MOHAMED KHIDE UNIVERSITY OF BISKRA

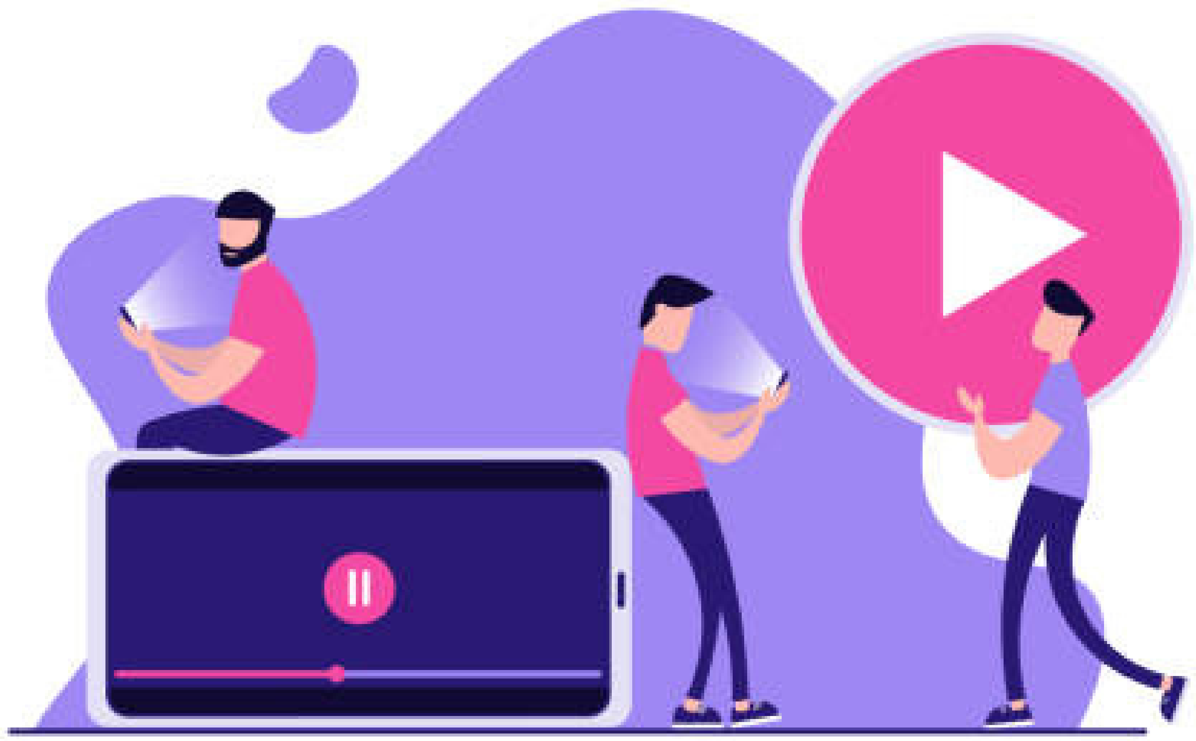
Faculty of Natural and Life Sciences
Computer science department

Deep fake Detection

Proposed by: Meadi Mohemed Nadjib

Directed by: Benaichi Maroua

PRESENTATION PLAN

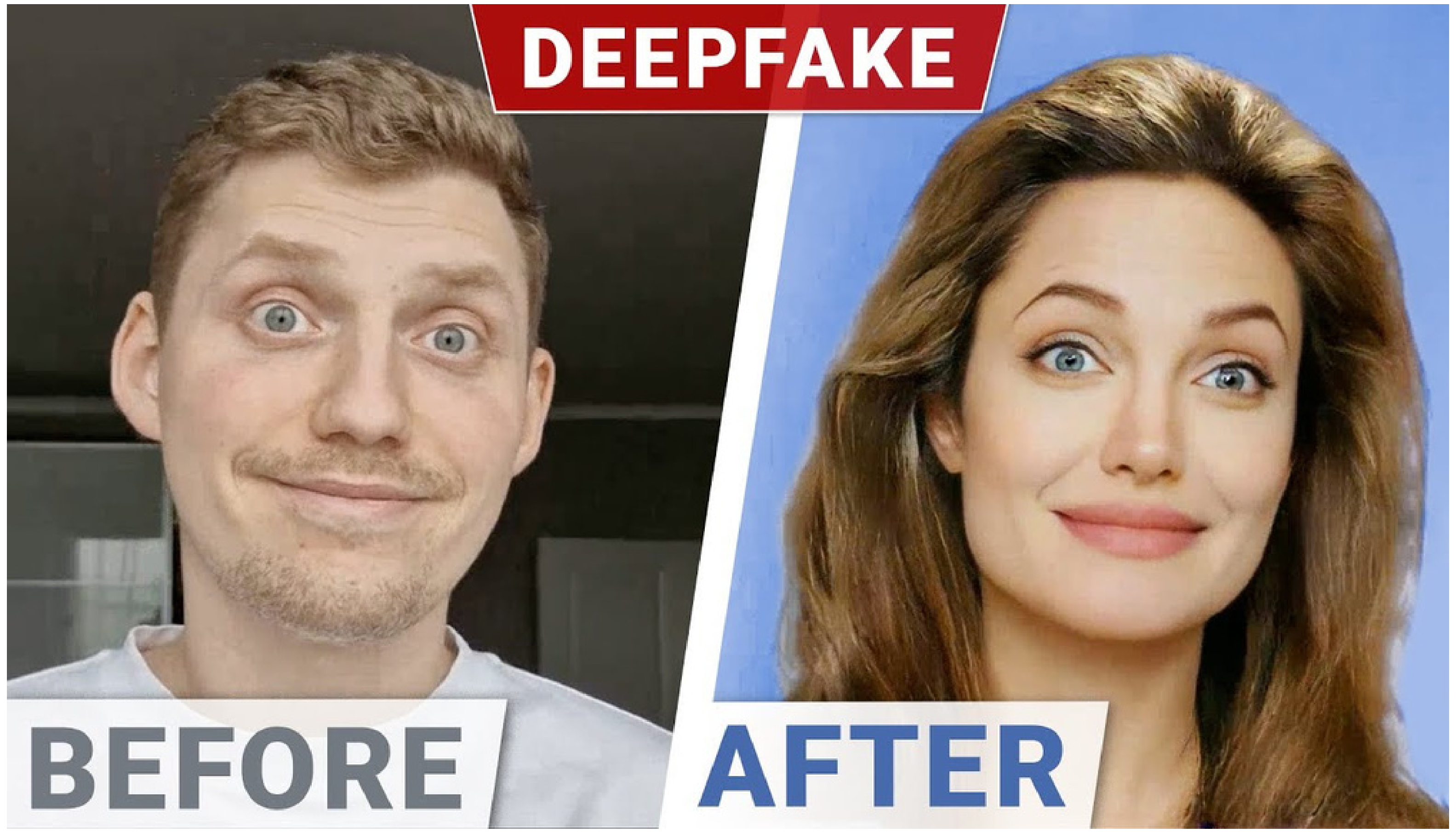


Introduction

As more people have access to technology, many fake videos called 'deepfakes' are being shared online. These videos change someone's face or body to look like someone else. Unfortunately, some people use deepfakes to make videos that hurt famous people and politicians. Although deep fake technology could be good for things like movies and virtual reality, we need to be careful how we use it.

Problematic

How to detect deep fake videos, which can use to spread misinformation?

