generate an image based on a collection of styles. The effects of each style are localized in the network**,** i.e., modifying a specific subset of the styles can be expected to affect only certain aspects of the image. The reason for this localization, is based on the AdaIN operation that first normalizes each channel to zero mean and unit variance, and only then applies scales and biases based on the style. The new per-channel statistics, as dictated by the style, modify the relative importance of features for the subsequent convolution operation, but they do not depend on the original statistics because of the normalization. Thus each style controls only one convolution before being overridden by the next AdaIN operation.

Face swap

Face swap is the most popular face manipulation category nowadays. The aim here is to detect whether an image or video of a person is fake after swapping its face. The most popular database with fake and real videos is Face Forensics++. The fake videos in this dataset were made using computer graphics (Face Swap) and deep learning methods (Deepfake Face Swap). The Face Swap app is written in Python and uses face alignment, Gauss-Newton optimization, and image blending to swap the face of a person seen by the camera with a face of a person in a provided image. (For further details check the official repo)

The Deepfake Face Swap approach is based on two autoencoders with a shared encoder that are trained to reconstruct training images of the source and the target face, respectively.

A face in a target sequence is replaced by a face that has been observed in a source video or image collection. A face detector is used to crop and to align the images. To create a fake image, the trained encoder and decoder of the source face are applied to the target face. The autoencoder output is then blended with the rest of the image using Poisson image editing.

- a CNN-based system trained through handcrafted features
- a CNN-based system with convolution layers that try to suppress the high-level content of the image

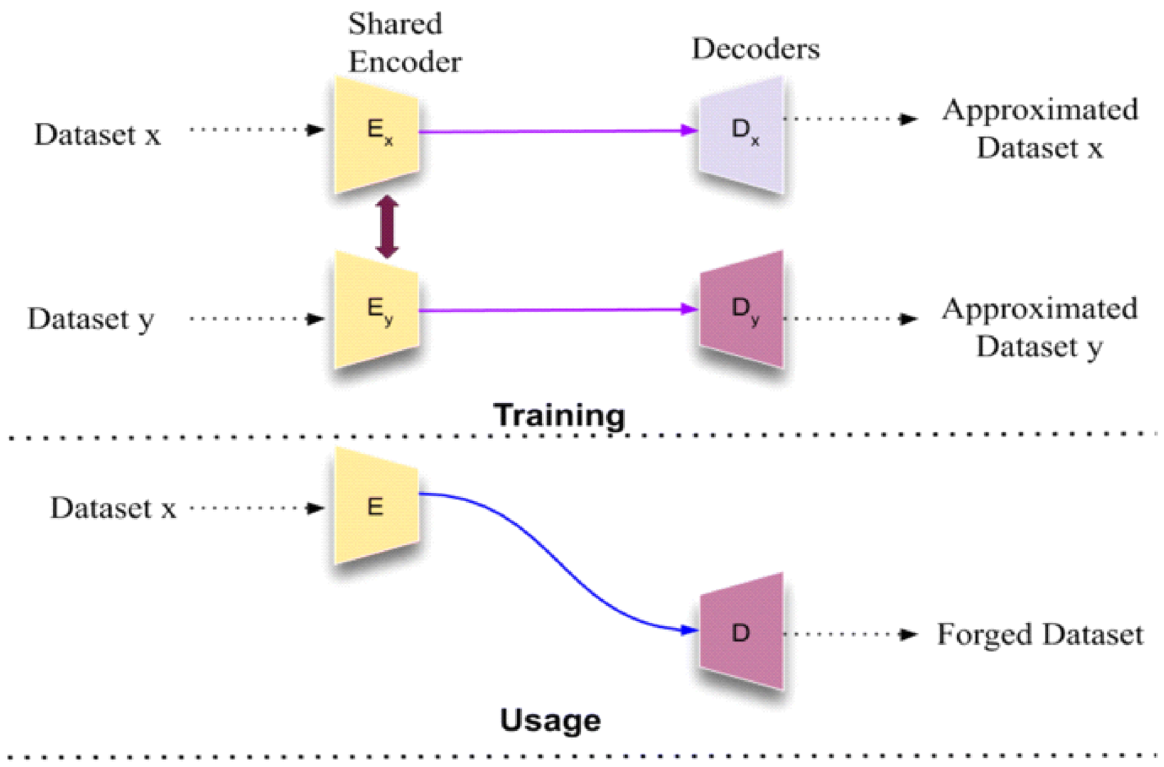
- a CNN-based system with a global pooling layer that computes four statistics (mean, variance, maximum, and minimum)
- the CNN MesoInception-4 detection system
- the CNN-based system XceptionNet pre-trained using ImageNet dataset and trained again for the face swap task. XceptionNet is a CNN architecture inspired from Inception and uses depth-wise separable convolutions.

Project Design

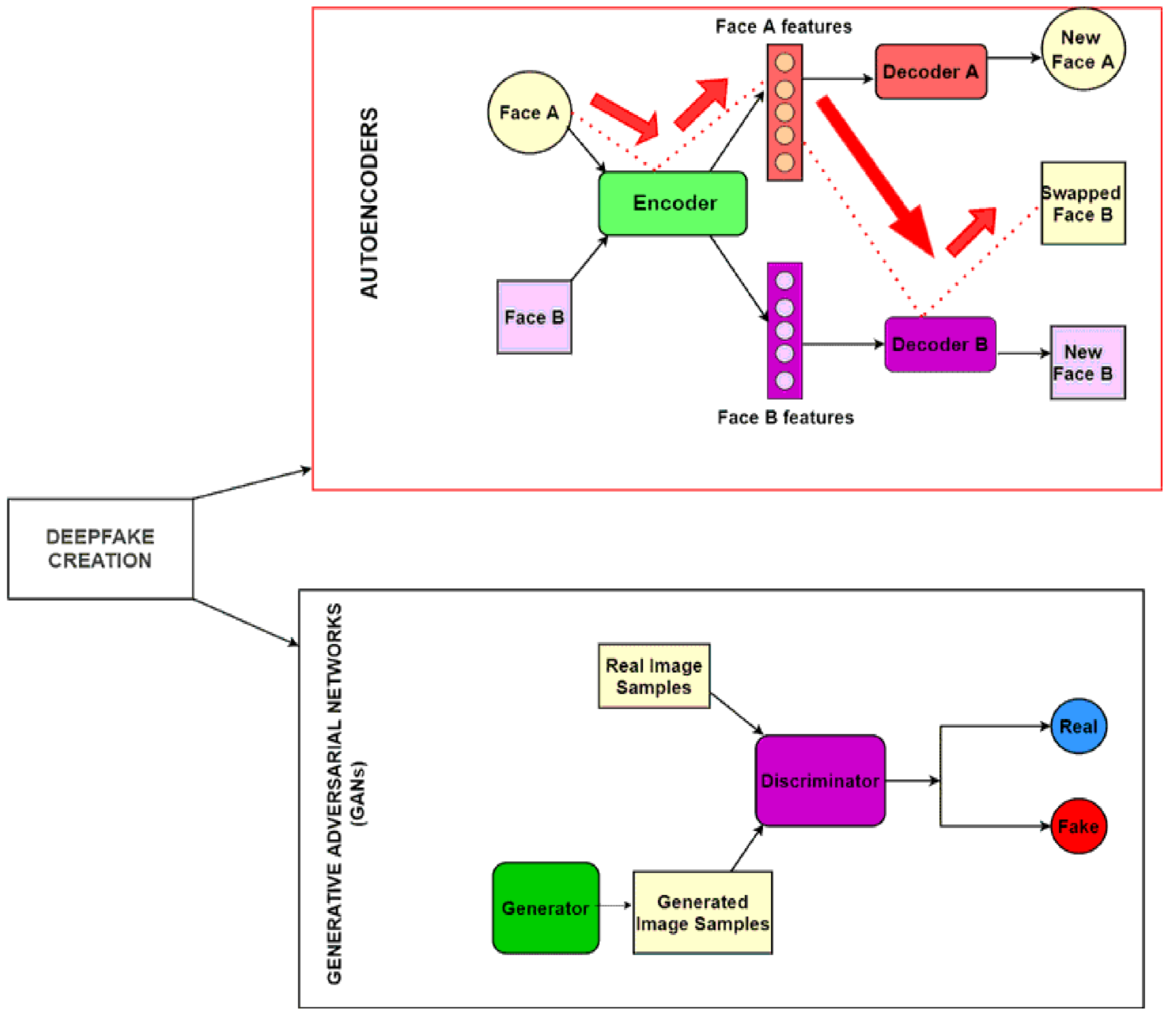
State of the art

Although the deepfake idea is relatively new, a few impressively effective methods has been already developed. The most frequently chosen approaches, in case of generative neural networks are autoencoders, variational au- to encoders, generative adversarial networks and so called CycleGAN networks. They are not complete, standalone methods for deepfake generation but rather key-parts of deepfake generation schemes.

