National College of Ireland

Hybrid Bayesian CNN-GAN architecture for Deepfake detection

MSc Research Project

MSc in Artificial Intelligence

Saloni Suhas Deshpande

Student ID: x22197001

School of Computing

National College of Ireland

Supervisor: Mayank Jain

National College of Ireland
MSc Project Submission Sheet
School of Computing

Student Name: ……Saloni Suhas Deshpande……

Student ID: ……x22197001……

Programme: …Msc Artificial Intelligence **Year:** …2023 - 2024……

Module: ……Practicum (H9PRAC)……

Supervisor: …Mayank Jain……

Submission Due Date: ……11/08/2024……

Project Title: Hybrid Bayesian CNN-GAN architecture for Deepfake detection

Word Count: **6568** **Page Count**……17……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……

Date: ……8/08/2024……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Hybrid Bayesian CNN-GAN architecture for Deepfake detection

Saloni Suhas Deshpande
x22197001
MSCAI1
National College of Ireland

Abstract

Deepfakes have a huge impact on from multiple standpoints. Current deep fake detection tools lack real-time detection and accuracy to accommodate the classification speed between classifying the data into real and fake. Many researchers have tried to implement various models like hybrid CNN-LSTM or ResNet, MesoNet, or InceptionNet V2 but each model has its drawbacks when it comes to attention to detail thus we proposed a new hybrid model of CNN-GAN with a Bayesian approach for its architecture and compared it against two models. This work aims to contribute a new model and in doing so we have implemented our new model on the famous Deepfake Detection Challenge (DFDC) dataset by using face extraction from video frames and then our model was evaluated using a Confusion Matrix and comparing the accuracy results to most commonly used deep learning models. We have discovered that our proposed model has 80% accuracy and works within the desired speed of the pipeline. Thus, we conclude our experiment on deep fake detection using deep learning.

Keywords—Deep fake, CNN, Generative Adversarial Networks, autoencoder, deep learning, Bayesian, detection

1. Introduction

The concept of deepfake technology has been around since the 1990s yet it only started gaining attention in the 2010s. The term ‘deepfake’ was coined by a Reddit moderator in 2017. Back in the 1990s, deepfakes referred to the use of CGIs to generate fake human images. In recent times, the word ‘deep fakes’ is taken in the context of audio or visual manipulation. Following is the categorization:

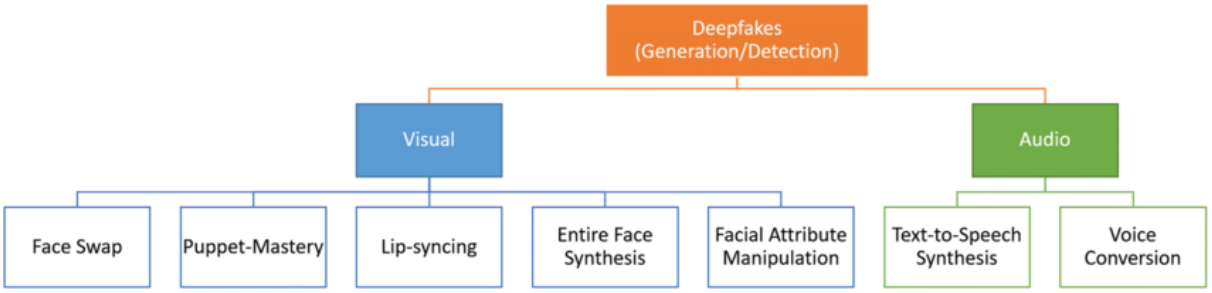


Figure 1: Categories of deep fakes (image source [23])

The use of this technology has been immensely implemented in creative media with ‘face swap’ or ‘facial attribute manipulation’. In this field, the biggest contribution was made by Ian Goodfellow as he and his team introduced the concept of GAN (Generative Adversarial Network). Eventually, this doubled the generation of all types of deepfakes including but not limited to audio, video, and images. [4]

The impact of deepfakes can be massive if these fabrications are used to create a false narrative in politics or to defame a public figure which might also cause social injustice by creating falsified events to spread misinformation like wildfire. Even though countries like the United States tried to create laws in 2019 to prevent destruction with the use of deepfakes, the laws haven’t been that impactful, thus we need detection tools that work in real-time. It’s a shocking fact that this invention is used to manufacture unsolicited explicit material where people’s images can be incorporated in improper situations without their permission. Such cyber violation not only breaches confidentiality but can also culminate in intense inner conflicts and spoil someone’s good name. Some other ways that deepfakes can influence are when scammers use audio deepfakes to scam people or when deepfakes are used to damage a company’s reputation which may also be leading to changes in the stock market if it is a publicly traded company. Deepfakes can also be used to disrupt the educational system by feeding false narratives to the younger generation that are easier to manipulate and can disrupt the trust our society puts in social media. Deepfakes can be used to replace jobs in the creative industry leaving many unemployed. [3]

As we read up on the many impacts, let us take a look at the advancement of this technology below:

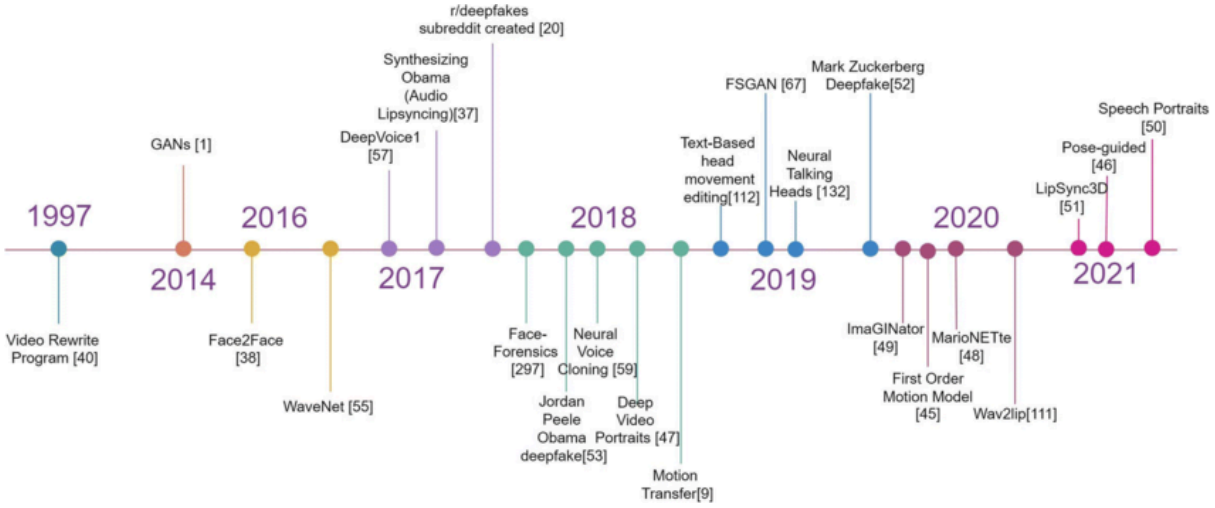


Figure 2: Timeline of the most significant events in the advancement of deep fakes (image source [23])

The generation of deep fakes is majorly used for face swapping in videos [2] which can detected with the following approaches[2]:

A. Visual Feature-based Deepfake Detection
B. Local Feature-based Deepfake Detection
C. Deep Feature-based Deepfake Detection
D. Temporal Feature-based Deepfake Detection

Figure 3: Face swap example [23]

In today's world, we have many tools for deep fake creation such as 'Deepfakes-FaceSwap', ' FaceSwap', 'dfaker', 'FaceSwap-GAN'and 'Face2Face' etc [5] Such apps can create deep fake content using the neural network architecture

called as 'Deep Autoencoder' that has two parts, namely encoder and decoder. When the encoder receives an image, it is converted into a latent vector that can be converted back into the original image by the decoder. This is done under conditions where there is little difference between the original picture and its reconstruction.

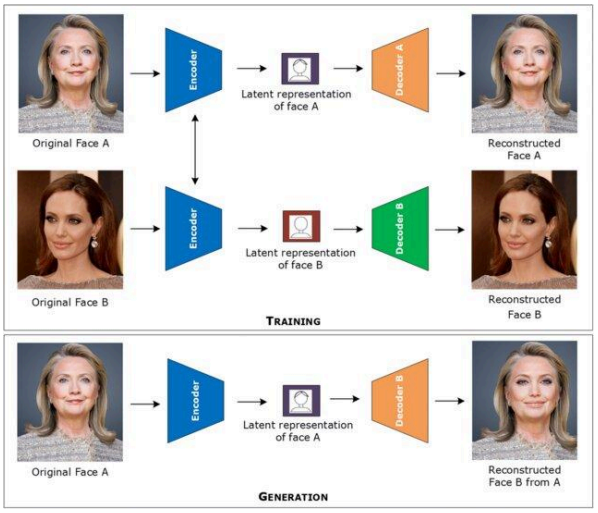


Figure 4: Generation using Deep Autoencoder [23]

As you can see the Figure 3 how it would work however there are multiple limitations in current detection techniques and how they impact the current field are the following down below:
Current detection tools have a hard time generalizing toward unseen data and they only do well on the datasets that they were trained on. In contrast, our proposed model provides a probabilistic approach that can better handle uncertainty and variability in the data. Adversary assaults can deceive deepfake recognition systems as insignificant changes to the input data can mislead them. The Bayesian framework is a tool that can help determine uncertainty and allow for the detection of adversarial examples. With the inclusion of GANs, it becomes possible to train the models with adversary instances thus making them resistant to these kinds of assaults. As the techniques for creating deepfakes continue to improve, the models used for identifying them must be developed rapidly enough that they can detect new kinds of forgery. The generative component (GAN) allows for new kinds of deepfake simulation, adapting detection systems immediately to new forgery types. Additionally, a Bayesian approach provides a framework for incorporating new data into the model in an organized manner.

Contributions of this paper:

The major contribution of this research is a novel neural network architecture in the deepfake detection study to solve the above-mentioned limitations so we can avoid the majorly impactful consequences of deepfakes in various areas.

This paper discusses deep learning models used for deepfake detection and in section 2 we discuss related work. The research methodology is discussed in section 3 and it also discusses the design components for the proposed novel deep learning framework in section 3 The implementation of this research is discussed in section 4. Section 5 presents and discusses the evaluation results. Section 6 concludes the research and discusses future work.

2. Related Work

In this section, we will focus on different state-of-the-art deep fake detection advancements contributed by various researchers, evaluate the efficiency of their deep-learning architectural models, and critically analyze numerous hypotheses.

In 2018, Afchar et al proposed detecting facial video forgery by focusing on the mesoscopic features of an image by extracting frames with faces in videos using the Viola-Jones object detection framework which is not as effective as compared to CNN, and then implemented the MesoNet and MesoInception-4 model which demonstrated high detection rates, with over 98% accuracy for Deepfake dataset and 95% for Face2Face dataset. Their idea was novel, however, the simplicity in the layers of the architecture could be better and unfortunately, the study does not contribute towards the detection of adversarial attacks on deep networks. In spite of its lightness, the Viola-Jones object detection framework does not have the same efficacy as modern CNN-based methods that are used for face detection. On the other hand, simple architectures of MesoNet may find it hard to detect advanced deepfakes; whereas, more complex ones might perform better in difficult datasets. Additionally, there is no defense against adversarial attacks in this study which is an important weakness in deep learning models. [1] At Purdue University in 2018, Guera and Delp used the HOHA dataset to implement CNN to extract feature frames and the use of a 2-node LSTM model for temporal sequence analysis on the frame. This method was speed efficient however the number of frames as input for the same video data could easily change the results creating a huge gap in the results every time. Inconsistent frame inputs affect model performance and result in instability in detection accuracy. Moreover, it implies a need for advanced models such as ConvLSTMs or transformers in order to improve its ability to generalize and accurately over a longer period of time. [8] Amerini, I., and Galteri, L., came up with a novel idea where they used an optical flow-based method to detect inter-frame dissimilarities and then ran two common neural networks. Optical flow represents a vector field determined based on two adjacent frames f(t) and f(t + 1) that help understand the visible movement between the observer and the scene. Specifically, they hypothesized that the optical flow can take advantage of differences in motion across synthetic frames compared to those original ones formed through a video camera. This experiment was implemented on the FaceForensics++ dataset which gave average results yet since it was a novel idea, grace must be given. Average performance has been obtained from faceforensics++ dataset using optical flow model which is the recent technique of measuring inter-frame divergences. But it can drift at some slight deceptions that cause deepfake making the need for better approaches. [9]