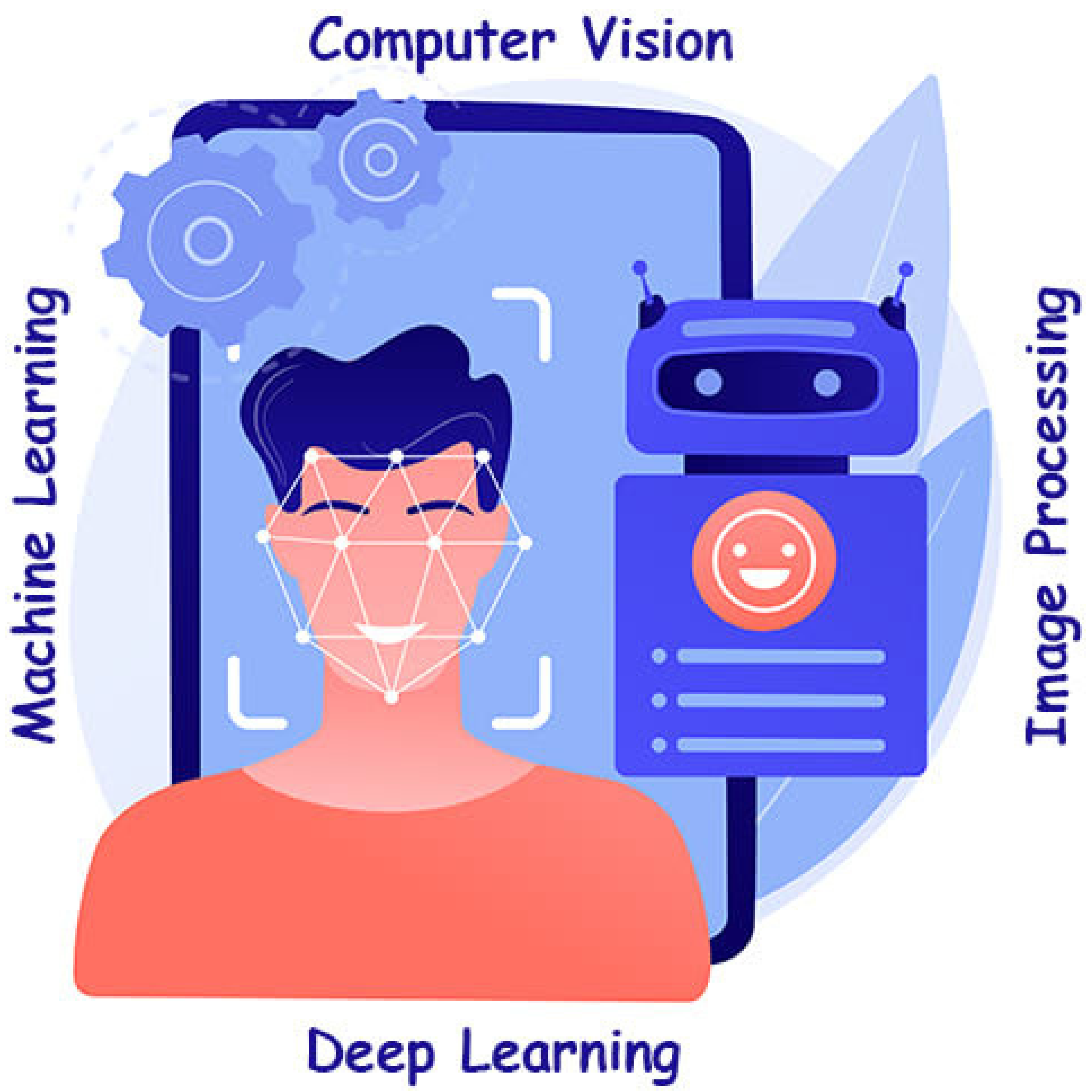


The goal of the project

The main goal of this research is to use artificial intelligence to tell apart real and fake videos. We aim to create a simple system that users can easily use to upload videos and determine if they are genuine or manipulated.

What is Deepfakes?

Deepfake is a technique that uses AI to create realistic but fake audio, video, or images of people, making them appear to say or do things they never actually did.



Types of deep fakes

Face swapping

involves replacing one person's face with another's, making it seem like the second person is saying or doing something they never actually did.

Lip-syncing

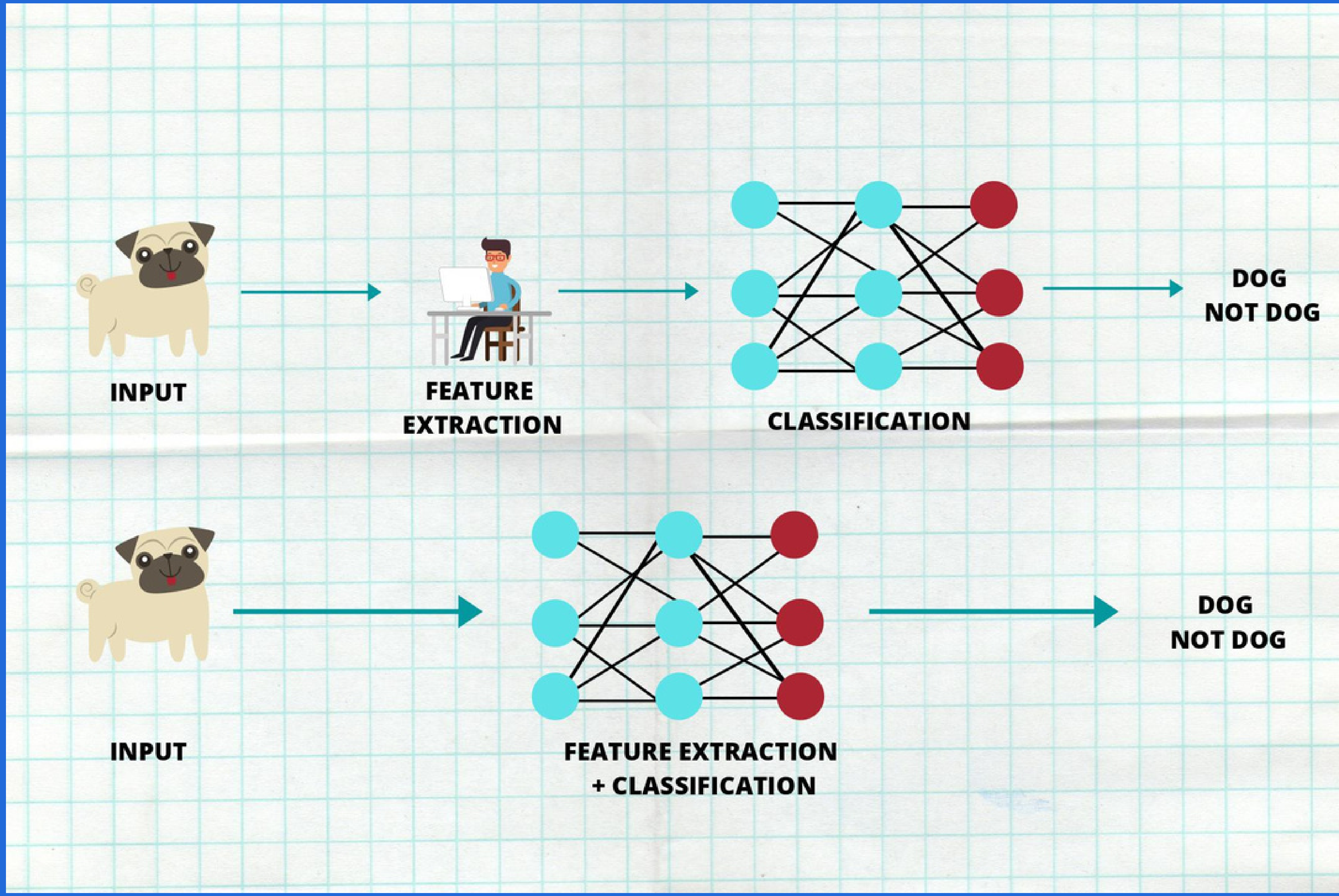
Lip-syncing deep fakes can make a person appear to be saying words they never actually said.

Voice synthesis

Voice synthesis deep fakes can create realistic audio recordings of people saying things they never actually said.

Voice synthesis

Face morphing is a technique that smoothly blends one face into another. Uses to generate artificial biometric face samples that mimic the characteristics of multiple individuals.



Machine Learning

Machine Learning is a subset of artificial intelligence (AI) that defines one of the basic principles of artificial intelligence.
- the ability to learn from experience rather than instructions.

Deep learning

Deep learning is a subset of machine learning that empowers computers to comprehend the world through a hierarchy of concepts. It focuses on training artificial neural networks with multiple layers to learn and extract intricate patterns and representations from data.

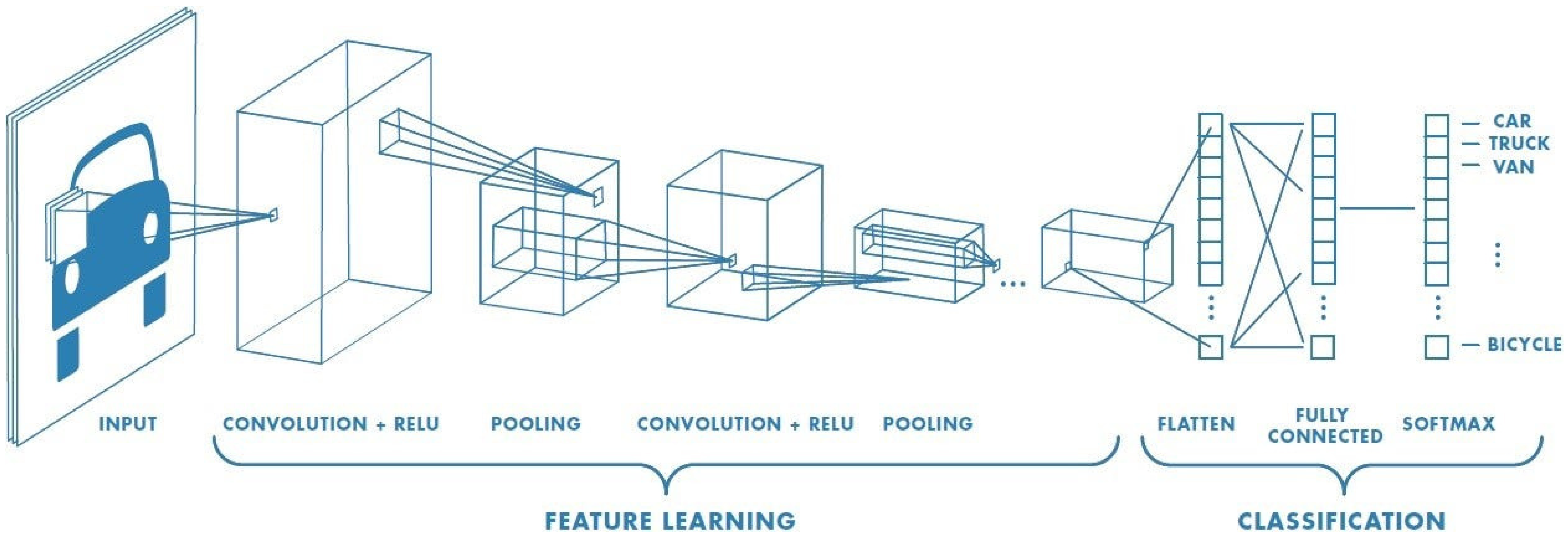
Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of deep learning algorithm designed for analyzing visual data. They use specialized layers and filters to automatically extract meaningful features from images, making them specifically effective in tasks such as image classification.