Arrangement of Dart Files and Packages



Architectural Layers of Flutter

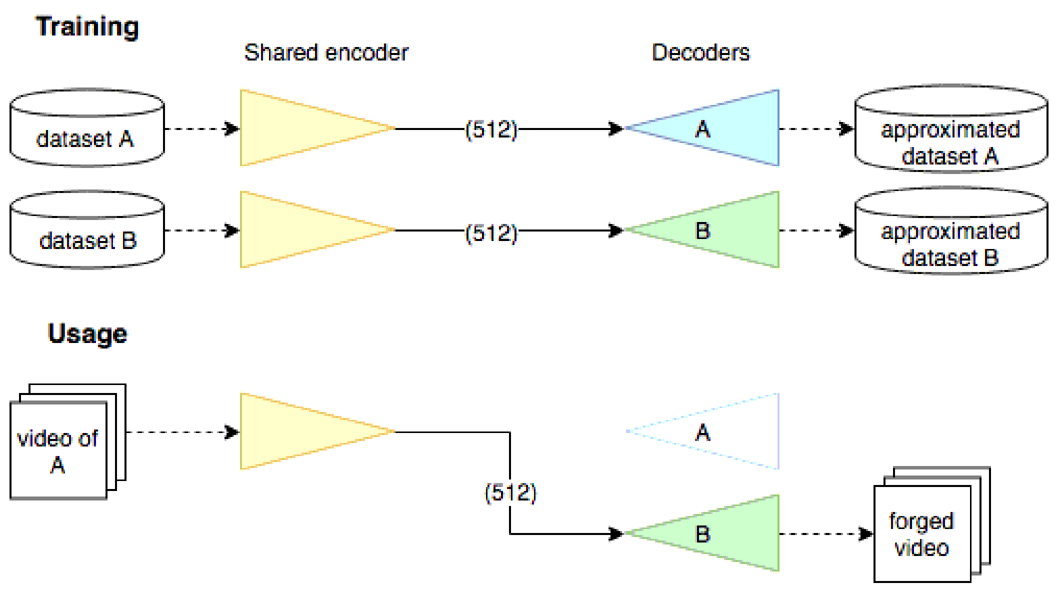


Data preprocessing

To prepare dataset best suited for purpose of this study, following preparations were made:

- Two actors (Leonardo DiCaprio and Robert Downey Jr.), further called subject X and subject Y, were chosen as targets of face replacement. This choice was made based on how well their faces are known and recognizable, which facilitates the final assessment.
- From the set of all videos of chosen subjects available in “VoxCeleb2” dataset, those which presented subjects in similar age, and had good recording quality, were selected.
- From each video, every tenth frame was extracted, to limit the amount of nearly identical images. Additionally, by Haar feature-based cascade classifiers, the face itself was cut out from each extracted frame to discard unnecessary parts of pictures.
- Resolutions of all obtained face images varied, therefore data had to be rescaled to a common resolution of 160 by 160 pixels.

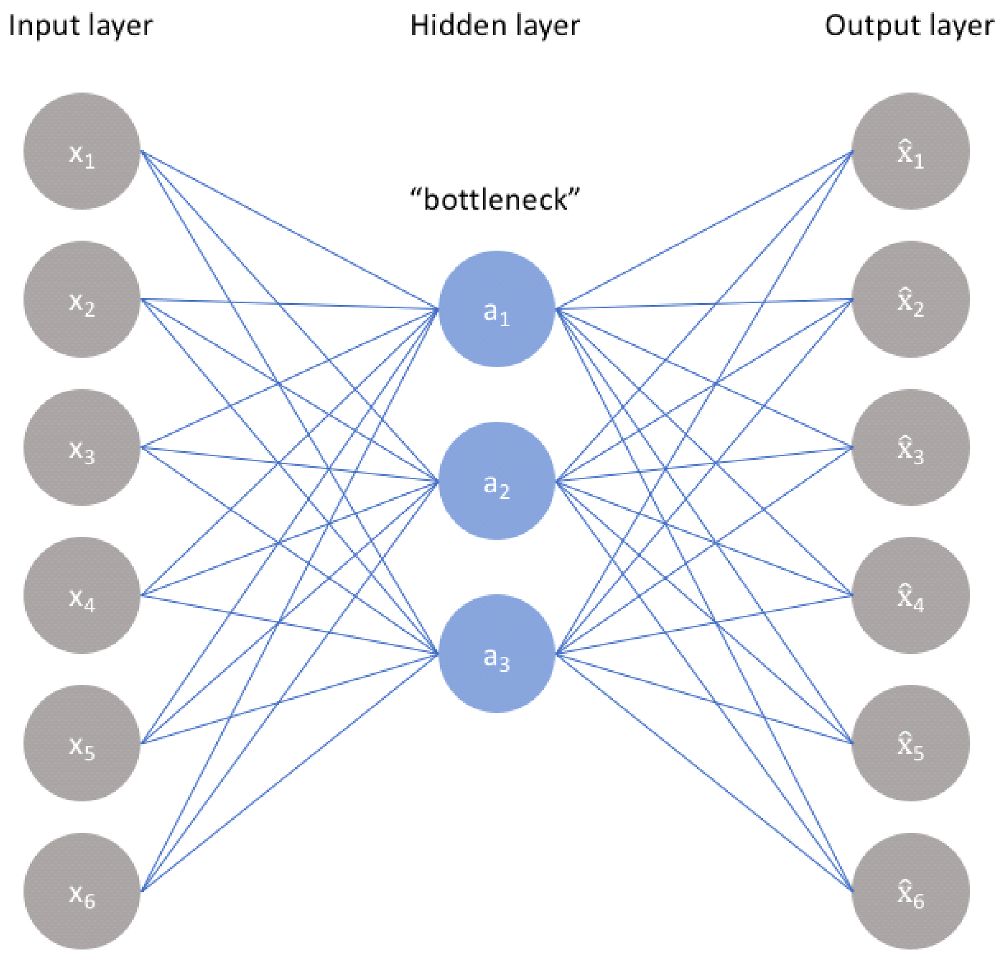
Class Diagram



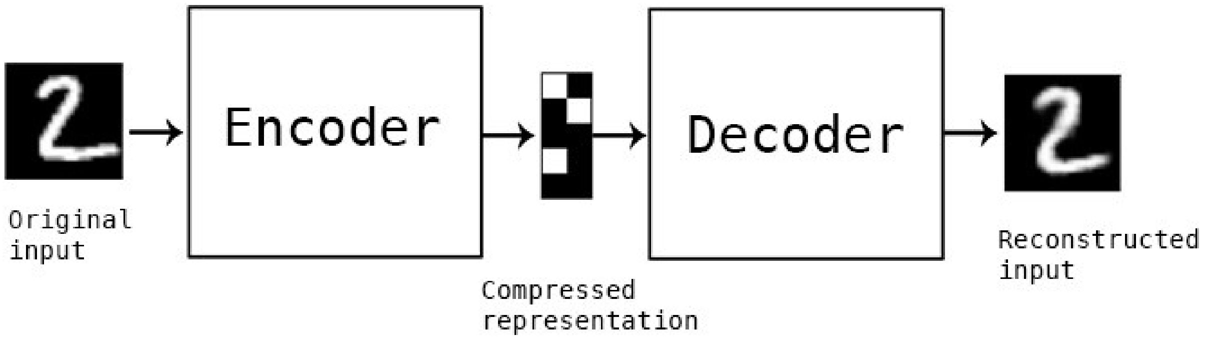
Autoencoder

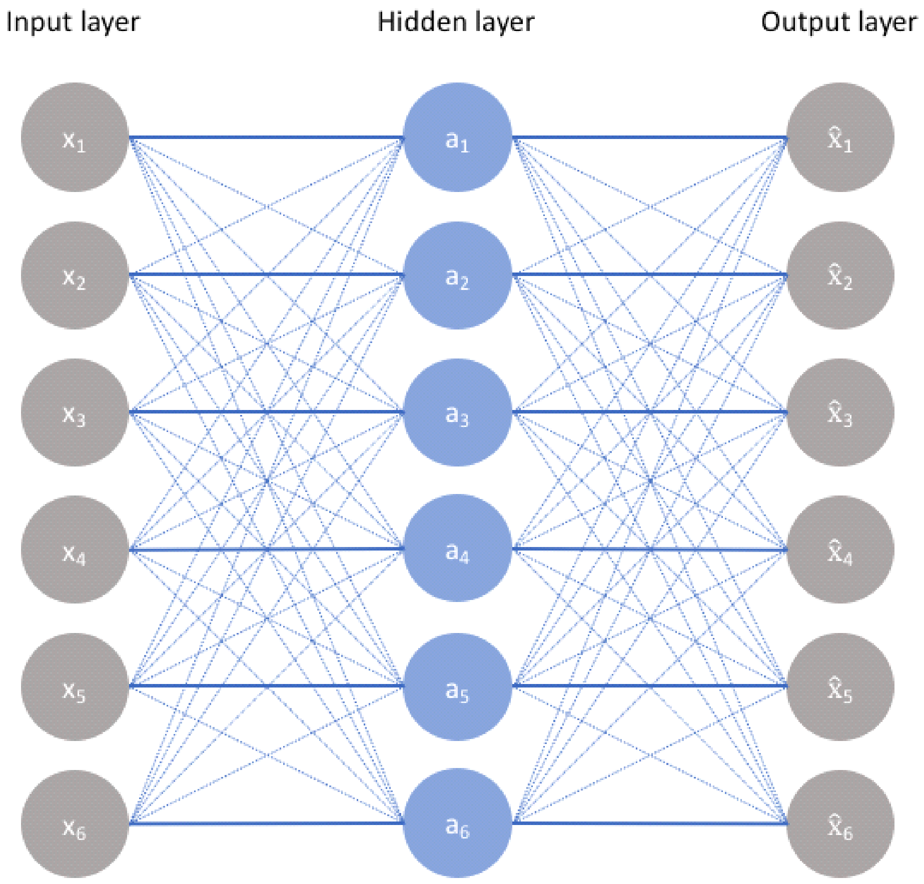
Autoencoder is an unsupervised artificial neural network that learns how to efficiently compress and encode data then learns how to reconstruct the data back **from** the reduced encoded representation **to** a representation that is as close to the original input as possible. Autoencoder, by design, reduces data dimensions by learning how to ignore the noise in the data.

