Group Project Report
Fake Face Detection: FaceAuth

Group-01
Anjali Mudgal
Arjun Bingly
Udbhav Kush

DATS-6303
Deep-Learning
Dr. Amir Jafari

December 8, 2023

Contents

Abstract

The group project report titled "Fake Face Detection: FaceAuth" presents the efforts of Group-01, consisting of Anjali Mudgal, Arjun Bingly, and Udbhav Kush, in developing an image classification model for distinguishing between real human faces and those generated by AI. The project focuses on using various datasets, including Celeb-HQ-2 and Wiki Celeb for real faces, and 1 million fake faces, iFakeFaceDB, and DeepFakeFace for fake faces. The modeling approaches explored include GANs, CNNs (specifically DenseNet), and transformers (ViT and DeiT).

The report outlines the training process, results, and evaluation metrics for each model, with a particular emphasis on the performance of DenseNet, ViT, and DeiT. The t-SNE plots are presented to visualize the clustering of latent representations, and an interactive web application using Streamlit is developed for real-time image classification. The app allows users to upload an image, and the model provides probabilities of the image being real or AI-generated, along with indications of the likely AI model used for generation.

1 Introduction

In the era of advanced artificial intelligence, the ability to discern between human-generated and AI- generated images has become crucial, this is especially true for AI generated human faces. This project aims to develop a image classification model capable of distinguishing between faces of real humans and those generated by AI.
Additionally, we hope to develop an auxiliary classification model that will be extended to identify the specific AI architecture responsible for generating the image, including state-of-the-art models like DALL-E, Imagen, and others.

We undertook a fairly deep literature review to identify the current state-of the art solution to this problem but we could not find any such model. Although, we did find commercial solutions that claim to be able to detect Deep-Fakes like sensity.ai but they do not publish the kind of models they are using neither can their effectiveness be evaluated by us.

2 Data

Even though there exists various data sources on Kaggle and other sources we wanted to make sure that the sources we choose explicitly tell us what models were used to generate these images, moreover we wanted atleast 20k images from each source to make sure we have enough data. Moreover, since we are classifying between fake and real we need both fake and real human faces.

2.1 Datasets Used

For real faces we used:

1. Celeb-HQ-2
2. Wiki Celeb

There are fairly clean datasets both scraped from the internet and contain mostly celebrity photos, this gives us some surety that the collection of these datasets did not involve any violation of privacy. This was also the reason we did not end up using the Flicker dataset, even thought the data publisher ensures that the images were extracted legally. Moreover, both these datasets contain high-resolution color images but since training on high-res images would be very resource consuming we decided to settle on a resolution of 256x256 pixels and all these images were cropped to face so we got away by not doing much pre-processing.
For the fake faces we used datasets:

1. 1millon fake faces: This dataset contains a millon fake faces generated by the StyleGan model developed by Nvidia.

2. iFakeFaceDB: This dataset is a collection of 87000 images which are again generated by the styleGan model but are transformed with the GANprintR approach. GANprintR is a tranformation to remove the GAN fingerprint from the images (Refer to paper for more details).

3. DeepFakeFace: This dataset contains fake face images which are generated by diffusion based models. This dataset contains images generated by Stable Diffusion Impainting model, Insightface toolbox and Stable Diffusion V1.5.

These datasets were chosen as we thought they encompass both GAN and Diffusion based models.

2.2 Splits

We did a standard split of training-test and dev. The training set was used for training the model and the test set was used to make decisions while training the model. The dev set was untouched and only evaluated on right before the project presentation. After splitting the dataset in training, test and dev, we had 120,000 images to train our model.
Moreover we ensured the following conditions:

- Equal number of fake and real images.
- If more than one fake dataset was used to train the model they have equal number of images from each fake dataset.
- The Dev set is not used until the day of submission to get the results.