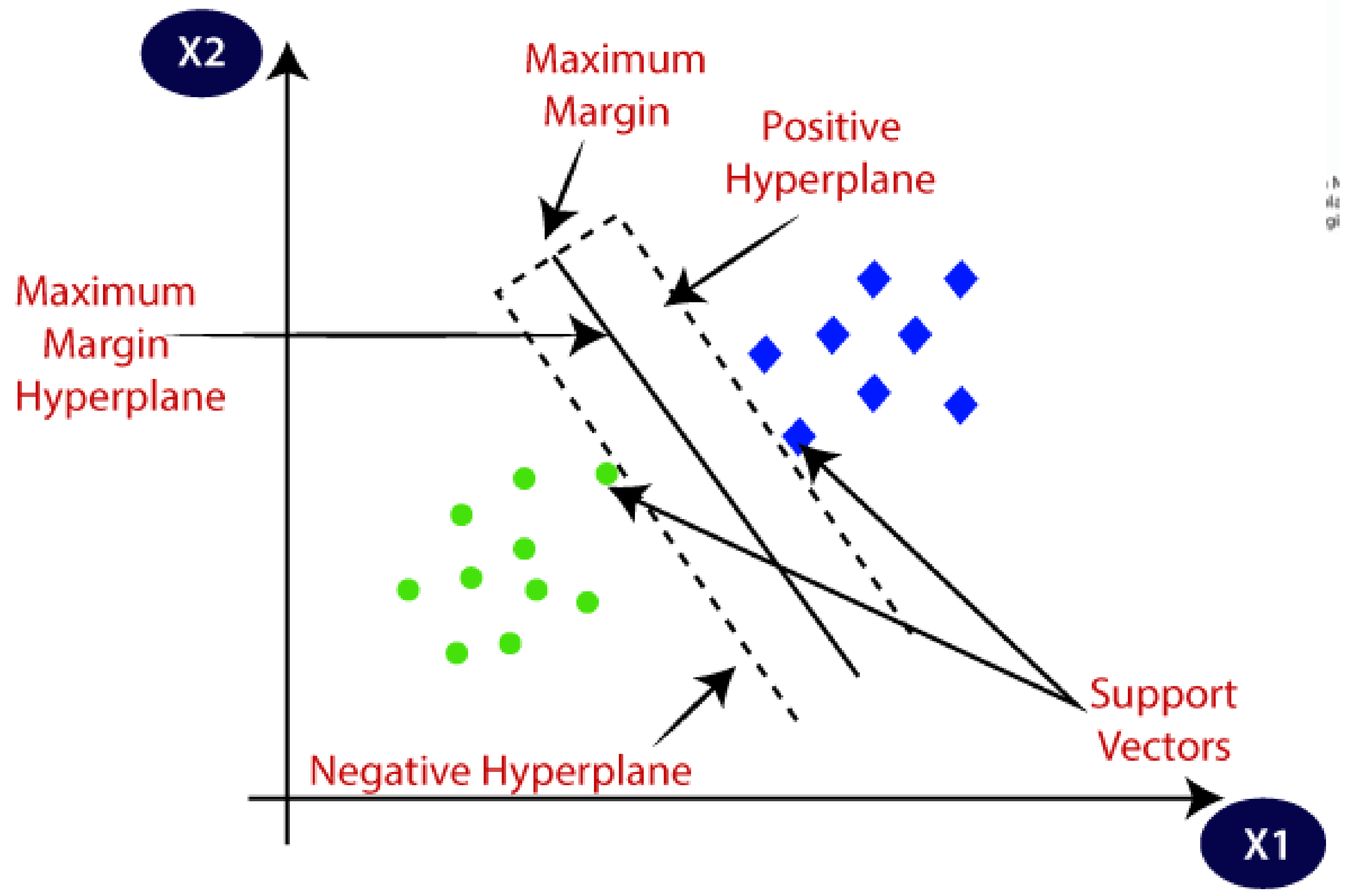
Support Vector Machines (SVM)

Support Vector Machines (SVM) is a supervised machine learning algorithm that can be used for classification or regression tasks.
The main idea behind SVM is to find the hyperplane that best separates the data points into different classes.

Convolutional Neural Networks



Support Vector Machines (SVM)



Related works

"Exposing Deep Fakes Using Inconsistent Head Poses"

- This paper explores the detection of deep fake videos by analyzing inconsistent head poses between the synthesized face and the background in the video.

The authors report accuracy results of around 80-85%.

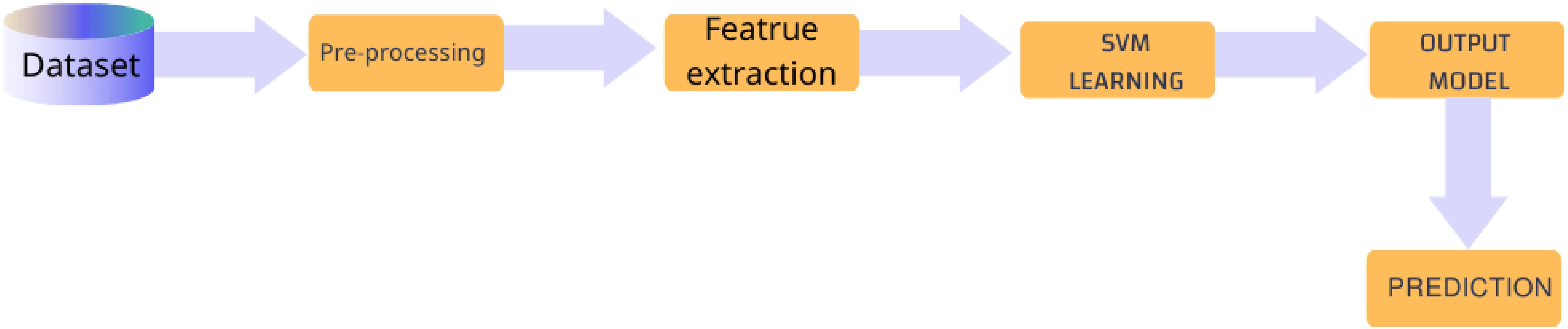
"Deepfake Video Detection Using Convolutional Neural Networks" (2020)

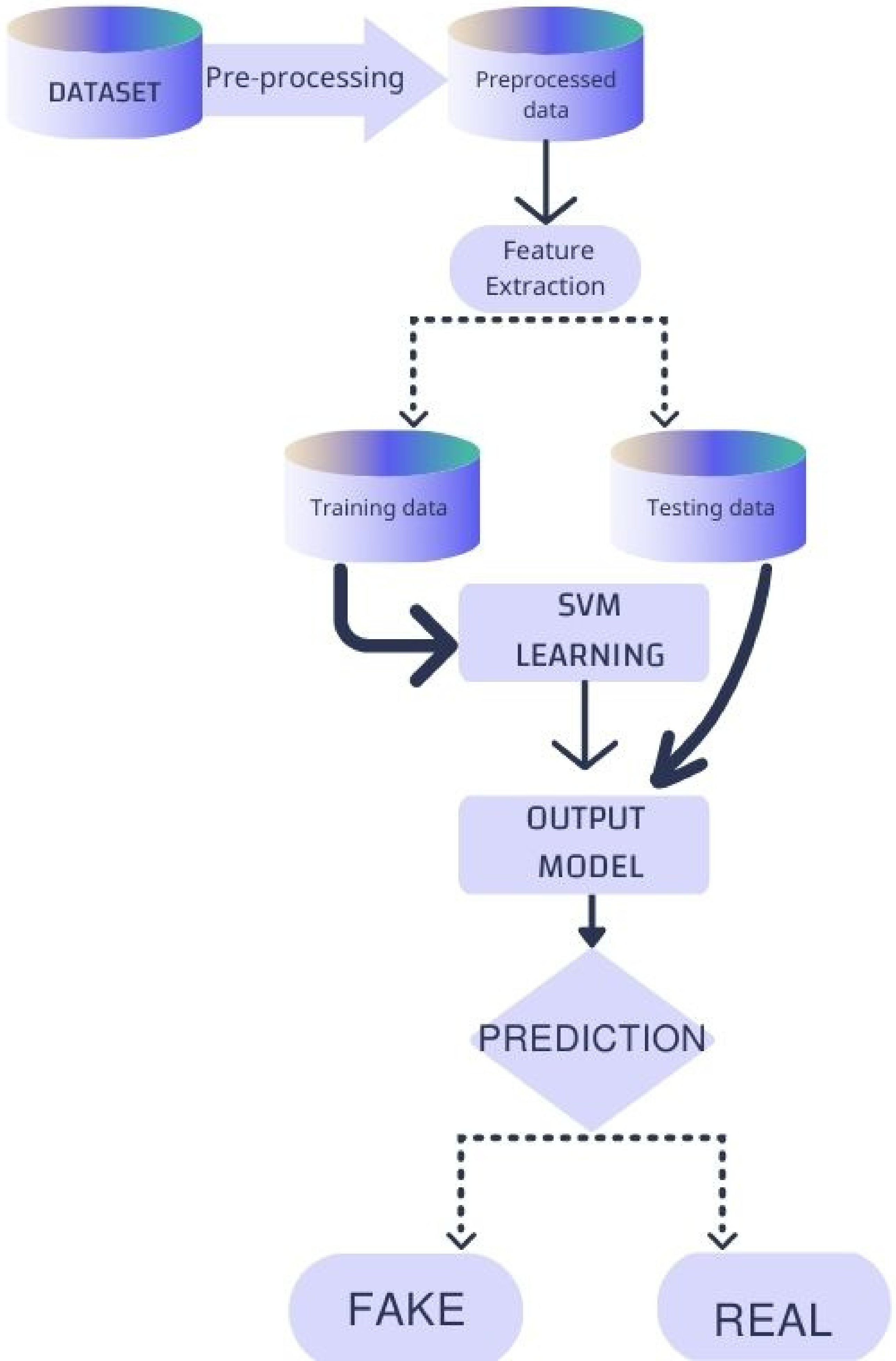
- This paper proposes a deep fake detection method that utilizes Convolutional Neural Networks (CNNs) to extract features from video frames and applies a classification algorithm. The authors report accuracy results of around 70%.