In 2020, DeepfakeStack, a deep ensemble learning technique was proposed by Rana and Sung and the first step in their method was to use 7 DL models and then use stack classifier this resulted in 1 AUCROC and 99% accuracy but using a large number of models will require immense processing which is not applicable for daily detection tools. Moreover, using seven deep learning models and a stack classifier is expensive in terms of computation but this enables us to attain great accuracy. Therefore in terms of real-time detection and low-resource environments, it becomes inappropriate since it carries too much load. On the other hand, concerning the ensemble technique concerning scenarios that require large-scale deepfake detection as well as instant fraud identification, scalability must be addressed. [10] The research conducted by Worku Muluye [11] was based on calculating the frequency of eye blinks in a given video to classify it between real and fake. Firstly VGG16 and ResNet-50 for classifying features on UADFV datasets and for sequence learning LSTM was used. In the years ahead, techniques for the detection of deepfakes that are more scalable, accurate, trustworthy and multi-platform will be required for research and

development. Even though this was an interesting approach, it's not the best approach. By emphasizing eye blink frequency, the generalizability of the deepfake detection model may be restricted because it may not take into account other facial components including lip movements or minor signs, while its robustness may be less efficient in handling complicated videos.

In the study [12], they proposed to use 'Action-Recognition' and in their preprocessing, they used 'RetinaFace' even though it was not the best method but their comparison resulted in their R3D model outperforming I3D which is better at 'Action-Recognition'. The paper discusses suboptimal preprocessing (RetinaFace) and its limitations in facial detection, particularly for subtle manipulations in deepfakes, and its lack of focus on real-time detection, which is crucial for practical applications like live-stream monitoring. The paper [13] presents a Convolutional Vision Transformer (CViT), which blends in CNNs as well as Vision Transformers. In this case, CNN extracts features from facial images while using an attention mechanism to classify these features by means of a Vision Transformer. Vision transformers along with self-attention methods may come at a cost because of their plus size computing demands which can make it hard for them to function well in remote settings where they must work on limited resources over a short period of time. In addition, in situations where there are few examples or where the samples themselves differ widely in representation, using transformers on smaller datasets or with imbalanced classes can result in poor model quality. In 2021, Ismail and Elpeltagy utilized YOLO for face extraction and used InceptionResNetV2 CNN to extract features from these faces at the same time. The features obtained from these faces are then passed on to the XGBoost which works as a recognizer at the uppermost level of CNN network. This experiment was performed on the CelebDF-FaceForencics++ (c23) merged dataset. However this paper has not mentioned real-time detection and anything around spoofing attacks thus we need more models. For deepfake detection scenarios, which usually involve live content validation, the model reflects the essential real-time detection aspect (Cristian, 2019). It does not take into account defense mechanisms against adversarial attacks or spoofing attempts usually associated with it. [14] In 2022, Chen and Li proposed a novel algorithm Xception-LSTM algorithm while using a spatiotemporal attention mechanism and convolutional long short-term memory (ConvLSTM). The algorithm has been built on a spatiotemporal attention mechanism, which consists of both spatial and temporal attention mechanisms for capturing and enhancing spatiotemporal correlations before dimension reduction of Xception. After that, ConvLSTM is presented to take into account the frame structure information when modeling temporal information. Unfortunately, this cannot keep up with the calculations within the frames but this is a great theory to build on in the future. The spatiotemporal attention mechanism and ConvLSTMs can be computationally expensive, making real-time deployment challenging. Additionally, the method faces challenges with frame sequence disruptions, affecting performance over long video inputs. [15]

The paper[16] presents a novel approach to deepfake prevention by embedding watermarks into video frames using an enhanced steganography technique based on GANs (Generative Adversarial Networks). The authors propose encoding watermarks into video features using an attention model trained on a 3D Convolutional Neural Network (CNN). This method aims to prevent deepfake creation by ensuring that any attempts to manipulate or alter the video frames result in the detection of inconsistencies in the watermarks. The GAN and CNN models necessitate extensive computational resources to be trained with this technique. The approach has been validated against certain datasets, limiting its efficacy to these data sets and possibly precluding its applicability to every other form of content or deepfake technology. Models like GANs and CNNs need a lot of computational power both during training and when making predictions. This may

limit their scalability and also make them dependent on particular datasets rendering them less effective across different kinds of deep fake data.

Using ConvNeXt [17] to extract features, Swin Transformer to process them, and Autoencoders and Variational Autoencoders for learning the latent data distribution along with capturing hidden patterns, the GenConViT model enhances deep fake detection. The model can detect subtle visual and latent features in deep fake videos and has addressed the gap of insufficient generalizability in previous deep fake detection models. Data scientists are optimizing their models for generalizability which makes them too intensive to be run on cheap devices. However, because it employs transformers and autoencoders, the model is computational heavy. This serves as a barrier to real-time and resource-constrained applications. It may constrain its real-time relevance because it uses sophisticated parts. The authors [18] proposed a fine-tuned ViT model that uses global feature extraction and self-attention mechanism and this model was also trained based on GAN-generated images. But it's biggest con is that the reliance of the ViT model on self-attention mechanisms and large-scale transformers could cost a lot in computation, which may put its utilization in real time or places with little resources at stake. The model's generalizability to unknown deepfake samples may be limited by overfitting to the training dataset, and its adaptation to dynamic environments may not adequately address catastrophic forgetting.

In deepfake detection is a new method proposed in the paper that consists of a GAN-CNN ensemble model with generative replay techniques to reduce catastrophic forgetting, one of the challenges in CNN models especially during continuous learning tasks. This method is similar to our proposed model. Therefore, one possibility for the decrease in accuracy is that the model might be overfitting to the training dataset; another is it may have difficulties generalizing to unknown samples. Dependence on blur and sharpness discrepancies is the central tenet of the method. As a result, it might not be applicable to all deepfakes, especially those created by sophisticated algorithms. Consequently, this means that it will yield unreliable detections of any new deepfake information from what it has been trained. [19] In a 2023, a blur inconsistency detection system was developed through an analysis of edge type and sharpness using MultiWavelet transform in this study. By this ability, it can evaluate if the facial region is hidden in the video or not. [18] Even though it has 93% this method would be unreliable for unseen data.