Autoencoders are an unsupervised learning technique in which we leverage neural networks for the task of **representation learning**. Specifically, we'll design a neural network architecture such that we *impose a bottleneck in the network which forces a* **compressed** *knowledge representation of the original input*. If the input features were each independent of one another, this compression and subsequent reconstruction would be a very difficult task. However, if some sort of structure exists in the data (i.e. correlations between input features), this structure can be learned and consequently leveraged when forcing the input through the network's bottleneck.

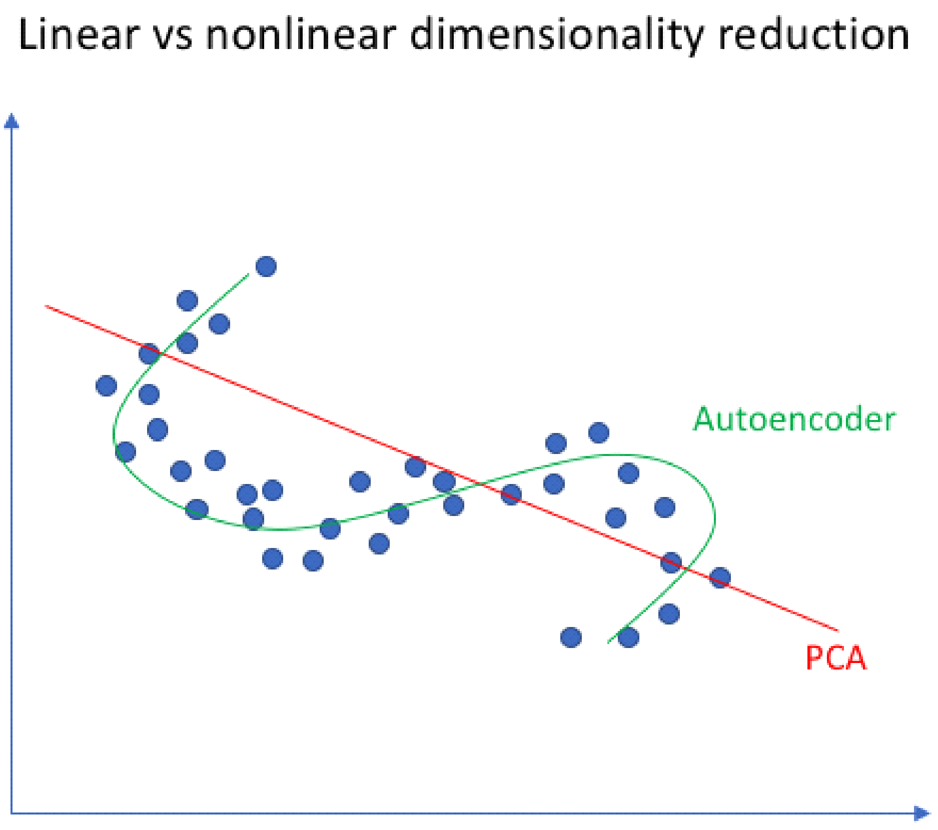


Here is an example of the input/output image from the dataset





Because neural networks are capable of learning nonlinear relationships, this can be thought of as a more powerful (nonlinear) generalization of PCA. Whereas PCA attempts to discover a lower dimensional hyperplane which describes the original data, autoencoders are capable of learning nonlinear manifolds (a manifold is defined in *simple* terms as a continuous, non-intersecting surface). The difference between these two approaches is visualized below.

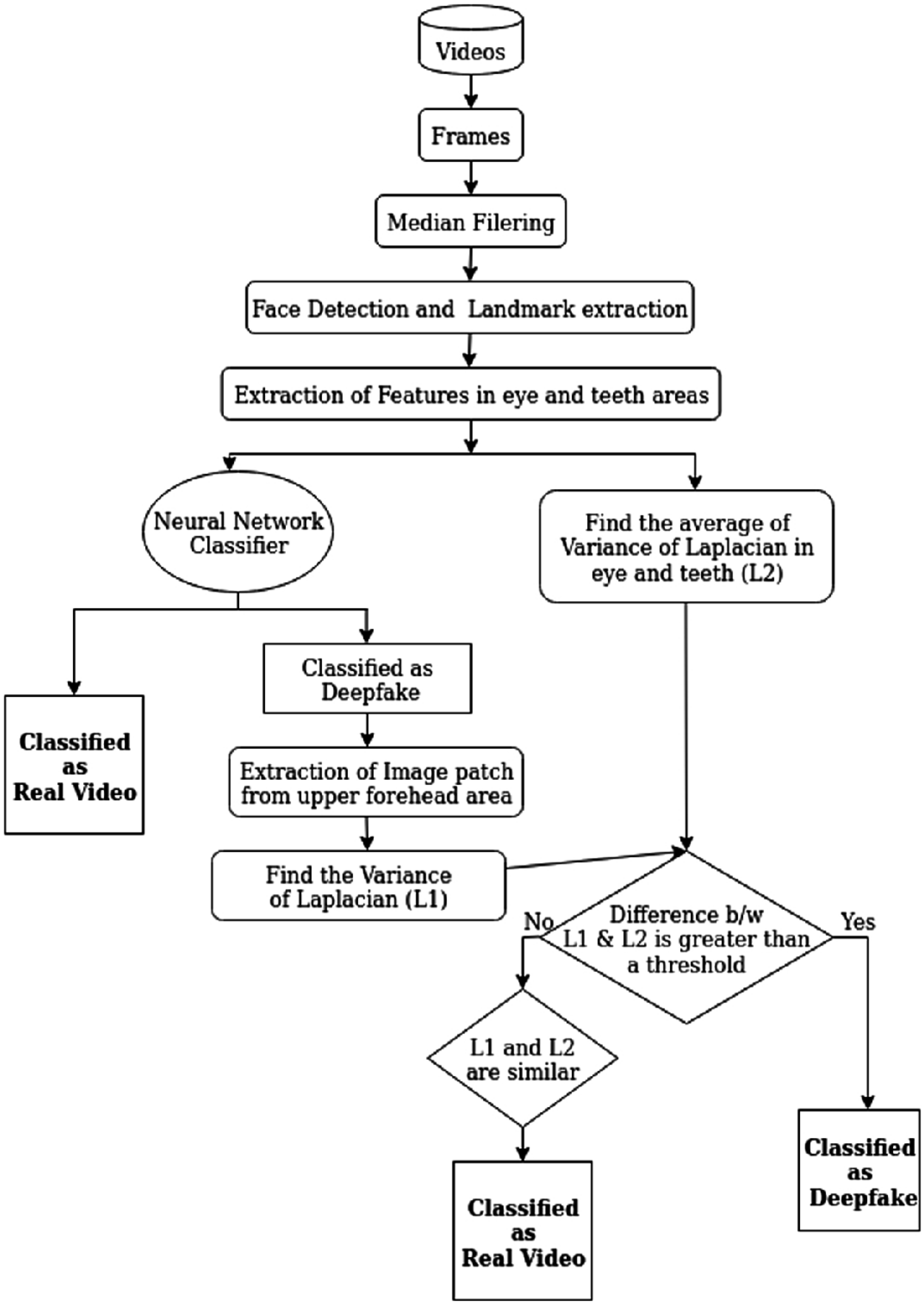


An undercomplete autoencoder has no explicit regularization term - we simply train our model according to the reconstruction loss. Thus, our only way to ensure that the model isn't memorizing the input data is the ensure that we've sufficiently restricted the number of nodes in the hidden layer(s).

For deep autoencoders, we must also be aware of the *capacity* of our encoder and decoder models.

Even if the "bottleneck layer" is only one hidden node, it's still possible for our model to memorize the training data provided that the encoder and decoder models have sufficient capability to learn some arbitrary function which can map the data to an index.
Given the fact that we'd like our model to discover latent attributes within our data, it's important to ensure that the autoencoder model is not simply learning an efficient way to memorize the training data. Similar to supervised learning problems, we can employ various forms of regularization to the network in order to encourage good generalization properties.