System design

System design

Global architecture.





System design

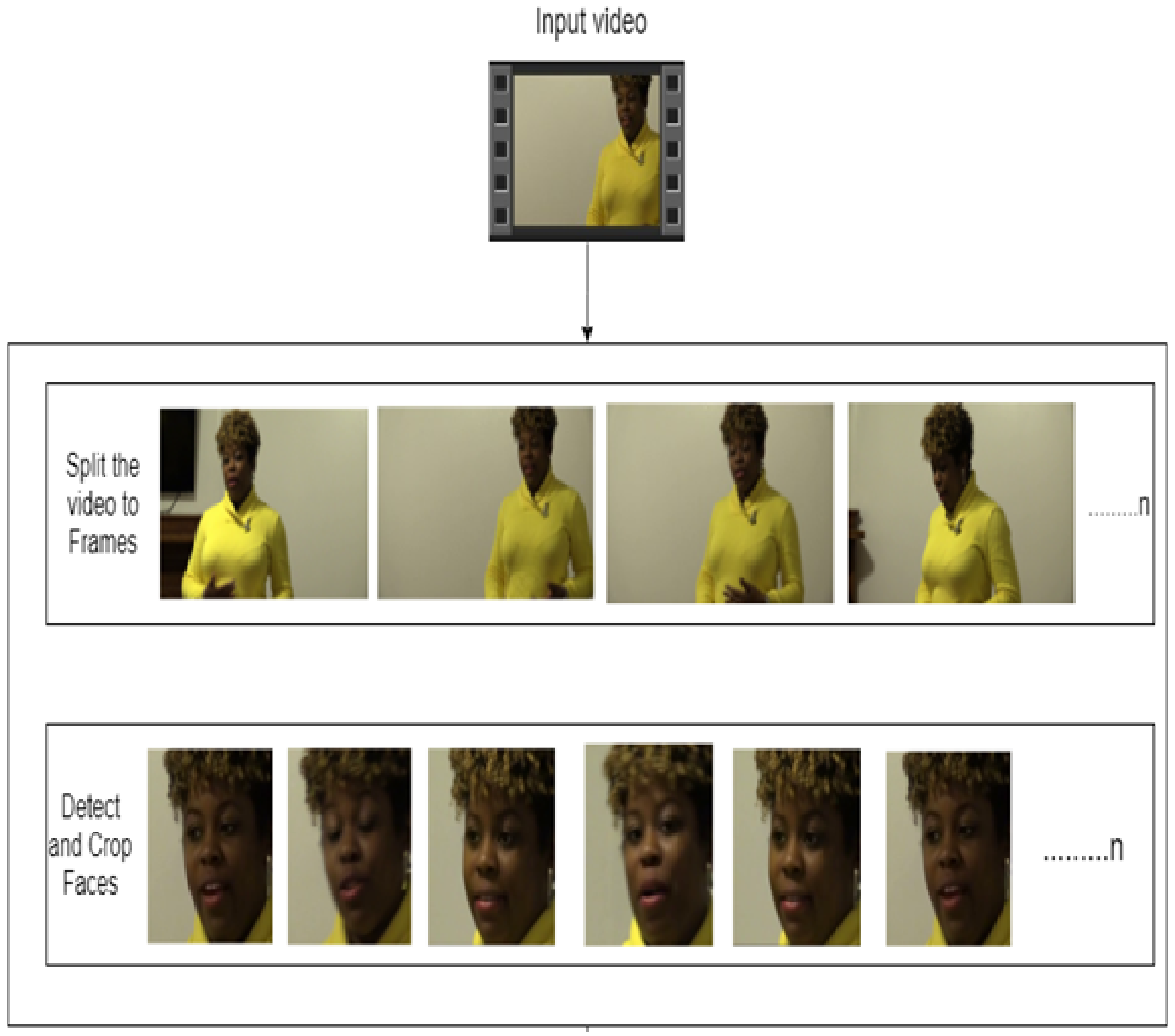
Detailed architecture of the system

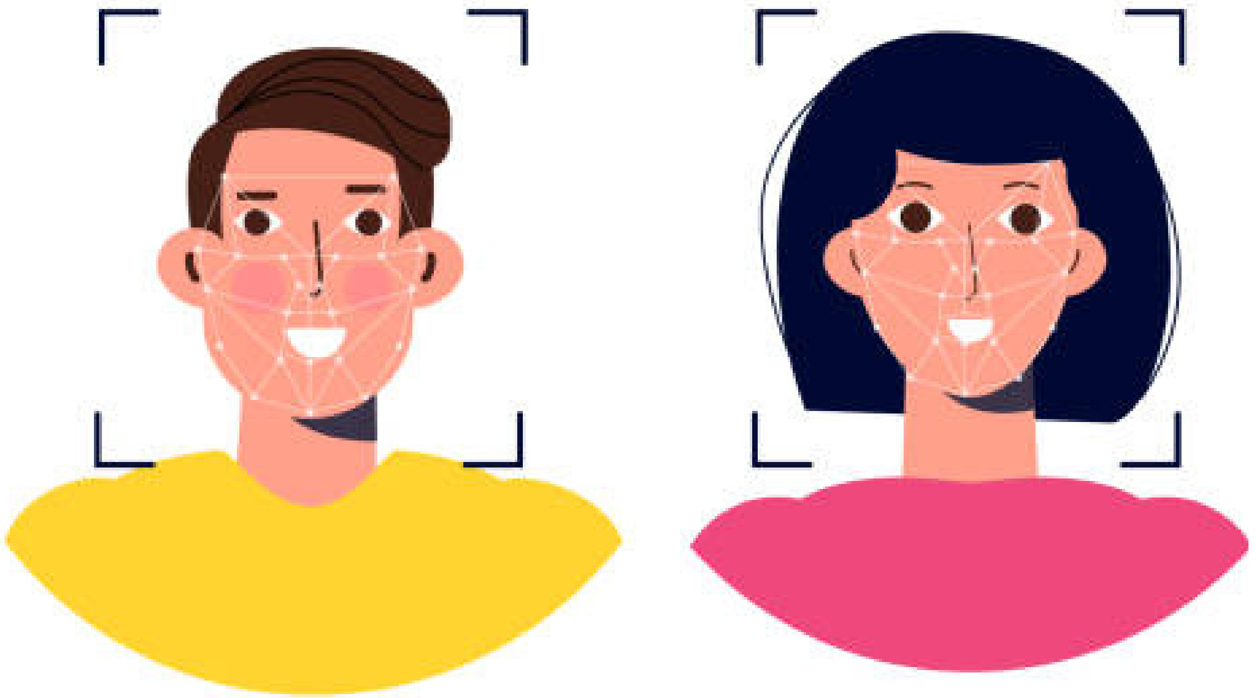
Preprocessing:

Dataset

- Split the video into frames to obtain individual images.
- Convert the color space of the images from BGR to RGB format.
- Perform face detection on each image using the MTCNN (Multi-Task Cascaded Convolutional Networks) algorithm.
- Add margins to the bounding box of the detected face to ensure that the entire face is captured.
- Crop the face from the image using the adjusted bounding box coordinates.