In 2024, Hasanaath and Luqman proposed the use of Frequency Enhanced Self-Blended Images (FSBI). The proposed methodology employs Discrete Wavelet Transforms (DWT), for purposes of extracting discriminative features from the self-blended images (SBI) so that they can be used for training a convolutional network architecture model. The SBIs blend an image with itself by introducing several forgery artifacts in a copy of the image before blending it, in order to avoid the classifier overfitting specific artifacts through learning more generic representations. Then, these blended images go into frequency features extractor, which detects those artifacts that cannot be easily detected in time domain but which are highly significant. This research can be a benchmark for further studies, however this had disruptive calculation in the frame sequence. As it analyzes sequentially arranged frames, the model faces some computational issues that could affect its performance in real-time. Moreover, since self-blended images are utilized as a means of introducing forgery artifacts, there is a possibility that due to artifact detection specificity in terms of overfitting may arise. [20]

The FFP-ChT (Facial Feature Points and Ch-Transformer) deepfake video detection model proposed by Rui Yang is made up of four components. To begin with, a number of frames from the input video are obtained and sent into the BlazeFace model for face detection. The outcomes of facial detection are sent to FaceMesh after which 468 facial feature points are taken

out. The feature point Re-Calibration technique is then used to reposition the extracted facial feature points. Finally, the facial feature points together with their corresponding displacement sequences are given for classification and forecasting purposes in this paper's designed Ch-Transformer. This leads us to combine the predictions on facial feature point displacement sequences with those on standard facial features so as to determine whether the original video is real or fake. [21] They could've chosen a better face-detection tool. Due to its complexity and potential constraints on scalability and real-time use, BlazeFace could not be the best choice for deepfake videos with minor face modifications rather than using it as a facial detection tool that can identify such subtleties effectively.

Detecting multiple classes of deep fake images using attention-based transformer models for visual imagery (ViTs) is a completely new approach that employs patch-based deep learning. The study [22] is concerned with more advanced methods of creating deep fakes such as stable diffusion and StyleGAN2 that can yield an F1 score of 99.90%. This study looks at how to deal with multiclass deepfake detection since deepfake generation methods have become more advanced, especially using ViTs to capture general characteristics like fishing. Even though it is effective in multi-class deepfake detection, the computationally intensive Vision Transformers (ViTs) approach is difficult to use in real-time detection systems and may need to be enhanced for future deepfake generation techniques such as stable diffusion. In such cases, the GAN-based CNN model can be further improved for example by cutting down computational costs especially when it is used in real-time situations that require immediate results.

To summarise our findings and more significant models, let's refer to the following table:

TABLE I: Summary of the different deep fake detection approaches in videos

Ref.	***Face detector***	***Dataset***	***Accuracy(%)***	***Description of the model***
Güera and Delp (2018)	CNN	HOHA dataset	97%	CNN + Lstm as sequence descriptor
Amerini and Galteri (2019)	CNN	FaceForensics++ dataset	75 %	Optical flow CNN + ResNet + VGG
Rana and Sung(2020)	7 models ensembled as base learners	FaceForensics++ dataset	99%	DeepfakeStack method
Worku Muluye (2020)	VGG16 and ResNet-50	UADFV datasets	93.20%	LSTM
Oscar Lima(2020)	NA	Celeb-DF dataset	82%	spatiotemporal convolutional methods
Solomon Atnafu(2021)	NA	DFDC	91 %	ViT + CNN
Ismail and Elpeltagy (2021)	YOLO	CelebDF-FaceForencics++ (c23) merged dataset	90.73%	CNN + XGBoost
Chen & Li (2022)	ConvLSTM	Multiple datasets	95%	Xception-LSTM
Preeti	GAN	Indian Actor	96.35%	GAN + CNN

Sharma(2024)		Images Dataset		
Hasanaath and Luqman (2024)	Discrete Wavelet Transforms (DWT)	FF++ & Celeb DF	95.49%	FSBI
R. Yang (2024)	BlazeFace	Multiple	92%	Ch-Transformer
Muhammad Asad (2024)	ViTs	Synthetic Faces High Quality (SFHQ)	70 %	Attention-based transformer models

To conclude the critical analysis of previous works in this area, let us discuss the gaps:
Many models (e.g., DeepfakeStack, CViT, Xception-LSTM) require significant computational resources which restrict their use in real-time applications. Therefore most of them are computationally inefficient. Some methods (e.g., CNN + 2-node LSTM, Blur Inconsistency Detection) produce inconsistent results based on different input frames and consequently have unreliable performance. It may be impossible for new deepfake techniques to profit from existing ones because of overfitting and dependency on certain datasets or artifacts (e.g., Worku Muluye's eyelid blink detection system, FSBI, GAN-CNN Ensemble). A major challenge in deepfake detection is posed by adversarial attacks and consequently many methods (e.g., MesoNet, InceptionResNetV2 + XGBoost) do not counter such attacks. The absence of real time detection in various models (such as InceptionResNetV2, Watermark-based techniques) can be characterized as detrimental to their practical utility.

Here is how we fill the above gaps with our model:
The last modification that inspired my model was simply just Bayesian CNNs. Bayesian convolutional neural networks (CNNs) offer techniques for estimating uncertainty by treating the weights as probability distributions instead of fixed values. This enables the model to capture epistemic uncertainty (model uncertainty), which is especially important for deepfake detection since such a model should be strong and dependable. In deepfake detection, it is important to estimate uncertainty when classifying video frames or images into real or fake ones. The uncertainty scores can help in identifying difficult or ambiguous instances where the model is less confident so that a warning is given when there is need for further analyses. Interpretability: Bayesian CNNs provide a measure of how much the model "trusts" its predictions, which may lead to more interpretable deepfake detection models, an essential factor in sensitive applications such as content moderation or legal environments.