3 Modeling

3.1 GAN

Our initial idea was to first find the state-of-the-art GAN models that generates realistic human faces and the to use the Discriminator section of the model to detect fake faces from the real one. This was clearly a very naive approach, upon reading the literature about training GAN based models we realised that usually a GAN is trained to a saddle point where the Discriminator can no longer differentiate between the real and generated images, this by itself would make the discriminator not so useful for our purposes. But we held onto the idea as we thought we could further train the discriminator to do our job.
We landed on two GAN models namely StyleGAN3 and StyleSwin but both these models were very challenging to work with since we could not dissect the published code to use the discriminator for our purpose. Hence after much delibratoin, we made a team decision to use other models due to the time constrain of the project.

3.2 CNN

Since we could not find any state-of-the-art models from the literature review, we decided to use a CNN based model since we knew that it would perform fairly well. The most popular CNN based model was ResNet (Residual Neural Network), (The ResNET paper) this model includes skip-connections. A skip-connection connects activations of a layer to further layers by skipping some layers in between. This helps combat the problem of the vanishing/exploding gradient. Therefore, allowing for much deeper networks and such an architecture has shown really good results on datasets like ImageNet-1k. So we went a step further and choose to use a DenseNet model, this model takes the idea of skip-connection and densely connects all layers within the dense block. This allows maximal information flow. Refer: DenseNet paper

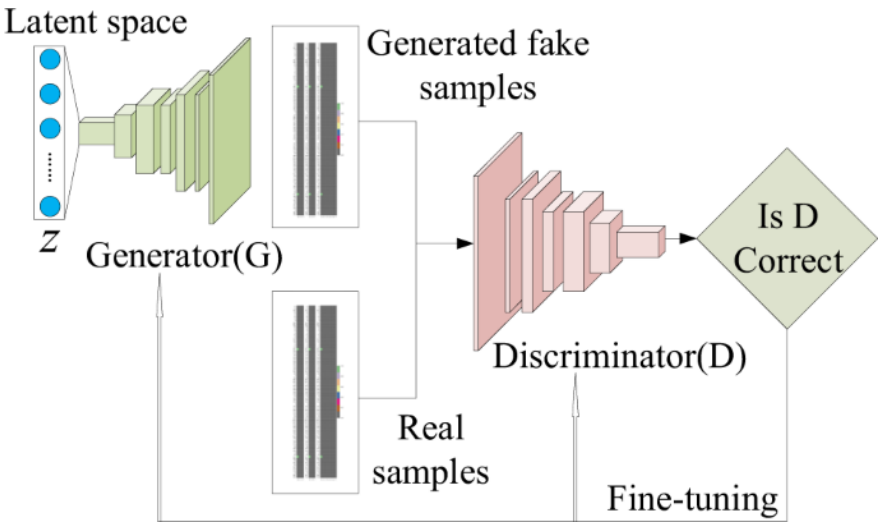

Figure 1: Basic Generative Adversarial Network (GAN) architecture

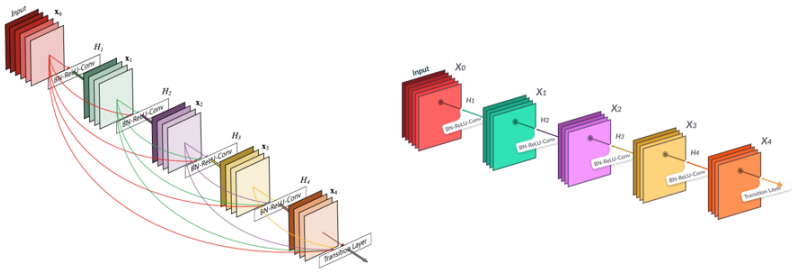

Figure 2: Architecture difference between ResNet and DenseNet

3.3 Transformer

ViT - Vision Transformer

During our literature review, we found one paper. The model gave us understanding of the Vision Transformer and its architecture. On further research, we found a pre-trained transformer model called vit-base-patch16-224 (https://huggingface.co/google/vit-base-patch16-224). The model was the most trending model on hugging face for the image classification task. On further studying the architecture, we found that the model has a transformer base with a linear layer as the classification head. We trained this model on our dataset and the model gave very good results on our test set.