PREPROCESSED DATASET





Feature Extraction:

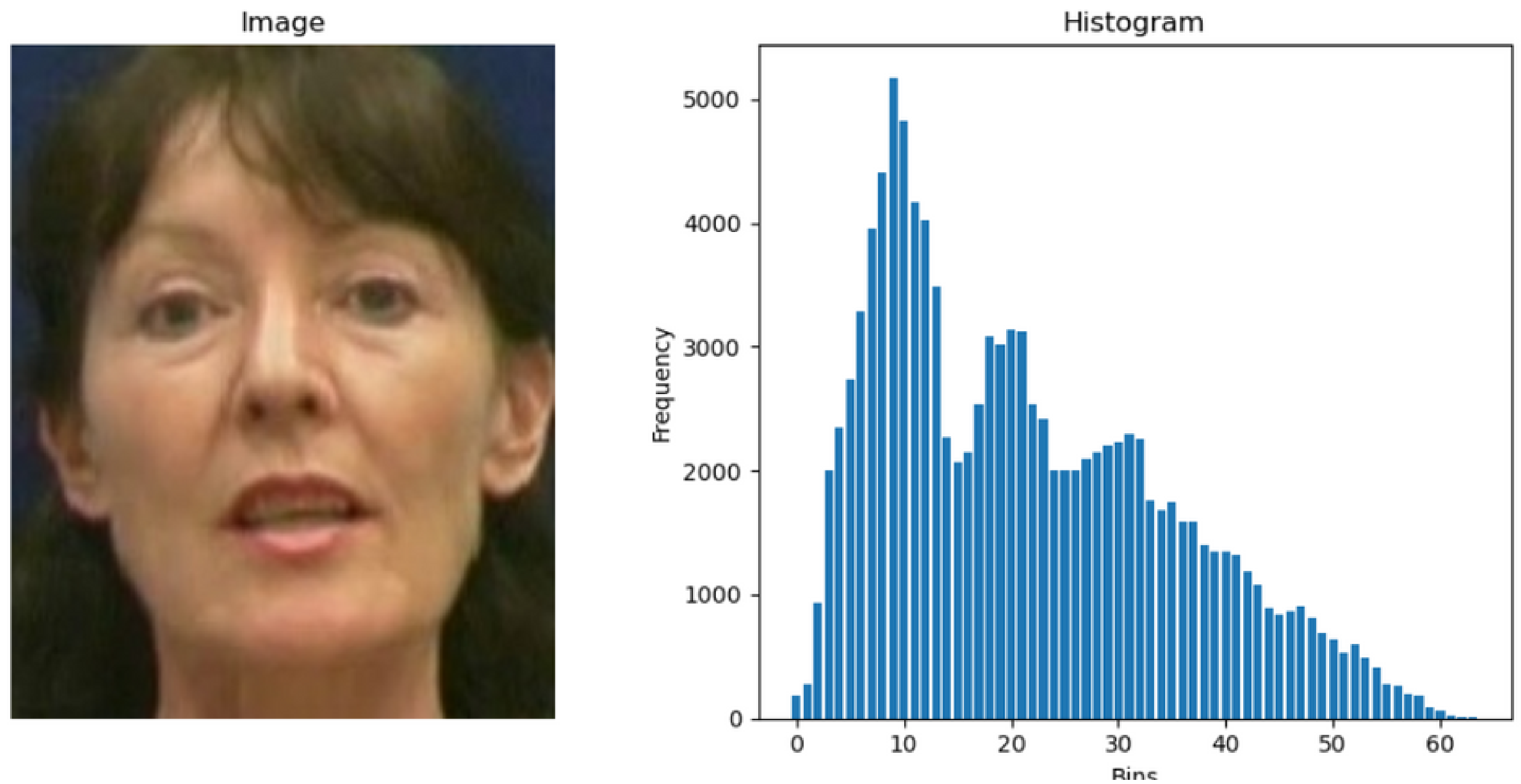
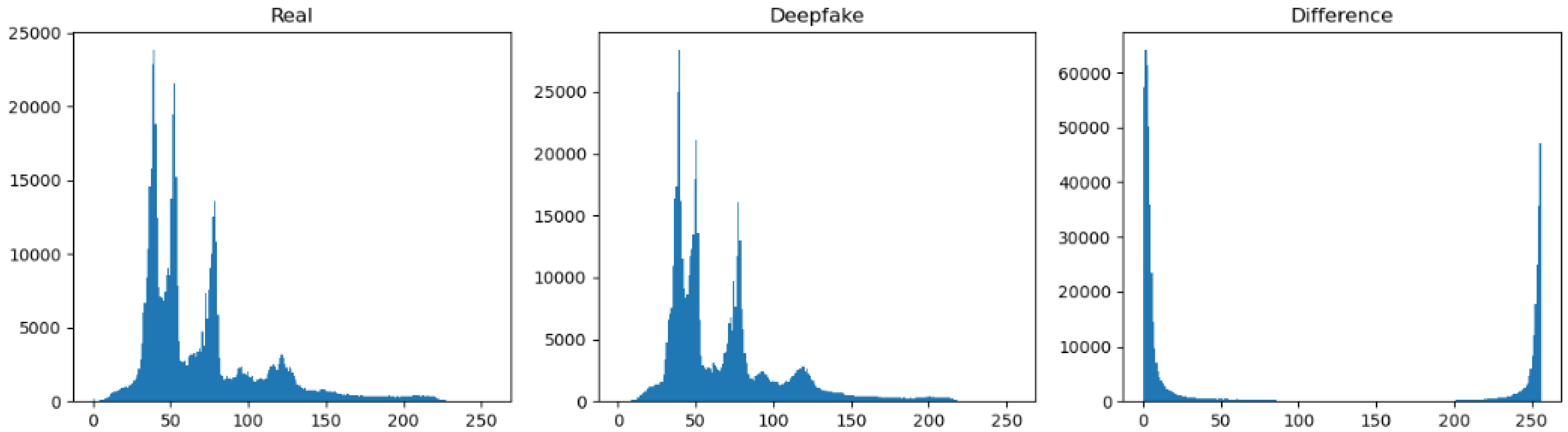