Advantages of adding GAN:
Generated Adversarial Networks (GANs) are expert in generating top-notch artificial data. In a CNN-GAN framework, the discriminator (CNN) can be trained to differentiate between original and synthetic images while adversarial examples generated by the generator aid in improving it over time. The use of GANs together with Bayesian CNNs enables the discriminator to enjoy both robust learning through adversarial training and understanding uncertainty through bayesian inference. With such an arrangement, it becomes possible for models to note even slight bogus traces existing on deep fakes as long as generators advance at making them look like real things. In a Bayesian CNN-GAN framework, the GAN component serves as a form of regularization to the Bayesian CNN by producing difficult examples that aid in better generalization and prevent overfitting. This

is specifically significant when subtle alterations are made within deepfakes making them harder to identify. By exposing the model to different kinds of modifications, real-world deepfakes which can range in quality and style can be addressed using chain training from GAN within bayesian CNN. The GAN introduces adversarial examples that mimic the real-world variations of deepfakes. This collaboration results in a more generalizable model for deepfake detection.

Combining Bayesian CNN with GAN Novelty in Deepfake Identification:

The Probabilistic Discriminative Capability: The application of Bayesian CNN enables a probabilistic interpretation of classification decisions which increases their trustworthiness and reliability in critical fields. Adaptive Learning: GANs create synthetic deepfakes which serve as a continuous challenge to the Bayesian CNN thereby enhancing its capacity to identify more intricate and high-quality deepfakes. Better Generalization: A fusion between uncertainty estimation and adversarial training can enhance the generalization ability of a model against new or rare deepfake techniques taking into account the ever-changing deepfake technology.

The Bayesian CNN-GAN model serves as a new method for the detection of deepfakes, addressing a number of major gaps. It reduces entire computational burden by amalgamating uncertainty into one model thus handling inefficiency from computation. The generation of sample, which forms an important role in GANs' operations, is carried out during training through synthetic example creation eliminating the need for some many pre-preparations of data or different kinds of input configurations. Bayesian CNNs also offer a probabilistic interpretation hence providing more consistent results across diverse types of deepfakes. A greater variety of manipulations allows the model to simulate deepfake variations and train on them making it less sensitive to small changes in the input frames. In the face of unknown patterns, unfamiliar deepfakes are taken care of by adjusting predictions based on increased uncertainty levels within Bayesian CNNs. This helps avoid overfitting as it enables continuous learning through generating new sample examples against changing deepfake techniques from time to time. Adversarial attacks are harder against Bayesian networks because they involve estimating uncertainty about their predictions (which is inherently larger than zero). Moreover, during the training phase such kind of model can generate adversarial deepfakes thereby allowing it recognize both types including those that are crafted intentionally to deceive their victims and ordinary ones at the same time. By integrating uncertainty estimation into the CNN directly, running GAN for generating training images at once allows Bayesian CNN-GAN model to achieve real time detection capability.

3. Methodology

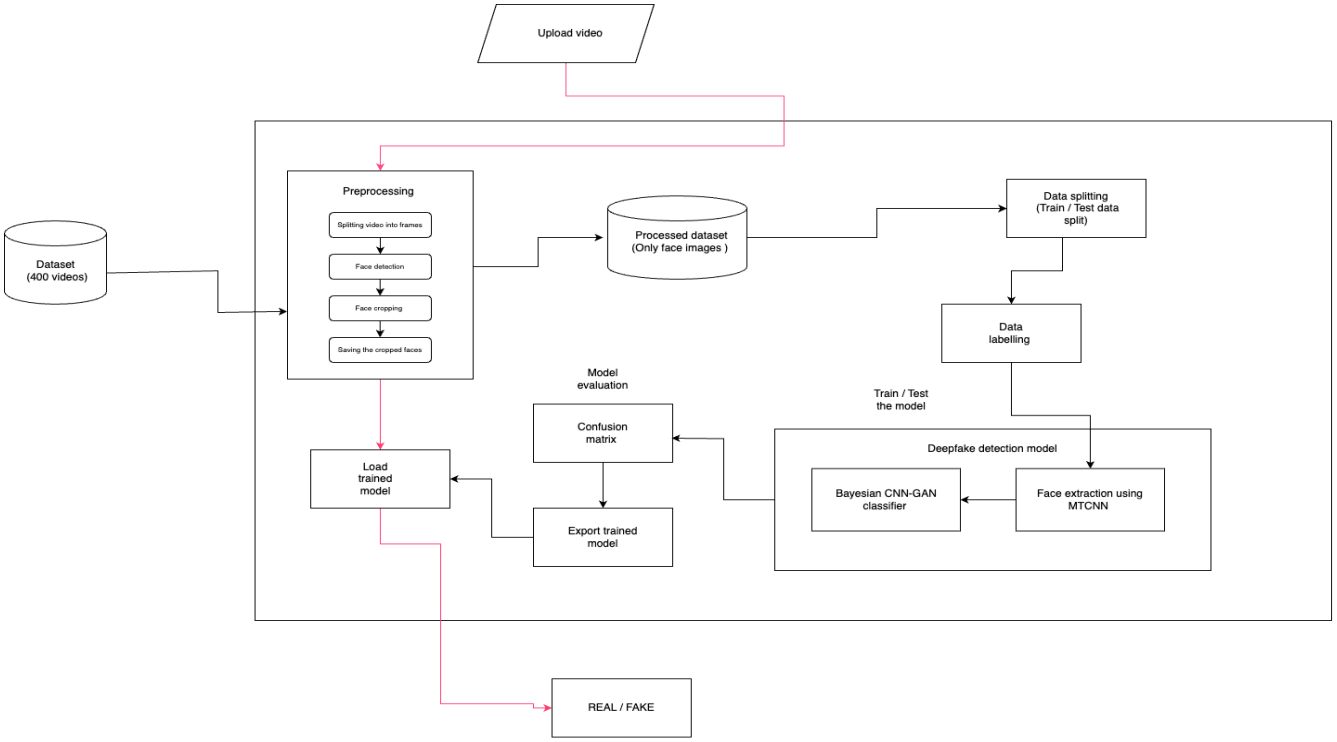
This section discusses the research methodology and the preprocessing before the implementation of our proposed model.