Sequence Diagram


Videos
Frames
Median Filering
Face Detection and Landmark extraction
Extraction of Features in eye and teeth areas
Neural Network Classifier
Find the average of Variance of Laplacian in eye and teeth (L2)
Classified as Real Video
Classified as Deepfake
Extraction of Image patch from upper forehead area
Find the Variance of Laplacian (L1)
No
Difference b/w L1 & L2 is greater than a threshold
Yes
L1 and L2 are similar
Classified as Deepfake
Classified as Real Video

Autoencoder Components:

Autoencoders consists of 4 main parts:

1- **Encoder**: In which the model learns how to reduce the input dimensions and compress the input data into an encoded representation.

2- **Bottleneck**: which is the layer that contains the compressed representation of the input data. This is the lowest possible dimensions of the input data.

3- **Decoder**: In which the model learns how to reconstruct the data from the encoded representation to be as close to the original input as possible.

4- **Reconstruction Loss**: This is the method that measures measure how well the decoder is performing and how close the output is to the original input.

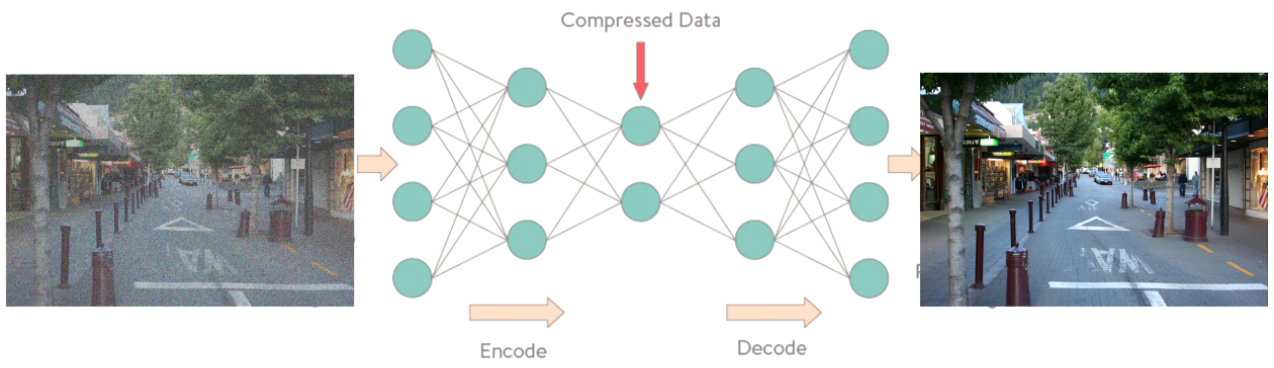
The training then involves using back propagation in order to minimize the network’s reconstruction loss.

Autoencoder Architecture

The network architecture for autoencoders can vary between a simple Feed Forward network, LSTM network or Convolutional Neural Network depending on the use case. We will explore some of those architectures in the new next few lines.

1.Autoencoder for Anomaly Detection

2.Image Denoising



Impact of Deepfakes

Deepfakes and AI avatars can have varying impacts depending on how it’s used. While the negative effects of deepfake can be scary and frightening to imagine, it also can be useful in other circumstances and use cases.

Pros of Deepfakes-

- Deepfakes can be used as a form of art to bring people from the past back to life. For example, a painting of the Mona Lisa can be used for generating a synthetic image of talking Mona Lisa as a form of art.

- Deepfake technology can be used to create AI avatars in training videos. Startups like London-based Synthesia have been getting more attention from the corporate world during the COVID pandemic since lockdowns and health concerns have made video shoots involving real people much more difficult to pull off.
- Deepfakes can be used to create personal avatars to try on clothes or new hairstyles before trying them in real.

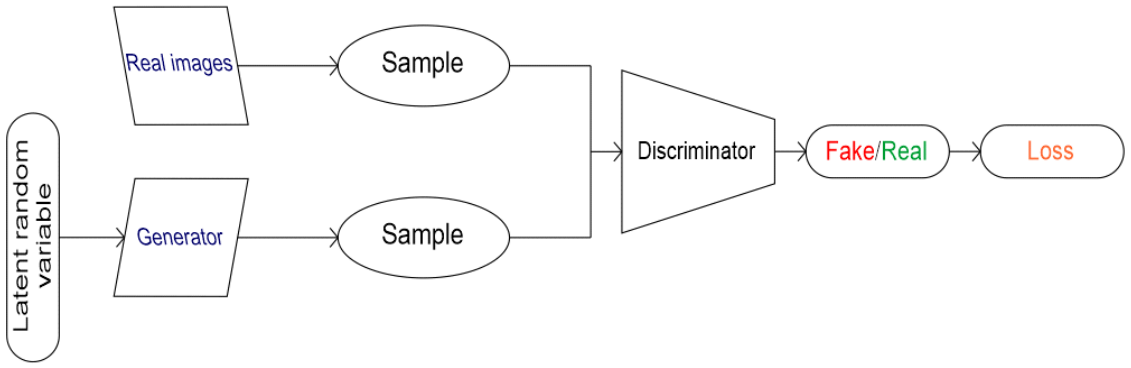
Deepfakes can also be used in identity protection

and anonymization in various fields like

investigative news reporting, finance, etc.

Cons of Deepfakes-

- Deepfakes can be used to spread fake news with morphed videos of celebrities.
- Deepfakes can also be misused for creating misinformation campaigns on social media that can shift public opinion and lead to negative consequences.


Real images
Sample
Latent random variable
Generator
Sample
Discriminator
Fake/Real
Loss

Working of Project

Creating Deepfakes

While deepfakes can be used or misused in multiple ways, creating them is becoming easier with more advancements in AI with every passing day.

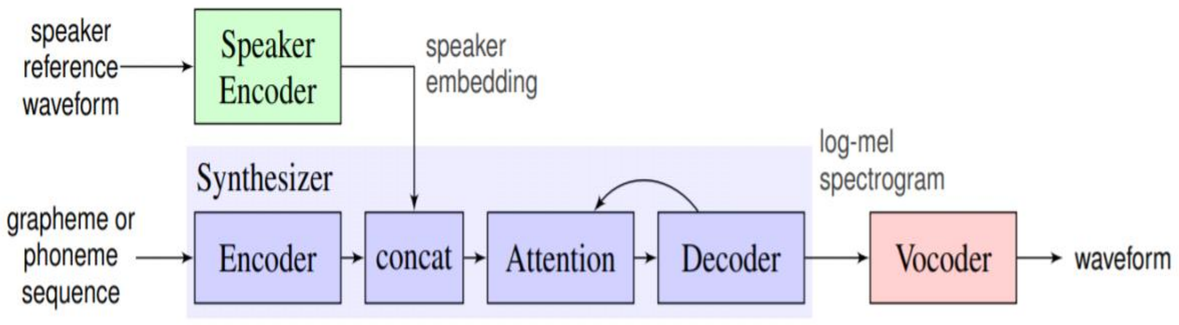
We can now create a deepfake with just 1 small source video of the person. Yes, that is now easily possible with the latest advancements in neural networks. Read to know more!