Image and its Histogram



Histogram Representation

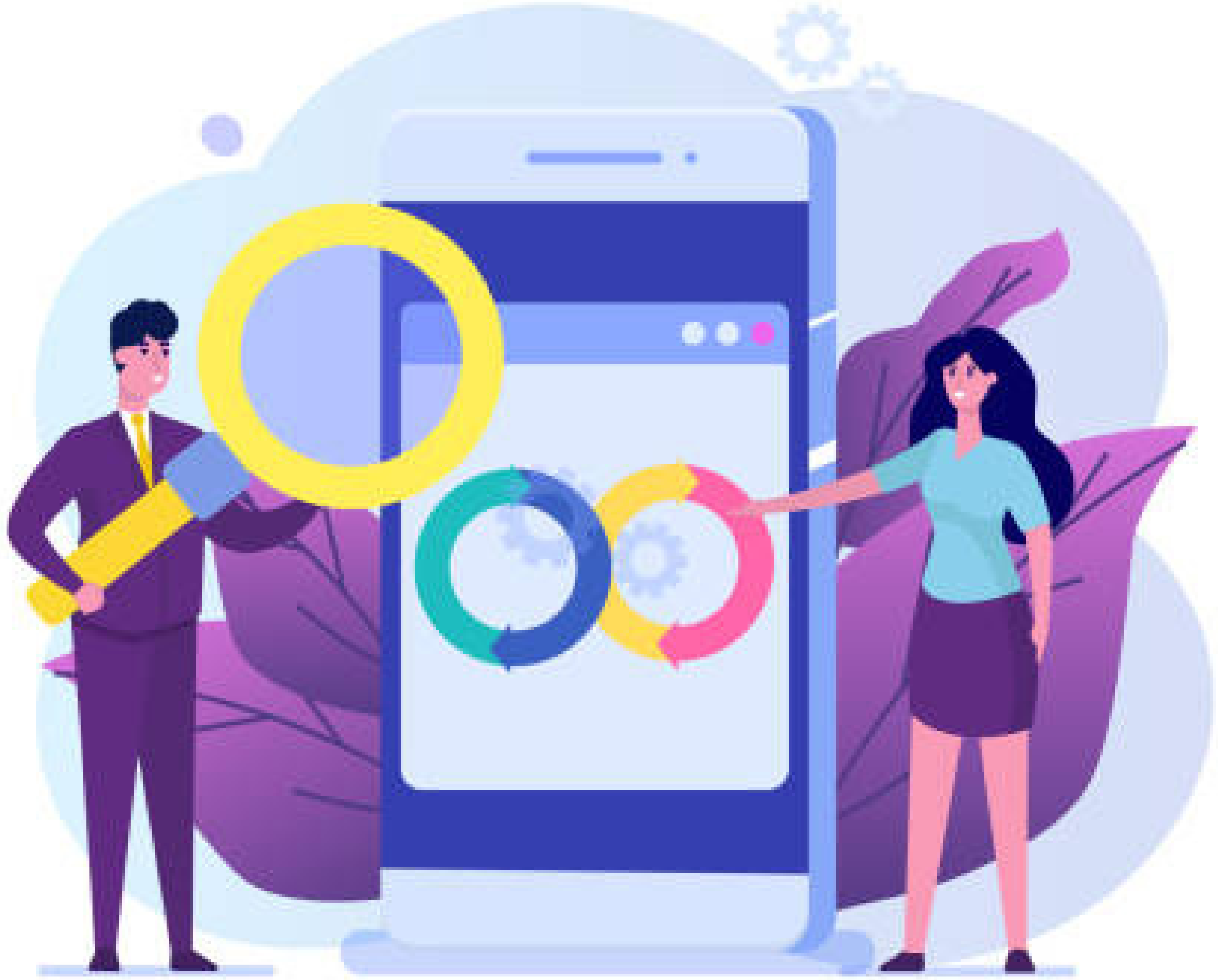
Original Image

Blurred Image

Original Image VS Blurred Image

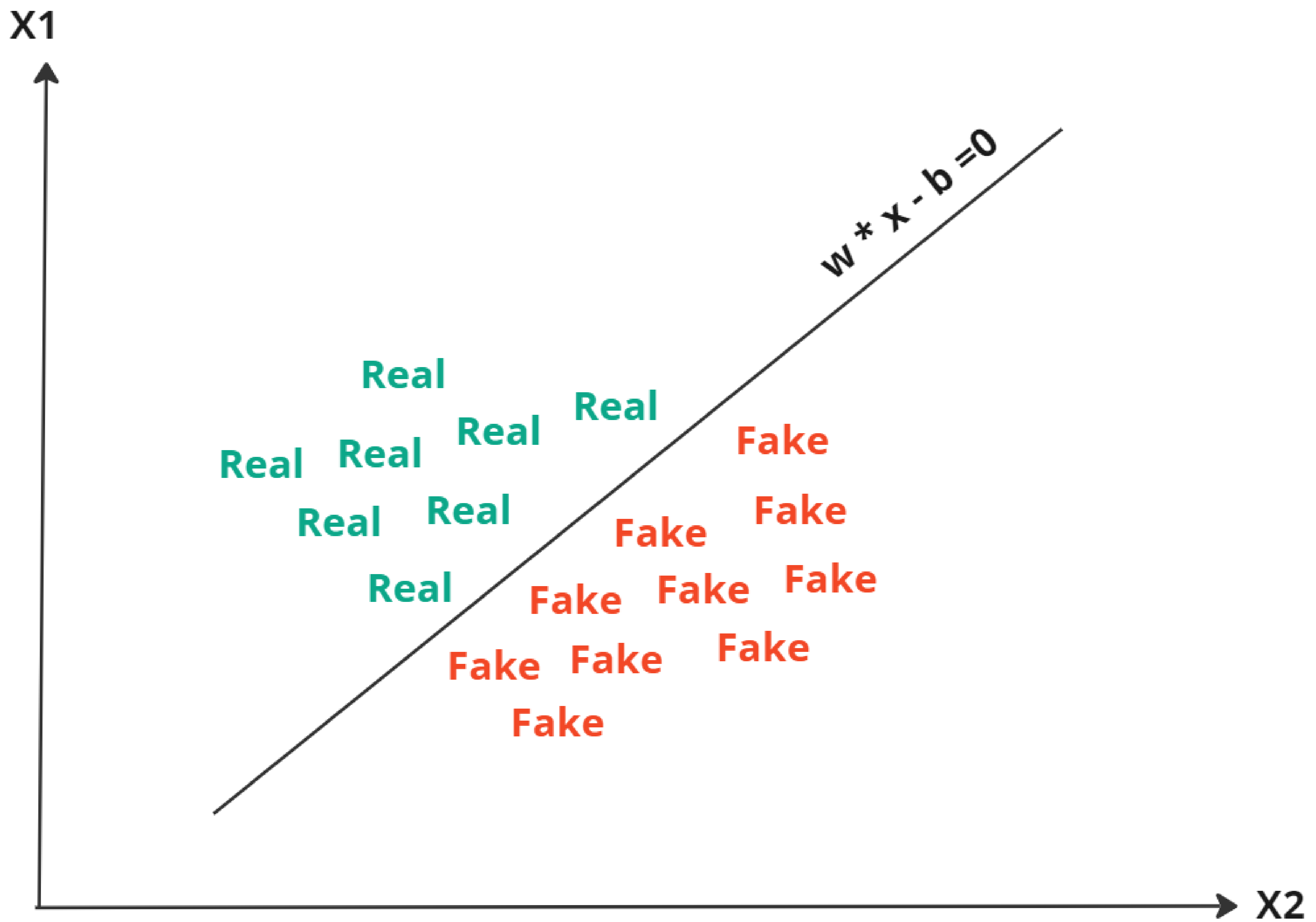
Metric	Value
MSE	5
PSNR	7
SSTM	8
Hist	[Array] [1238]

Models:

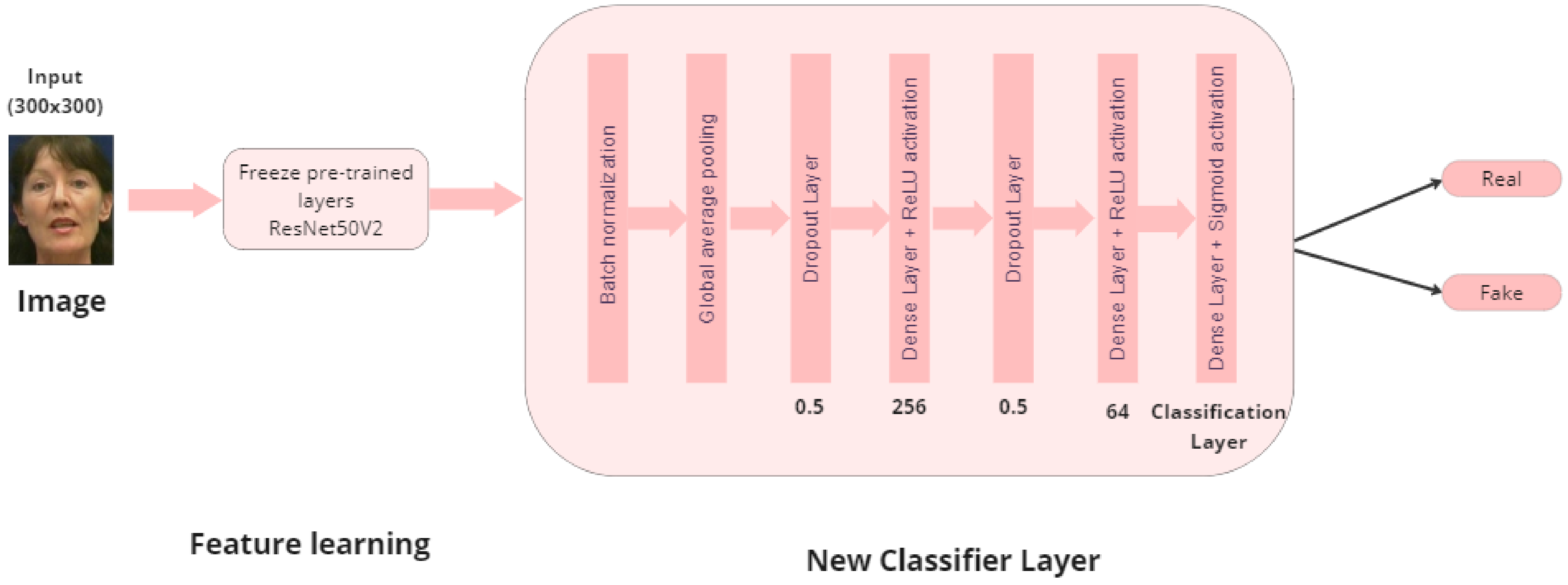


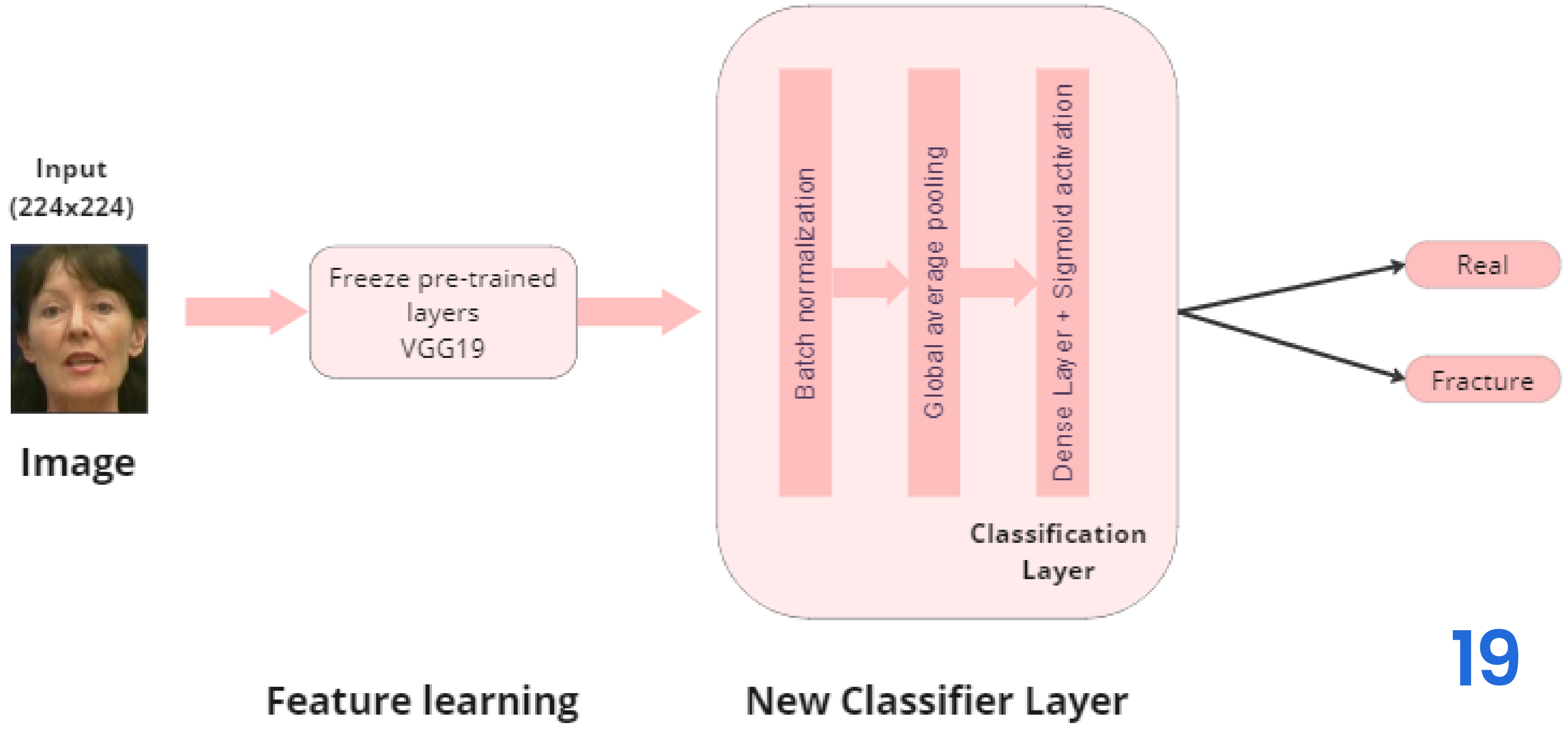
Machine learning Model

Support Vector Machines (SVM)