3.1 Data description:

For our experiment, we have chosen the 'DFDC' dataset from Kaggle which a bunch of videos with a mix of original and deepfakes. There are two files, one with 400 videos and the other one has 802 files and we chose the one with 400 videos to save memory and it comes with train data and test data split and a CSV file and lastly a metadata JSON file.

3.2 System architecture

Following is a diagram of the system architecture of how our experiment executes:



3.3 Preprocessing

For preprocessing the data given, firstly we focus on dataset features for which we created a plotting function. After this step, we used cv2 to take in the video path as the input and perform a video capture and then read these images. Then we have the third function where we extract and compare different images from the videos. For this function we will take two inputs i.e video path list and the folder with all the videos. In this process, we will perform the capture and then the function reads the first frame by opening the file, checking if it was successfully read, convert frame from BGR to RGB (matplotlib expects RGB format). Now that we have the frames, we have the next function to detect face features from the image like identify frontal face, eyes, smile and profile face and display the detected objects over the image. We hae used MTCNN and pretrained haarcascades to extract the objects in the image frames and for a processed frame we put a red box around it to show the user. There is a huge need for a folder for all the generated frames and save paths, we shall do so with the proved 'metadata' file in the training video folder. Initialising 'dlib.get_frontal_face_detector()' for the face and the convert it to PIL to create an array of classified paths and save them in a file called 'real' or 'fake' based on their characteristics. After processing all 400 videos, we have the following stats that there were 25% real images and 75% fake images. We even displayed a few examples by labelling the frames which are zoomed on the face. After finishing with the preprocessing, we move to implement the neural networks.

4. Implementation

In this section, we will discuss the details of all the models used in the research experiment and their different layers and why they were chosen. The following are the models implemented:

4.1 XceptionNet

We have used the pre-trained XceptionNet from Keras and it is deep CNN used for feature extraction making it a great choice as it has also been a great benchmark for other research studies in this topic. For our model, we also imported 'ImageDataGenerator' to generate batches of tensor images in real time. In order to match the expected input for this model, the code loads a pre-trained Xception model on ImageNet, without its top classification layer, and sets the input shape to (224, 224, 3). To bring down the dimension, it passes through a GlobalAveragePooling2D layer. One can set up high-level feature learning by hooking in a dense layer with 1024 units and using ReLU as the non-linearity. Finally, a dense layer with only one unit is added and its output is activated using sigmoid so that we can do binary classification (0 or 1 corresponding to real or fake). Using the Adam optimizer, binary cross-entropy loss function (which is appropriate for binary classification), and accuracy for evaluation are features of the compiled model. Batches of image information can be generated using an image data generator. This will reduce pixel values to a range from -1 to 1 by normalizing them with the factor of 1/255. Using the 'flow_from_directory' method, we created 'train_generator' and 'validation_generator', which read images from the directory, resize them to '224x224' pixels and assign labels according to the corresponding folder structures. These generators will later serve as data feeders for training and validation purposes. For this particular model evaluation on validation datasets, the evaluate function is used; during this process, both validation loss and accuracy are computed before being outputted to standard output.

4.2 EfficientNet

We have implemented the pretrained EfficientNet as our second model. The EfficientNetB0 model is pre-trained with weights from ImageNet. By including the "include_top=False" parameter, one can exclude the top-most fully connected layers meant for the ImageNet classification problem, thus rendering it appropriate for transfer learning to another task. The input shape has been set at (224, 224, 3) in such a way that it satisfies the dimensionality requirements of EfficientNetB0. We used the following three custom layers in the architecture: 1) GlobalAveragePooling2D: This layer is important in reducing size of data received from a main model, thus boosting the presence of features in the whole photo. 2) Dense Layer with 1024 Units: In this task to identify deepfake, a fully connected layer having 1024 neurons along with ReLU activation is included so as to learn more about specific complex characteristics related to the deepfake detection process. 3) Output Layer: A dense layer having only one neuron and sigmoid activation produces a probability score that indicates if an image represents either authenticity or fakeness (binary classification). We used the Adam optimizer as the optimizer for speed and Binary cross-entropy is chosen as the loss function since the task is binary classification.

4.3 Proposed model i.e Bayesian CNN-GAN

We import 'Matplotlib' to help visualise the generated images from the GAN. The Bayesian CNN model has the following layers: it involves Conv2D, MaxPooling2D, Flatten, and Dense layers.l2 regularization is used on each layer to avoid overfitting. Dropout: In the course of training, a proportion of the input units are set equal to zero at random in order to lessen the chances of overfitting. To perform binary classification tasks (outputting probabilities for two classes), the output layer employs two neurons with softmax activation function. Then we have the second half which is the generator model for GAN: This is made to take all the noise and act to generate synthetic images. The dense layer receives the input noise and converts it into an image format. The Conv2DTranspose layers are used to increase the dimensions of the image by shrinking it. During

training of generator, this activation function allows a small gradient when the input is negative, which helps solve the vanishing gradient problem. The final layer outputs image comprised of 3 channels (RGB) using tanh activation function, to keep pixel values within [-1, 1]. For the third half of this hybrid, we have the discriminator model which distinguishes between real or fake. We use three layers here: Conv2D Layers: They are applied in order to downsample input image and extract features. LeakyReLU: They serve as the activation function for convolutional layers. Dense Layer: A dense layer with a sigmoid activation function outputs a probability score indicating whether the image is real or fake. Then we have the GAN class: The 'Keras.Model' class is extended by this custom class that combines the generator with the discriminator into a GAN. The 'train_step method' implements a customized training loop for the GAN. It generates phony images, calculates losses for both the generator and discriminator, and updates the model weights by applying gradients. The Adam optimization method and the categorical cross-entropy loss function (which is appropriate for multi-class classification tasks) are used to compile the Bayesian CNN. A binary cross-entropy loss function is employed in training the GAN with different optimizers for both generator and discriminator. The Bayesian CNN is trained on a dataset (X_train, Y_train) with validation data (X_val, Y_val) for 10 epochs. On the other hand, the GAN is trained on the same dataset (X_train) but in another context concentrating more on generating non-existing pictures than classifying them. To conclude why I proposed this model: The main emphasis of the code for the Bayesian CNN is image classification with advanced techniques added to control overfitting. Secondly, GAN can be referred to as Generating Adversarial Networks where two models compete against one another (i.e., generator & discriminator) and learn through their interactions.