Let’s break down the solution into two parts –

- Voice Cloning
- Video Lip Syncing

Voice Cloning part of Deepfakes

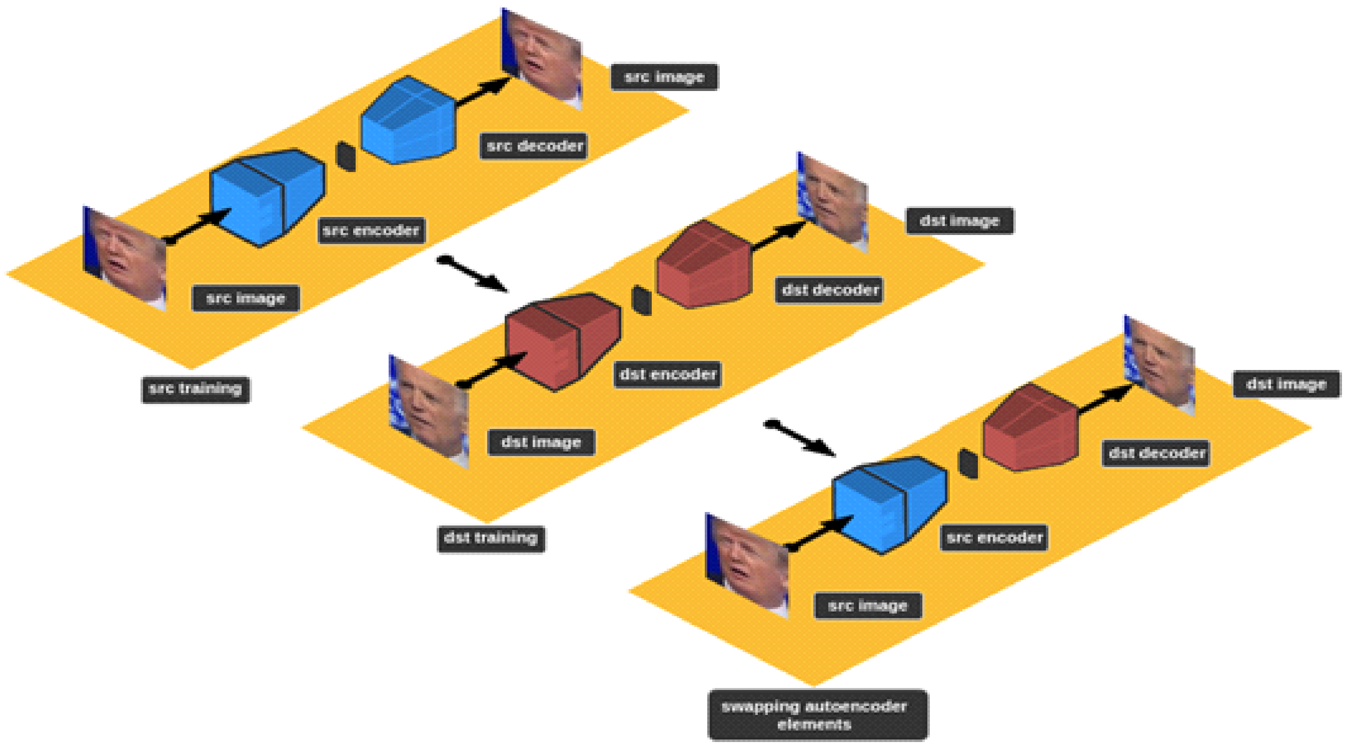
SV2TTS is a framework for deep learning that can be trained to quantify and represent audio as numbers and parameters based on only a small few second of audio of the voice of a person. This numeric depiction of the voice sample can be used to guide and train a text-to-speech model to generate new audio with the exact same voice with any text data as input. Thus, Using the extracted audio from the sample source video, a voice clone can be easily created with SV2TTS.



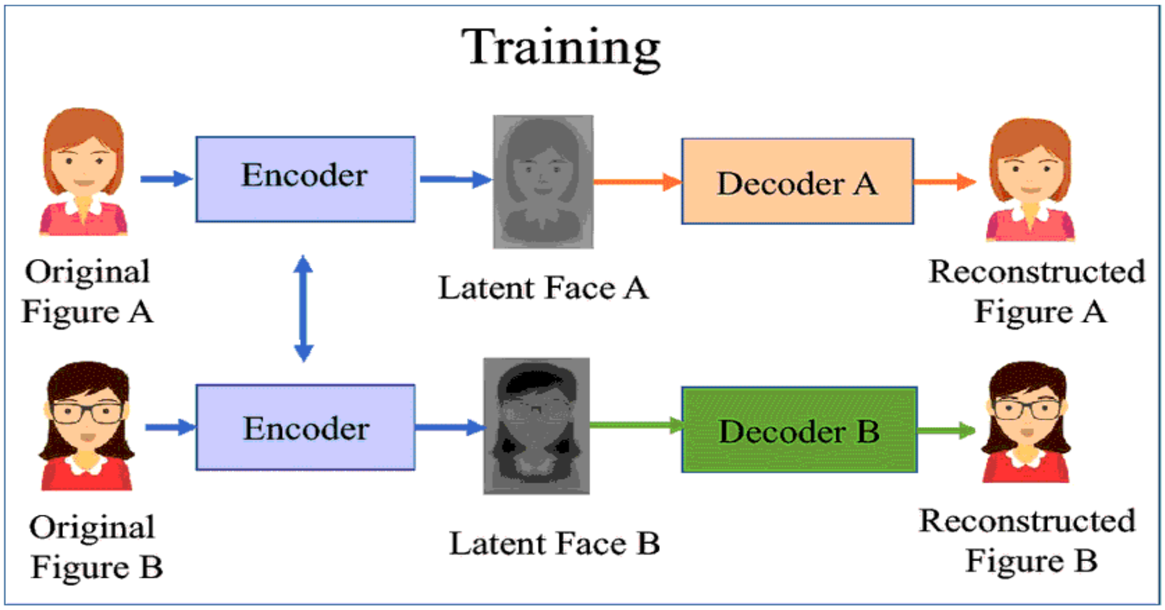
Example of Deepfake

- Deepfake video has also been used in *politics*. In 2018, for example, a Belgian political party released a *video of Donald Trump* giving a speech calling on *Belgium to withdraw from the Paris climate agreement.* Trump never gave that speech, however - it was a deepfake.
- That was not the first use of a deepfake to *create misleading videos*, and tech-savvy political

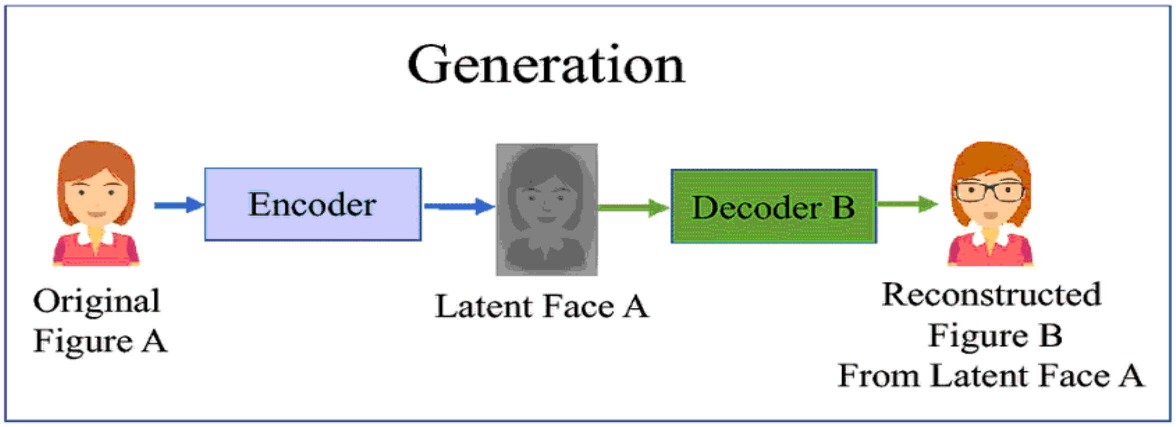
experts are bracing for a future wave of fake news that features convincingly realistic deepfakes.



Flow Diagram of Deepfake in Picture using Autoencoder

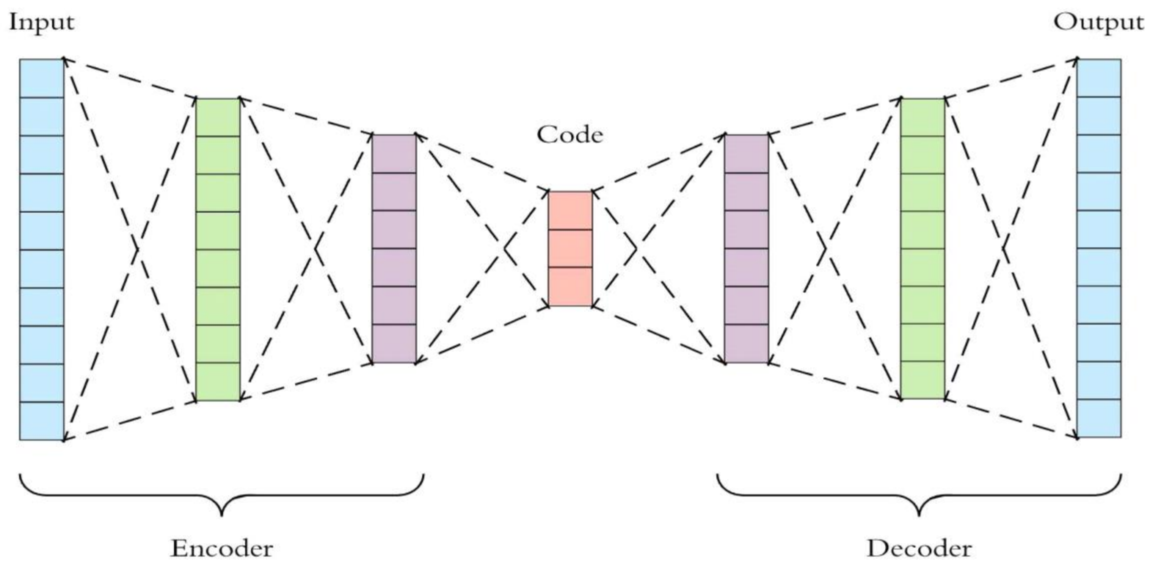


(a) Training Phase



(b) Generation Phase

Technical Diagram



Advantages of Deepfake

- In Art:

Star Wars fans are likely aware of the use of deepfake tech to bring the actor Peter Cushing back to “life” for 2016’s Rogue One.

- Professional training:

Deepfake technology can be used to create AI avatars for use in training videos.

- Identity protection:

AI-generated avatars have been used to protect the identity of interviewees in news reports.