Breaking down the architecture of the Google's ViT, the image is fed into the model of fixed-size patches of 16x16 and are linearly embedded. The linear embedding of the image along with positional embedding of the image are fed to the transformer encoder. The transformer head is connected to the MLP head to classify images into different classes. Since, the model is a transformer based model, we use the concept of positional embedding (like it is done for textual data) along with the [CLS] token which is taken as the image representation.

In the code above, we define the processor object of the model. This processor takes care of all the preprocessing required for the image. We just need to pass the image to the processor object. Linear embedding, positional embedding, and [CLS] token, all are taken care of to pass to the model to inference and train our model.

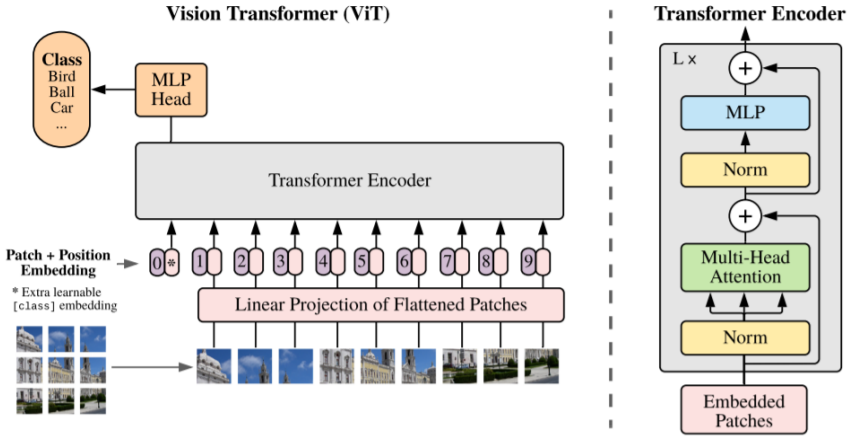

Figure 3: Architecture of Vision Transformers

DEIT - Data Efficient Image Transformer

DeiT stands for Distilled data-efficient Image Transformers. This model is an image transformer with a very similar architecture to the Google's ViT model discussed above with just an addition of a distillation layer in DeiT models. In this model too, images are presented to the model with a fixed-size patch (16x16) linearly embedded that are linearly embedded. From the paper we could understand that the distillation process requires the distillation token. It learns from a teacher model (CNN) using a distillation token in addition to the usual class ([CLS]) token. The distillation token learns by paying attention to both the class token and patch tokens in the image during training.

3.4 Training

As mentioned in the Data section of the report, we collected fake images dataset from various sources that were iFakeFaceDB, deepFakeFace and diffusion based model generated images. We trained our model with the combination of all the fake images i.e, training set had images from iFakeFaceDB set, deepFakeFace set and diffusion models generated images. Other than this, we also trained our model on each of these specific datasets. The thought process behind this was the know how biased a model is getting to one particular dataset if the model is given exclusively those pictures or how well can the model generalize with just a single source of images.

3.5 Results

DenseNet

Densenet was the only CNN based model we trained on our dataset. The model accuracy was decent but densenet's performance was subpar as compared to the vision transformer models we tried. The thing to note here is that densenet model trained on exclusively trained on diffusion, GAN, and GANPrintR dataset are not performing very well on the dev set. The reason for this that the model gets biased if just trained on type of images and does not generalize well.

Model	Accuracy	Precision	Recall	AUROC	F1Score
Densenet	0.78239	0.86374	0.70928	0.7883	0.77892
Densenet_diffusion	0.59407	0.57277	0.9797	0.5601	0.72291
Densenet_GAN	0.54048	0.54054	0.99883	0.5001	0.70146
Densenet_GANPrintR	0.66416	0.61676	1	0.63457	0.76296

Table 1: DenseNet model results

DEIT

Model	Accuracy	Precision	Recall	AUROC	F1Score
deit	0.83607	0.82924	0.87737	0.83243	0.85262
deit_iFakeFaceDB	0.66425	0.61683	1	0.63467	0.76301

Table 2: DEIT model results

The DeiT model gave decent metrics on our dev set. As we can see that accuracy is around 83%. The only problem we had while training this model was that the training time for this model was high as compared to other models we trained.