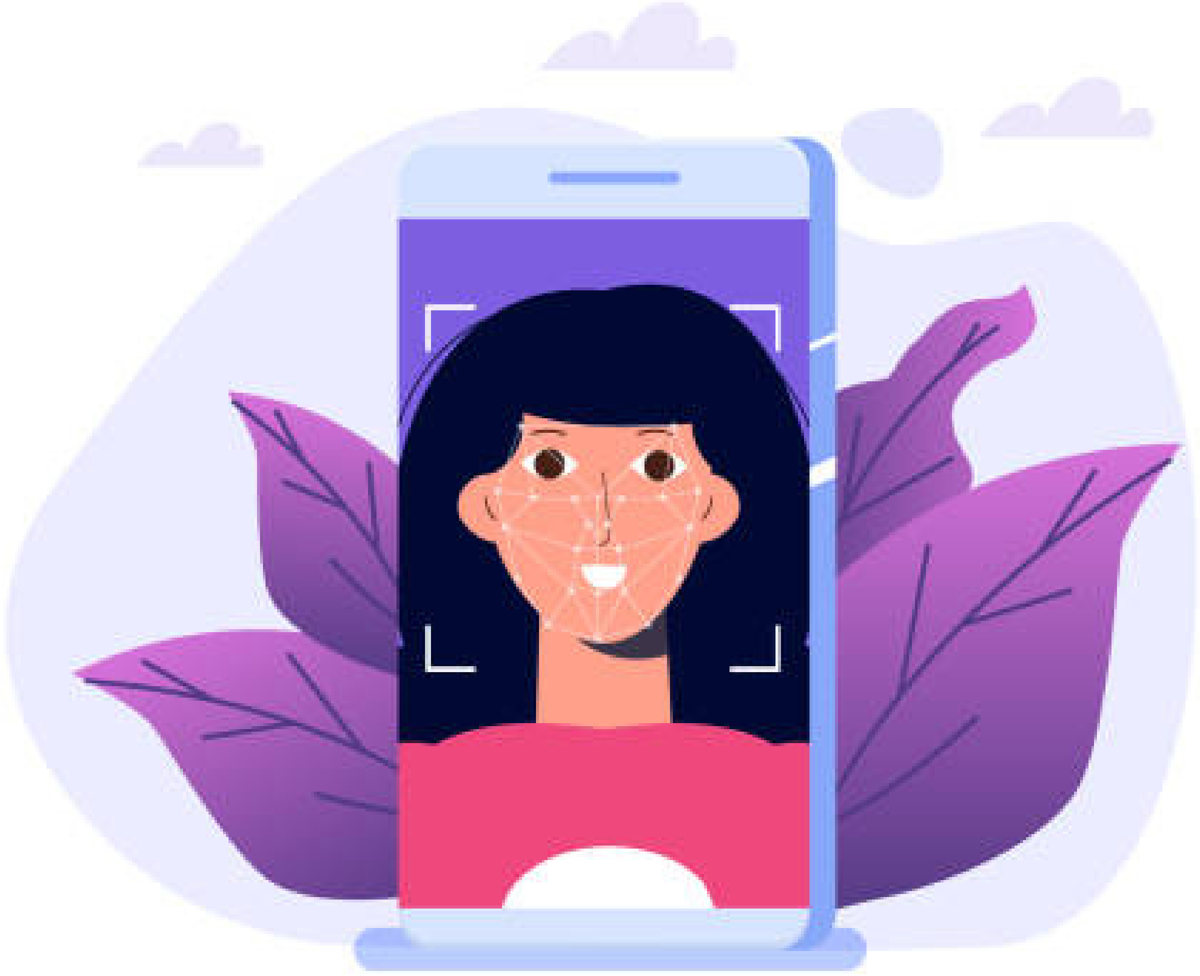
Deep learning Models





IMPLEMENTATION

Preprocessing:



```python
#Function detect a face, convert it to RGB, and crop the image around the face
def detectFace(image, box_scale = 0.15):
    RGBImage = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    detection = detector.detect_faces(RGBImage) #Using the detect_faces function in the MTCNN library
    #Putting a box around the face
    if len(detection) > 0:
        squareBox = detection[0]['box']
        x = int(squareBox[0] - box_scale * squareBox[2])
        y = int(squareBox[1] - box_scale * squareBox[3])
        w = int(squareBox[2] + box_scale * squareBox[2] * 2)
        h = int(squareBox[3] + box_scale * squareBox[3] * 2)

        #Cropping the image
        return image[y:y+h, x:x+w, :].copy()
    return None

#Function to detect the faces and save the cropped images to disk
def detectAndSave(videoName, box_scale = 0.15, limitFaces = -1, saveFaces = True):
    detector = MTCNN()
    faces = list()
    # add '_face' at the end to differentiate face images
    faceName = os.path.splitext(videoName)[0] + '_face'

    #Reading the frames like we did before but this time only saving the faces instead of the entire frames
    capture = cv2.VideoCapture(videoName)
    numFrames = int(capture.get(cv2.CAP_PROP_FRAME_COUNT))
    for frameNum in range(numFrames):
        #Loop until the limit is no longer -1
        if limitFaces != -1 and frameNum >= limitFaces:
            break
        capture.set(cv2.CAP_PROP_POS_FRAMES, frameNum)
        ret, frame = capture.read()
        #Using our detect_face function above
        face = detectFace(frame, box_scale=box_scale)
        if face is not None:
            faces.append(face)
            if saveFaces:
                cv2.imwrite(faceName + '_' + str(frameNum) + '.jpg', face)
    return faces
```

Feature Extraction:

```python
#Creating functions to blur an image, compute an MSE, PSNR, and SSIM
import skimage.metrics

#Function to blur an image using Gaussian blur
def blurImage(image, kernel_size = 5, sigma = 0.5):
  return cv2.GaussianBlur(image, (kernel_size, kernel_size), sigma)

#Function to compute MSE
def computeMSE(x, y):
  return skimage.metrics.normalized_root_mse(x, y)

#Function to compute the PSNR
def computePSNR(x, y):
  return skimage.metrics.peak_signal_noise_ratio(x, y, data_range=255)

#Function to compute the SSIM
def computeSSIM(x, y):
    return skimage.metrics.structural_similarity(x, y, multichannel=True, win_size = -1,gaussian_weights=True, sigma=1.5,
                                                  use_sample_covariance=False, data_range=255)

#Instead using matplotlib to look at the histograms, let's actually compute a histogram to use as a feature
def computeHist(image, bins = 64):
  hist, bins = np.histogram(image.ravel(), bins, [0,256], density = True)
  return hist
```

```python
def computeFeatures(image):
    blurredImage = blurImage(image)
    mse = computeMSE(image, blurredImage)
    psnr = computePSNR(image, blurredImage)
    ssim = computeSSIM(image, blurredImage)
    hist = computeHist(image, bins=64)

    featuresArray = np.concatenate([[mse], [psnr], [ssim], hist])
    return featuresArray
```

Split data

Deep learning Models

```
data = pathlib.Path(path)
splitfolders.ratio(data, output='data/', seed=42, ratio=(0.8, 0.1, 0.1), group_prefix=None)
```

Machine learning Model

```
from sklearn.model_selection import train_test_split

#Splitting our datasets into 80% training and 20% testing
realTrain, realTest = train_test_split(realVideosList, test_size = 0.2, random_state = 42)
fakeTrain, fakeTest = train_test_split(fakeVideosList, test_size = 0.2, random_state = 42)
```

Machine learning Model

Support Vector Machines (SVM)

```
svmModels = []
for kernel in ['linear', 'sigmoid', 'rbf']:
  svm = sklearn.svm.SVC(kernel = kernel, gamma = "auto", class_weight = "balanced")
  svm.fit(normalizedFeatures, trainingLabels)
  svmModels.append(svm)
```

Deep learning Models

Transfer learning model

```
import tensorflow as tf
from tensorflow.keras.applications import ResNet50V2
from tensorflow.keras.models import Model
from tensorflow.keras.layers import BatchNormalization,GlobalAveragePooling2D, Dropout, Dense
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler

# Load pre-trained ResNet50V2 model
base_model = ResNet50V2(include_top=False, weights='imagenet', input_shape=(300, 300, 3))

# Freeze pre-trained layers
base_model.trainable = False

# Add your own layers on top of the pre-trained model
x = base_model.output
x = BatchNormalization()(x)
x = GlobalAveragePooling2D()(x)
x = Dropout(0.5)(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(64, activation='relu')(x)
predictions = Dense(2, activation='sigmoid')(x)
# Create a new model with pre-trained base and your own layers on top
model = Model(inputs=base_model.input, outputs=predictions)
```

```
gg = VGG19(input_shape=(224, 224, 3), weights='imagenet', include_top=False)

# Do not train the pre-trained layers of VGG-19
for layer in gg.layers:
    layer.trainable = False

x = Flatten()(gg.output)
prediction = Dense(2, activation='sigmoid')(x)
model = Model(inputs=base_model.input, outputs=predictions)

# View the structure of the model
model.summary()
```