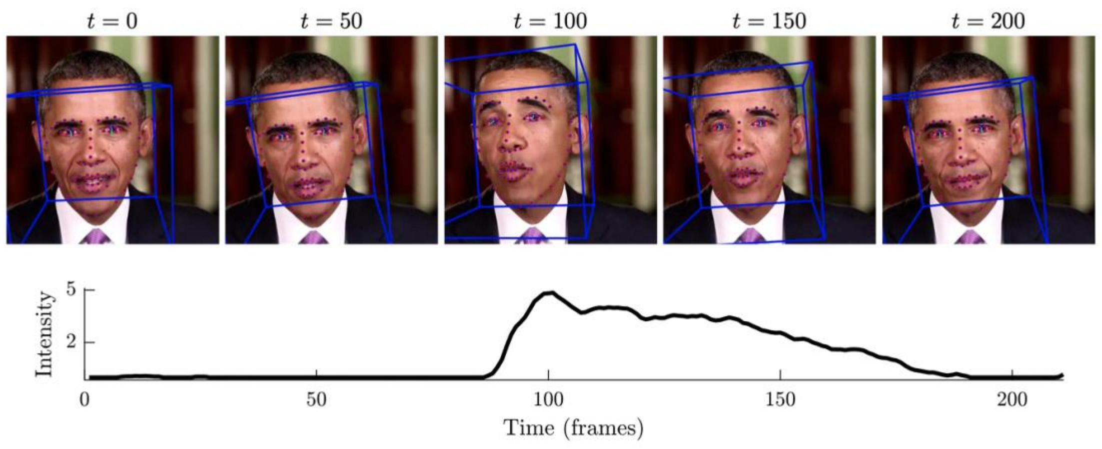
Disadvantages of Deepfake

- Scamming :

Another area of concern is financial scams. Audio deepfakes have already been used to clone voices and convince people they are talking to someone trusted and defraud them. Earlier this year, scammers used a deepfake of a tech CEOs voice to try and convince an employee at the company to transfer money to the scammer’s account.

And this is not the first time: last year, scammers using the exact same trick managed to defraud a company out of $240,000.

- Damage reputations.
- Fabricate evidence.
- Defraud the public.
- Undermine trust in democratic institutions.