5. Evaluation and results

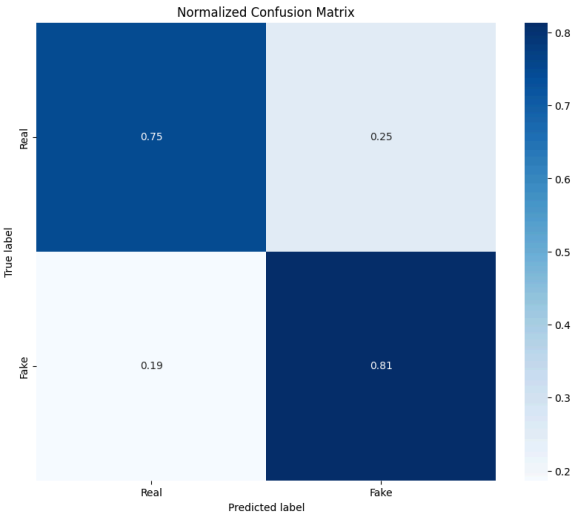
In this section, we evaluate all three models and for the first model i.e. the XceptionNet, the accuracy was 0.9828 and the loss was 0.5585. The process executes in the following way: The built-in function called 'evaluate()' is used for computing the loss and accuracy of the model on the supplied 'validation_generator' which is likely to give batches of validation data that comprise both features and labels. The evaluate method returns a list with: The first element is the loss incurred while running the validation set; and secondly, its correct classification rate in that same set. Step performance: 4s/step - accuracy: 0.9828 - loss: 0.5585 indicates that during evaluation, the model achieved an accuracy of 0.9828 and a loss of 0.5585 per step (batch). Validation Loss and Accuracy: The two print statements display: Validation Loss: 0.5599 indicates that the total average loss over the entire validation set was approximately 0.5599. Validation Accuracy: 0.9733 indicates that the overall accuracy across the validation set was approximately 97.33%.

From the above, we can see that this model has good performance on the validation set with high accuracy but moderate loss value. Loss and accuracy are typically used to measure a model's performance for tuning purposes.

Next, we move on to the second model which is our EfficientNet model which resulted in 0.0013 for loss and with a validation accuracy of 1.0 which is pretty good. The model performance on the validation set is computed using evaluation = model.evaluate(validation_generator). The validation loss can be found in evaluation[0] while evaluation[1] contains the validation accuracy. The following information is presented for each epoch: Training Accuracy and Loss: In each of the epochs, the accuracy starts very low at first epoch (0.7293 or 72.93%) while it reaches 100%

accuracy in the latter epochs. The loss values are also relatively high during the first epoch (0.2866), but they decrease in substantial terms when training extends further. Validation Accuracy and Loss: The model has a consistent 100% validation accuracy from the first epoch to date. The validation loss starts as low as 0.0010 at first epoch and remains so for a while signifying good generalizability of the model. The validation loss for final epoch is 0.0013. After training: Final Validation Loss: 0 (this implies minimal error on validation set). Final Validation Accuracy: 1.0 (it means that predictions were perfect). Therefore, this model learned well since both trainings and validations are at 100%. It also shows low loss values indicating precise predictions especially on validation set; however whether overfitting actually occurs is still left open even in simple tasks or datatsets.

Finally, we discuss our proposed model, and to evaluate the main model we implemented a confusion matrix as you can see in the figure on the right. The accuracy is 0.8219 and the loss is 0.9150. After 10 epochs, the test accuracy is 0.80 which is great for a novel architectural hybrid. Following is our classification report for the main model:



	precision	*recall*	*f1-score*	*support*
Real	0.50	0.75	0.60	75
Fake	0.93	0.81	0.87	300
accuracy			0.80	375
macro avg	0.71	0.78	0.73	375
weighted avg	0.84	0.80	0.81	375

We conclude the evaluation with the overall accuracy being 80.00%.

6. Conclusions and Future Work

In summary, we took the DFDC sample dataset and preprocessed it to create frames out of videos, and used feature and object detention using pre-trained MTCNN then we implemented three models: XceptionNet, EfficientNet, and our proposed model i.e the hybrid Bayesian CNN-GAN which is a novel neural network architecture in the field of deep fake detection which resulted in 80% test accuracy. There is scope for improvement in the model for the coming future. The Bayesian CNN-GAN network meant for detecting deepfake records has up to an 80% accurate rate

which is highly preferred compared to other machine-learning models such as XceptionNet and EfficientNet. With the Bayesian inference incorporated in its functions, it is able to manage any ambiguous or malicious instance, thereby giving it a stronger force even though its present accuracy may be low. Account for uncertainties or variability in data that originally deterministic models like XceptionNet and EfficientNet may not be good at; hence, this increases the model's generalization capability. Another advantage of Bayesian CNN-GAN is its adaptability. The GAN part can build sophisticated and high-quality synthesized images of deepfakes that are hard to detect thereby enhancing the learning process of the whole system over time. However, with an accuracy rate of 80%, there are some shortcomings in order to surpass completely other existing models including XceptionNet and EfficientNet. Potential limitations include limited training data especially when dealing with rare or subtle deepfake examples. Future directions will introduce new strategies including: Transfer learning, Class balance methods among others such as Synthetic Data Generation overfitting Progressive GAN training Conditional GANS extraction features Network Depth etc., Computer Capacity Pruning Techniques among others too.

Data augmentation aims to diversify and balance the dataset for improved generalization. This means that conditional GANs and progressive training are used in the training of GANs. Hybrid architectures combine advanced feature extractors with Bayesian uncertainty estimation, regularization involves dropout, Bayesian ensembling and cross-validation in order to reduce overfitting. Variational inference or Monte Carlo dropout might help to achieve this problem by providing a more efficient approximation of the Bayesian posterior.

These suggestions could aid in overcoming the current constraints of the Bayesian CNN-GAN model and enhance its present 80% prediction accuracy so that it could even outperform XceptionNet, EfficientNet and other models in terms accuracy as well as interpretability.