Training the model

Machine learning Model

```
svm.fit(normalizedFeatures, trainingLabels)
```

Deep learning Models

```
history = model.fit(
    train_data,
    validation_data=val_data,
    batch_size=64,
    epochs=15,
    callbacks=[
        tf.keras.callbacks.EarlyStopping(
            monitor='val_loss',
            patience=5,
            restore_best_weights=True
        )
    ]
)
```

RESULTS

Model	Accuracy	Precision (Class 0)	Precision (Class 1)	Recall (Class 0)	Recall (Class 1)	F1-Score (Class 0)	F1-Score (Class 1)
Linear SVM	0.83	0.74	0.89	0.83	0.83	0.78	0.86
Sigmoid SVM	0.60	0.46	0.73	0.62	0.59	0.53	0.65
RBF SVM	0.90	0.86	0.93	0.88	0.92	0.87	0.92
VGG19 Model	84	0.81	0.87	0.86	0.82	0.83	0.84
ResNet50V2 Model	0.78	0.79	0.77	0.80	0.76	0.78	0.71

RESULTS

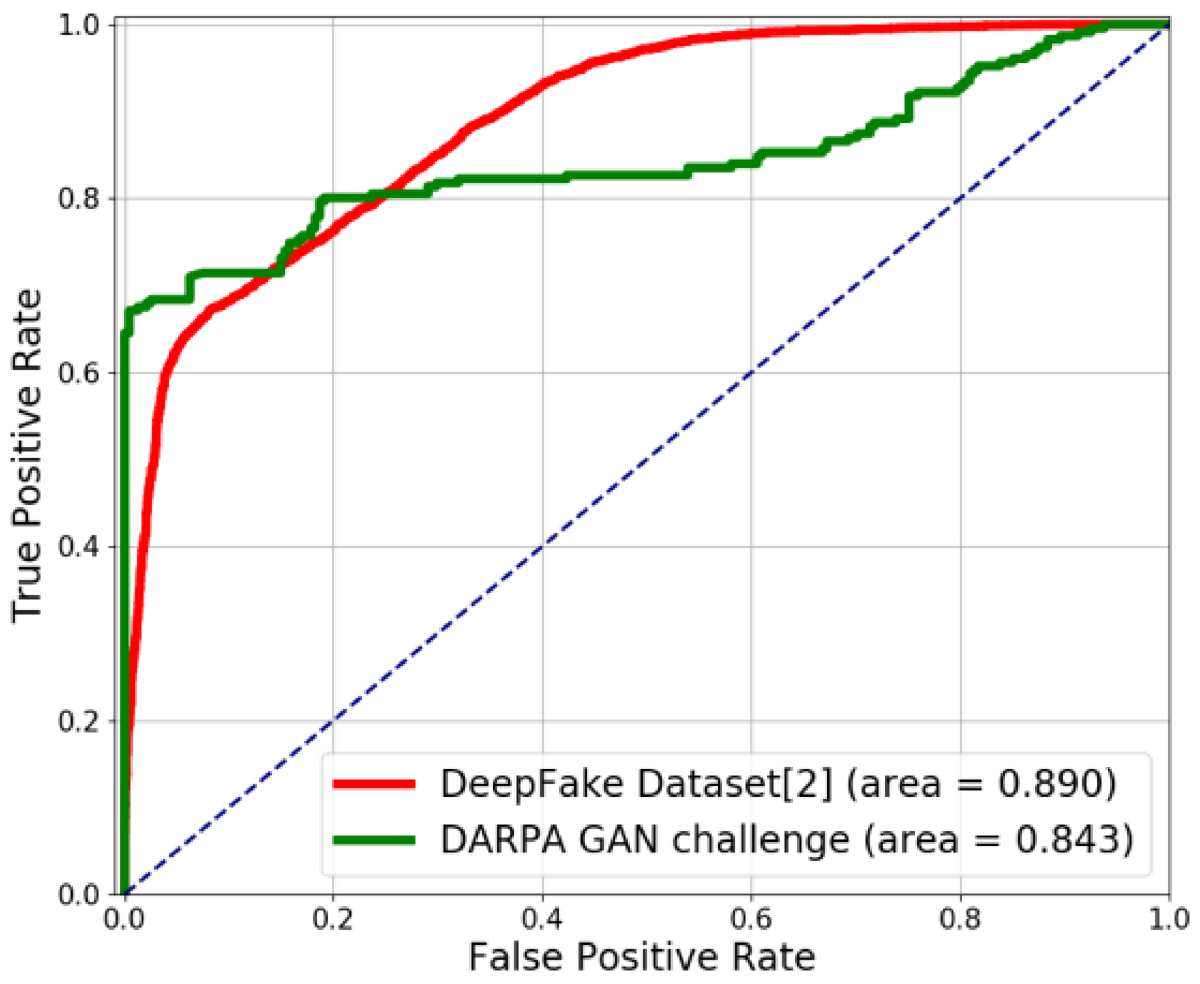
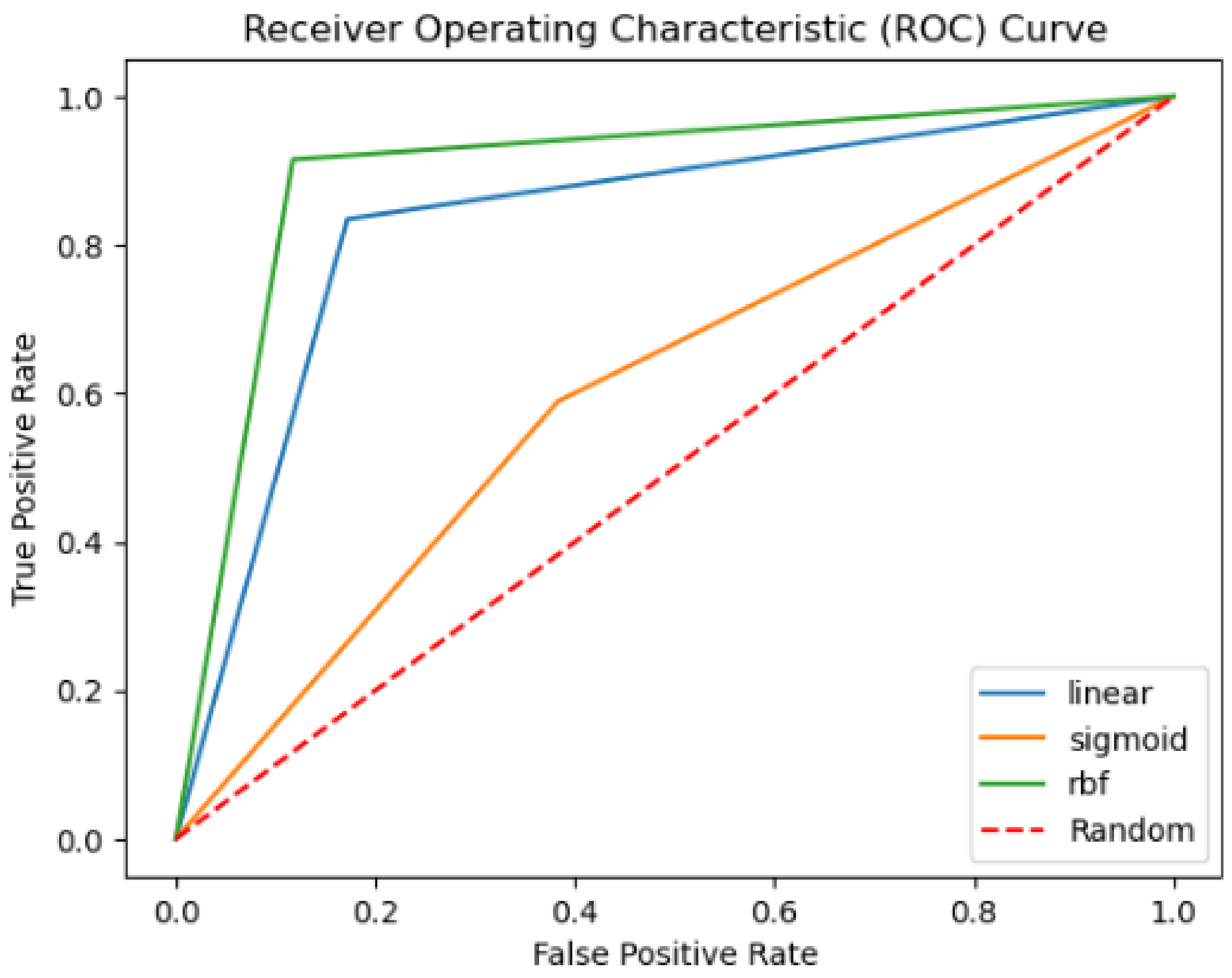


Fig. 4. *ROC curves of the SVM classification results, see texts for details.*



CONCLUSION

What is created by AI can also be corrected by AI.

Future work

In our future work on deep fake detection, we want to develop innovative methods that enable the simultaneous analysis of visual and auditory cues.

Thank you