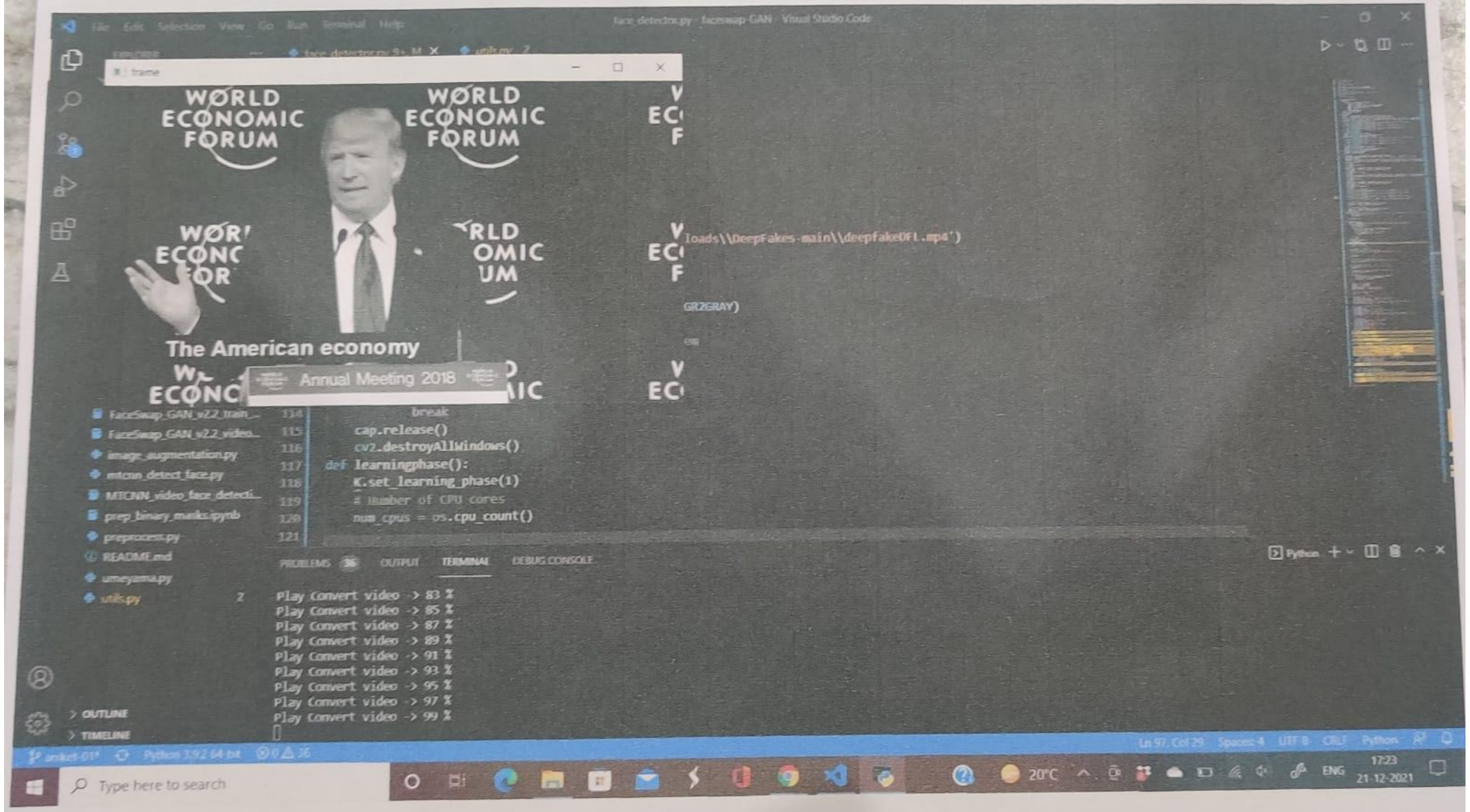


SOURCE CODE AND OUTPUT

```
import  numpy  as  np

import  pandas  as  pd

import os

import matplotlib.pyplot as plt

import cv2

from tensorflow.keras.models import load_model

from sklearn.model_selection import train_test_split

import keras

from keras import layers
from tensorflow.keras.callbacks import ModelCheckpoint
import tensorflow as tf
import time
from PIL import Image

def create_dataset(path):
    images = []
    for dirname, _, filenames in os.walk(path):
        for filename in filenames:
            image = cv2.imread(os.path.join(dirname, filename))
            image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
            image = image.astype('float32')
            image /= 255.0
            images.append(image)
    images = np.array(images)
    return images
def deepfake():
    faces_1 = create_dataset('/kaggle/input/presidentsfacesdataset/trump/')
    faces_2 = create_dataset('/kaggle/input/presidentsfacesdataset/biden/')
    X_train_a, X_test_a, y_train_a, y_test_a = train_test_split(faces_1, faces_1, test_size=0.20,
random_state=0)
    X_train_b, X_test_b, y_train_b, y_test_b = train_test_split(faces_2, faces_2, test_size=0.15,
random_state=0)

    input_img = layers.Input(shape=(120, 120, 3))
    x = layers.Conv2D(256,kernel_size=5, strides=2, padding='same',activation='relu')(input_img)x
    = layers.MaxPooling2D((2, 2), padding='same')(x)
    x = layers.Conv2D(512,kernel_size=5, strides=2, padding='same',activation='relu')(x)x
    = layers.MaxPooling2D((2, 2), padding='same')(x)
    x = layers.Conv2D(1024,kernel_size=5, strides=2, padding='same',activation='relu')(x)x
    = layers.MaxPooling2D((2, 2), padding='same')(x)
    x = layers.Flatten()(x)
    x = layers.Dense(9216)(x)
    encoded = layers.Reshape((3,3,1024))(x)
    encoder = keras.Model(input_img, encoded,name="encoder")
    decoder_input= layers.Input(shape=((3,3,1024)))
    x = layers.Conv2D(1024,kernel_size=5, strides=2, padding='same',activation='relu')(decoder_input)x
    = layers.UpSampling2D((2, 2))(x)
    x = layers.Conv2D(512,kernel_size=5, strides=2, padding='same',activation='relu')(x)x
    = layers.UpSampling2D((2, 2))(x)
    x = layers.Conv2D(256,kernel_size=5, strides=2, padding='same',activation='relu')(x)x
    = layers.Flatten()(x)
    x = layers.Dense(np.prod((120, 120, 3)))(x)
    decoded = layers.Reshape((120, 120, 3))(x)
    decoder = keras.Model(decoder_input, decoded,name="decoder")
    auto_input = layers.Input(shape=(120,120,3))
    encoded = encoder(auto_input)
    decoded = decoder(encoded)

    autoencoder = keras.Model(auto_input, decoded,name="autoencoder")
    autoencoder.compile(optimizer=keras.optimizers.Adam(lr=5e-5, beta_1=0.5, beta_2=0.999),
loss='mae')
    autoencoder.summary()
    checkpoint1 = ModelCheckpoint("/kaggle/working/autoencoder_a.hdf5", monitor='val_loss',
verbose=1,save_best_only=True, mode='auto', period=1)

    history1 = autoencoder.fit(X_train_a, X_train_a, epochs=2700, batch_size=512, shuffle=True,
validation_data=(X_test_a, X_test_a), callbacks=[checkpoint1])
```

```
    # %matplotlib inline

    plt.figure()

    plt.imshow(X_test_a[30])

    plt.show()

    autoencoder_a = load_model("/kaggle/working/autoencoder_a.hdf5")

    output_image = autoencoder_a.predict(np.array([X_test_a[30]]))

    plt.figure()

    plt.imshow(output_image[0])

    plt.show()

    checkpoint2 = ModelCheckpoint("/kaggle/working/autoencoder_b.hdf5", monitor='val_loss',
verbose=1,save_best_only=True, mode='auto', period=1)

    history2 = autoencoder.fit(X_train_b,
X_train_b,epochs=2700,batch_size=512,shuffle=True,validation_data=(X_test_b,
X_test_b),callbacks=[checkpoint2])

    plt.figure()

    plt.imshow(X_test_b[0])

    plt.show()

    autoencoder_b = load_model("/kaggle/working/autoencoder_b.hdf5")

    output_image = autoencoder_b.predict(np.array([X_test_b[0]]))

    plt.figure()

    plt.imshow(output_image[0])

    plt.show()

    # TO LOAD ONLY THE ENCODER A

    encoder_a = keras.Model(autoencoder_a.layers[1].input, autoencoder_a.layers[1].output)#

    TO LOAD ONLY THE DECODER A

    decoder_a = keras.Model(autoencoder_a.layers[2].input, autoencoder_a.layers[2].output)#

    TO LOAD ONLY THE ENCODER B

    encoder_b = keras.Model(autoencoder_b.layers[1].input, autoencoder_b.layers[1].output)#

    TO LOAD ONLY THE DECODER B

    decoder_b = keras.Model(autoencoder_b.layers[2].input, autoencoder_b.layers[2].output)
    # TO TRANSFORM SRC IMAGES
    input_test = encoder_a.predict(np.array([X_test_a[30]]))
    output_test = decoder_b.predict(input_test)

    # TO TRANSFORM DST IMAGES
    input_test = encoder_b.predict(np.array([X_test_b[30]]))
    output_test = decoder_a.predict(input_test)
def convert(input):
    for i in range(1, 101, 2):
        print(input, "->", i, "%")
        time.sleep(1)

    showResult(input)

def showResult(input):
    cap = cv2.VideoCapture('C:\\Users\\anike\\Downloads\\DeepFakes-main\\deepfakeDFL.mp4')
    while(cap.isOpened()):
        ret, frame = cap.read()if
        ret == True:
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            cv2.imshow('frame', gray)
            # & 0xFF is required for a 64-bit systemif
            cv2.waitKey(20) & 0xFF == ord('q'):
                break
        else:
            break
    cap.release()
    cv2.destroyAllWindows()def
learningphase():
    K.set_learning_phase(1)
    # Number of CPU cores
    num_cpus = os.cpu_count()

    # Input/Output resolution
    RESOLUTION = 64 # 64x64, 128x128, 256x256
    assert (RESOLUTION % 64) == 0, "RESOLUTION should be 64, 128, or 256."

    # Batch size
    batchSize = 4

    # Use motion blurs (data augmentation)
    # set True if training data contains images extracted from videos
    use_da_motion_blur = False

    # Use eye-aware training
    # require images generated from prep_binary_masks.ipynb
    use_bm_eyes = True

    # Probability of random color matching (data augmentation)
    prob_random_color_match = 0.5

    da_config = {
        "prob_random_color_match": prob_random_color_match,
        "use_da_motion_blur": use_da_motion_blur,
        "use_bm_eyes": use_bm_eyes
    }

    # Path to training images
    img_dirA = './faceA/rgb'
    img_dirB = './faceB/rgb'
    img_dirA_bm_eyes = "./faceA/binary_mask"
    img_dirB_bm_eyes = "./faceB/binary_mask"

    # Path to saved model weights
    models_dir = "./models"

    # Architecture configuration
    arch_config = {}
    arch_config['IMAGE_SHAPE'] = (RESOLUTION, RESOLUTION, 3)
    arch_config['use_self_attn'] = True
    arch_config['norm'] = "hybrid" # instancenorm, batchnorm, layernorm, groupnorm, none
    arch_config['model_capacity'] = "lite" # standard, lite

    # Loss function weights configuration
    loss_weights = {}
    loss_weights['w_D'] = 0.1 # Discriminator
    loss_weights['w_recon'] = 1. # L1 reconstruction loss
    loss_weights['w_edge'] = 0.1 # edge loss
    loss_weights['w_eyes'] = 30. # reconstruction and edge loss on eyes area
    loss_weights['w_pl'] = (0.01, 0.1, 0.3, 0.1) # perceptual loss (0.003, 0.03, 0.3, 0.3)

    # Init. loss config.
    loss_config = {}
```

```
    loss_config["gan_training"] = "mixup_LSGAN"
    loss_config['use_PL'] = False
    loss_config["PL_before_activ"] = True
    loss_config['use_mask_hinge_loss'] = False
    loss_config['m_mask'] = 0.
    loss_config['lr_factor'] = 1.
    loss_config['use_cyclic_loss'] = False
def trainig():
    train_A = glob.glob(img_dirA+"/*.*")

    train_B = glob.glob(img_dirB+"/*.*")
    train_AnB = train_A + train_B
    for config, value in loss_config.items():
        print(f"{config} = {value}")
        model = FaceswapGANModel(**arch_config)
        model.load_weights(path=save_path)
        #vggface = VGGFace(include_top=False, model='resnet50', input_shape=(224, 224, 3)) vggface
        = RESNET50(include_top=False, weights=None, input_shape=(224, 224, 3))
        vggface.load_weights("rcmalli_vggface_tf_notop_resnet50.h5")
        model.build_pl_model(vggface_model=vggface, before_activ=loss_config["PL_before_activ"])
        train_batchA = DataLoader(train_A, train_AnB, batchSize, img_dirA_bm_eyes,
                            RESOLUTION, num_cpus, K.get_session(), **da_config) train_batchB
        = DataLoader(train_B, train_AnB, batchSize, img_dirB_bm_eyes,
                            RESOLUTION, num_cpus, K.get_session(), **da_config)
        assert len(train_A), "No image found in " + str(img_dirA)
        assert len(train_B), "No image found in " + str(img_dirB)
        print ("Number of images in folder A: " + str(len(train_A)))
        print ("Number of images in folder B: " + str(len(train_B)))
    data_A = train_batchA.get_next_batch()

    data_B = train_batchB.get_next_batch()

    errDA, errDB = model.train_one_batch_D(data_A=data_A, data_B=data_B)

    errDA_sum +=errDA[0]

    errDB_sum +=errDB[0]


    # Train generators for one batch

    data_A = train_batchA.get_next_batch()

    data_B = train_batchB.get_next_batch()

    errGA, errGB = model.train_one_batch_G(data_A=data_A, data_B=data_B)

    errGA_sum += errGA[0]

    errGB_sum += errGB[0]

    for i, k in enumerate(['ttl', 'adv', 'recon', 'edge', 'pl']):

   errGAs[k] += errGA[i]errGBs[k]

               += errGB[i]

    gen_iterations+=1
convert("Play Convert video")
```



```
loads\\DeepFakes-main\\deepfakeDFL.mp4')
GR2GRAY)
            break
    cap.release()
    cv2.destroyAllWindows()
def learningphase():
    K.set_learning_phase(1)
    # Number of CPU cores
    num_cpus = os.cpu_count()
```

```
Play Convert video -> 83 %
Play Convert video -> 85 %
Play Convert video -> 87 %
Play Convert video -> 89 %
Play Convert video -> 91 %
Play Convert video -> 93 %
Play Convert video -> 95 %
Play Convert video -> 97 %
Play Convert video -> 99 %
```

Conclusion and Future Scope

Deepfake can be used to anonymize voice and faces to protect their privacy. Deepfakes may be used to create avatar experiences for individuals online for self-expression.

Open-source intelligence, i.e., information collected from public sources on the internet, can be used to verify the credibility of images, videos, and other sources. Using open-source techniques, it’s possible to establish the veracity of a piece of information by providing a context for it. Because with deepfakes, it’s impossible to tell if an image or video is real or not simply by looking at it. You have to go deeper. Intelligence that indicates whether or not a piece of information has been manipulated could include information about when an image or video was taken, where it was taken, and if it correlates to a specific event. Decoding a fake could also involve taking a closer look at the file information. If the EXIF data has been stripped or changed in any way, that is an indication that someone wants to hide something and, therefore, that the file might be manipulated.