References

[1] Afchar, D., Nozick, V., Yamagishi, J. and Echizen, I. (2018). MesoNet: a Compact Facial Video Forgery Detection Network. 2018 IEEE International Workshop on Information Forensics and Security (WIFS). doi:https://doi.org/10.1109/wifs.2018.8630761.

[2] Ramadhani, K.N. and Munir, R. (2020). A Comparative Study of Deepfake Video Detection Method. [online] IEEE Xplore. doi:https://doi.org/10.1109/ICOIACT50329.2020.9331963.

[3] Show, S. (2024). Rise of DeepFake Technology - Legamart. [online] legamart.com. Available at: https://legamart.com/articles/deepfake-technology/.

[4] realitydefender.com. (n.d.). A Brief History of Deepfakes — Reality Defender. [online] Available at: https://www.realitydefender.com/blog/history-of-deepfakes.

[5] Zhang, D. (2024). Daisy-Zhang/Awesome-Deepfakes. [online] GitHub. Available at: https://github.com/Daisy-Zhang/Awesome-Deepfakes [Accessed 6 Jul. 2024].

[6] A. Heidari, Nima Jafari Navimipour, H. Dag, and M. Unal, "Deepfake detection using deep learning methods: A systematic and comprehensive review," Wiley interdisciplinary reviews. Data mining and knowledge discovery/Wiley interdisciplinary reviews. Data mining and knowledge discovery, Nov. 2023, doi: https://doi.org/10.1002/widm.1520.

[7] Ensemble of Machine Learning Classifiers for Detecting Deepfake Videos using Deep Feature. (n.d.). Available at: https://www.iaeng.org/IJCS/issues_v50/issue_4/IJCS_50_4_15.pdf [Accessed 21 Mar. 2024].

[8] Guera, D. and Delp, E.J. (2018). Deepfake Video Detection Using Recurrent Neural Networks. 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). doi:https://doi.org/10.1109/avss.2018.8639163.

[9] Amerini, I., Galteri, L., Caldelli, R. and Del Bimbo, A. (2020). Deepfake Video Detection through Optical Flow Based CNN. [online] IEEE Xplore . Available at: https://ieeexplore.ieee.org/document/9022558.

[10] Rana, Md.S. and Sung, A.H. (2020). DeepfakeStack: A Deep Ensemble-based Learning Technique for Deepfake Detection. 2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom). doi:https://doi.org/10.1109/cscloud-edgecom49738.2020.00021.

[11] The Deepfake Challenges and Deepfake Video Detection. (2020). International Journal of Innovative Technology and Exploring Engineering, 9(6), pp.789–796. doi:https://doi.org/10.35940/ijitee.e2779.049620.

[12] De Lima, O., Franklin, S., Basu, S., Karwoski, B. and George, A. (n.d.). Deepfake Detection using Spatiotemporal Convolutional Networks. [online] Available at: https://arxiv.org/pdf/2006.14749 [Accessed 8 Aug. 2024].

[13] Wodajo, D. and Atnafu, S. (2021). Deepfake Video Detection Using Convolutional Vision Transformer. [online] Available at: https://arxiv.org/pdf/2102.11126 [Accessed 8 Aug. 2024].

[14] Ismail, A., Elpeltagy, M., S. Zaki, M. and Eldahshan, K. (2021). A New Deep Learning-Based Methodology for Video Deepfake Detection Using XGBoost. Sensors, 21(16), p.5413. doi:https://doi.org/10.3390/s21165413.

[15] Chen, B., Li, T. and Ding, W. (2022). Detecting deepfake videos based on spatiotemporal attention and convolutional LSTM. [online] ScienceDirect. Available at: https://doi.org/10.1016/j.ins.2022.04.014.

[16] Noreen, I., Muneer, M.S. and Gillani, S. (2022). Deepfake attack prevention using steganography GANs. PeerJ Computer Science, 8, p.e1125. doi:https://doi.org/10.7717/peerj-cs.1125.

[17] Wodajo, D., Atnafu, S. and Akhtar, Z. (n.d.). Deepfake Video Detection Using Generative Convolutional Vision Transformer. [online] Available at: https://arxiv.org/pdf/2307.07036 [Accessed 9 Aug. 2024].

[18] Saadi Mohammed Saadi and Ameen, W. (2023). Proposed DeepFake Detection Method Using Multiwavelet Transform. International Journal of Innovative Computing, 13(1-2), pp.61–66. doi:https://doi.org/10.11113/ijic.v13n1-2.420.

[19] Sharma, P., Kumar, M. and Hitesh Kumar Sharma (2024). GAN-CNN Ensemble: A Robust Deepfake Detection Model of Social Media Images Using Minimized Catastrophic Forgetting and Generative Replay Technique. Procedia computer science, 235, pp.948–960. doi:https://doi.org/10.1016/j.procs.2024.04.090.

[20] Hasanaath, A., Luqman, H., Katib, R. and Anwar, S. (n.d.). FSBI: Deepfakes Detection with Frequency Enhanced Self-Blended Images. [online] Available at: https://arxiv.org/pdf/2406.08625 [Accessed 9 Aug. 2024].

[21] Yang, R., Lan, R., Deng, Z., Luo, X. and Sun, X. (2024). Deepfake Video Detection Using Facial Feature Points and Ch-Transformer. ACM transactions on multimedia computing, communications and applications/ACM transactions on multimedia computing communications and applications. doi:https://doi.org/10.1145/3672566.

[22] Muhammad Asad Arshed, Mumtaz, S., Ibrahim, M., Dewi, C., Tanveer, M. and Ahmed, S. (2024). Multiclass AI-Generated Deepfake Face Detection Using Patch-Wise Deep Learning Model. Computers, 13(1), pp.31–31. doi:https://doi.org/10.3390/computers13010031.

[23] Masood, M., Nawaz, M., Malik, K.M., Javed, A., Irtaza, A. and Malik, H. (2022). Deepfakes Generation and Detection: State-of-the-art, Open Challenges, Countermeasures, and Way Forward. Applied Intelligence, 53(4). doi:https://doi.org/10.1007/s10489-022-03766-z.