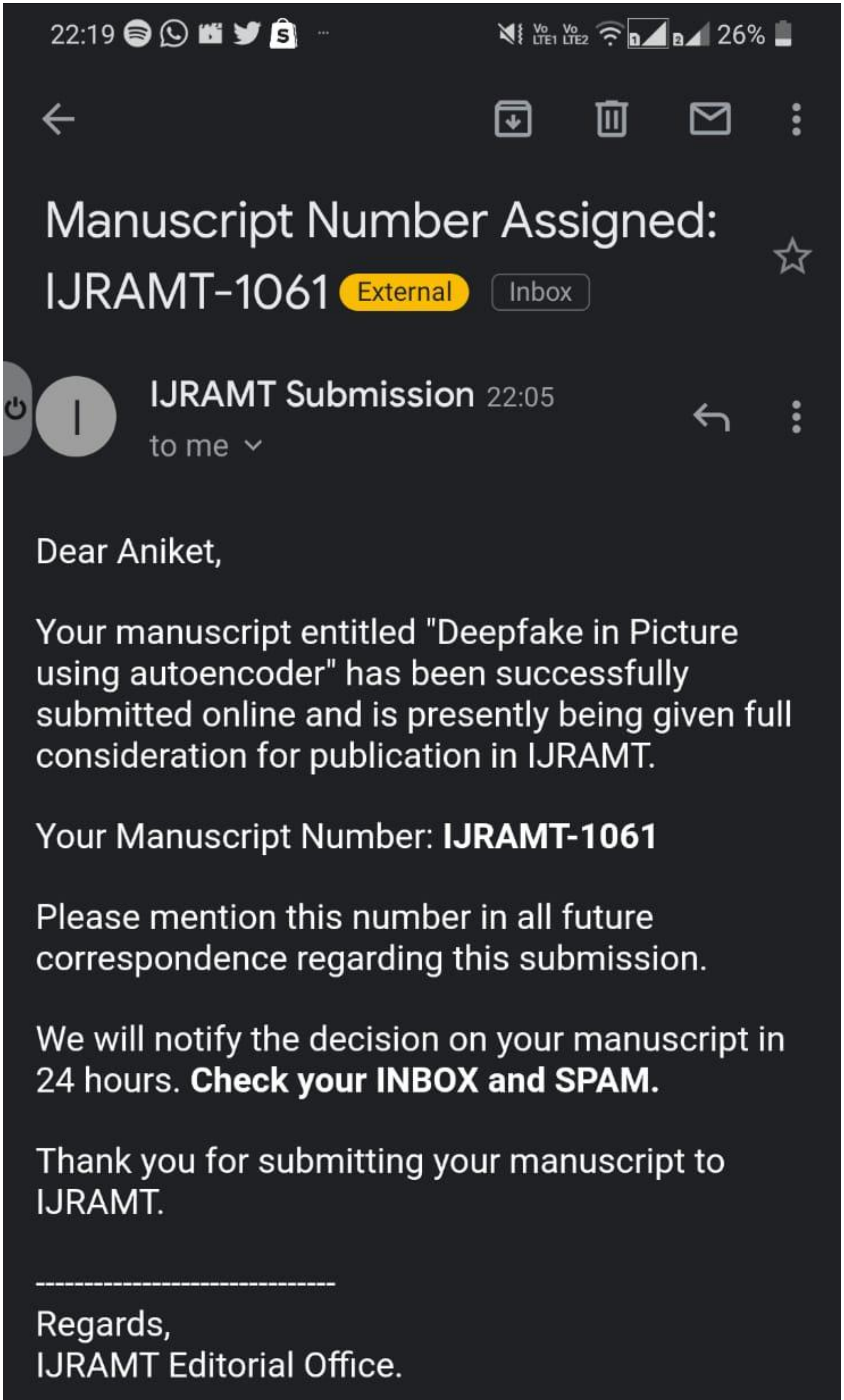
Even though the growing field of innovation around spotting deepfakes is tremendous, we still have a long way to go. The problem is that as soon as someone creates technology that can spot fakes, someone else creates even better fakes. And with advances in artificial intelligence, fakes will only become more sophisticated and realistic over time.

For this reason, it becomes even more critical to take matters into your own hands and conduct a simple search of the information existing behind and around an image or video. With that intelligence you can

create a more comprehensive picture and, ultimately,

discern what is fake and prove what is real.

Deepfake technology facilitates numerous possibilities in the education domain. Schools and teachers have been using media, audio, video in the classroom for quite some time. ***Deepfakes can help an educator to deliver innovative lessons*** that are far more engaging than traditional visual and media formats.


22:19 26%

Manuscript Number Assigned: IJRAMT-1061 External Inbox

IJRAMT Submission 22:05
to me

Dear Aniket,

Your manuscript entitled "Deepfake in Picture using autoencoder" has been successfully submitted online and is presently being given full consideration for publication in IJRAMT.

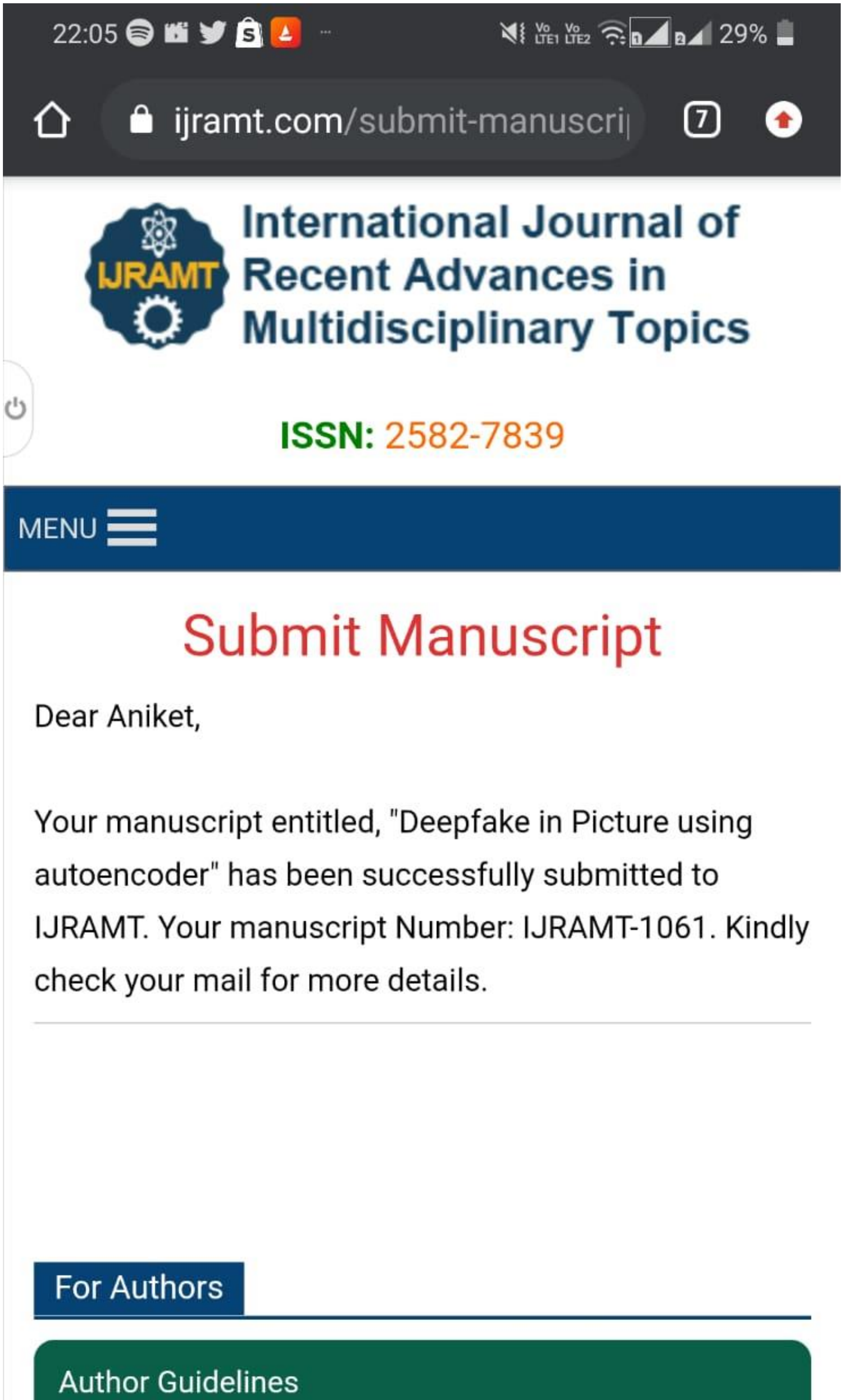
Your Manuscript Number: **IJRAMT-1061**

Please mention this number in all future correspondence regarding this submission.

We will notify the decision on your manuscript in 24 hours. **Check your INBOX and SPAM.**

Thank you for submitting your manuscript to IJRAMT.

Regards,
IJRAMT Editorial Office.



International Journal of Recent Advances in Multidisciplinary Topics

ISSN: 2582-7839

Submit Manuscript

Dear Aniket,

Your manuscript entitled, "Deepfake in Picture using autoencoder" has been successfully submitted to IJRAMT. Your manuscript Number: IJRAMT-1061. Kindly check your mail for more details.