ViT

Model	Accuracy	Precision	Recall	AUROC	F1Score
ViT	0.83616	0.84791	0.84919	0.83501	0.84885
ViT_diffusion	0.75791	0.75311	0.82134	0.75232	0.78575
ViT_GAN	0.56116	0.55192	0.9995	0.52254	0.71115
ViT_GANPrintR	0.66407	0.6167	1	0.63447	0.76291

Table 3: ViT model results

Google's ViT model gave us the best metrics compared to all other models. Google ViT trained on just diffusion datasets gave decent results too. Google ViT generalized the best among all the models tried for this project.

4 t-SNE

t-SNE stands for t-Distributed Stochastic Neighbor Embedding. It is a non-linear dimensionality reduction technique. It aims to find a low-dimensional representation of the data (usually 2D or 3D) while preserving local structure and relationships between data points.

In most of the research paper that we have seen especially this https://arxiv.org/pdf/2302.10174.pdf Two main points were made by this paper which offered us some insights and the way we can visualize the latent vector approach:
When they observed the tSNE plots for a GAN based images trained classifier the cluster for real images was more open and the cluster for GAN generated images was more compressed. Also the images that were trained on diffusion generated images they were falling in the same cluster as the real images. Which gave insight into this might be because the model is learning the fingerprint of fake images but not the real images. f can easily distinguish FGAN from the other three, but the learned real class does not seem to have any property (a space) of its own, but is rather used by f to form a sink class, which hosts anything that is not FGAN.

To investigate the clusters of latent images, we made the tSNE plots for the embedded images. We removed the last layer of our trained transformer, then passed 50 set of images from each class to it. The

results are shown in the below figures.

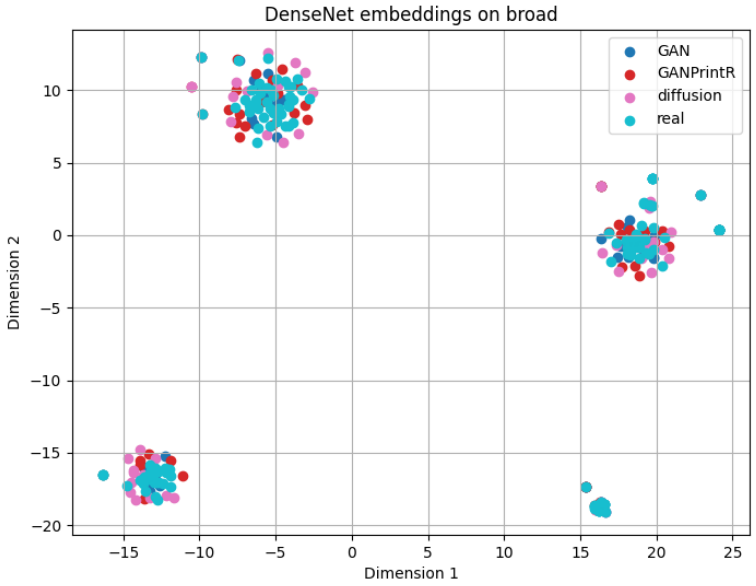

Figure 4: t-SNE plot of DenseNet model trained on all images.

For DenseNet (*Figure:4*), since it is convolutional based we had to perform an average pooling after removing the last layer and the dimension size was 1052. The results were not what we expected in this case since the real class points are overlapping everywhere. This might be because of loss of information while pooling or while reducing the 1058 dimensional space to 2 dimension. This might have been the reason the results in conclusion are not inline with the plot we have.

ViT network trained on combination of images performed the best, As seen in *Figure:5* it is able to form the clusters for diffusion and GANprintR. Some of the real images are still merging with the GAN image clusters which was slightly unexpected. From *Figure:6* we can see that the GAN images are able to be in a completely different cluster , but the model also understands GANprintR and diffusion. From *Figure:7* we can see that the GANprintrR images are able to be in a completely different cluster , but the distinction of other 2 with real images might still be difficult. Which is why we have the worst results on this one.

So our best model is when the dataset with multiple ai generate images is used and then the latent representation of images from transformer is used, enabling the model to learn some representations of real image cluster as well.

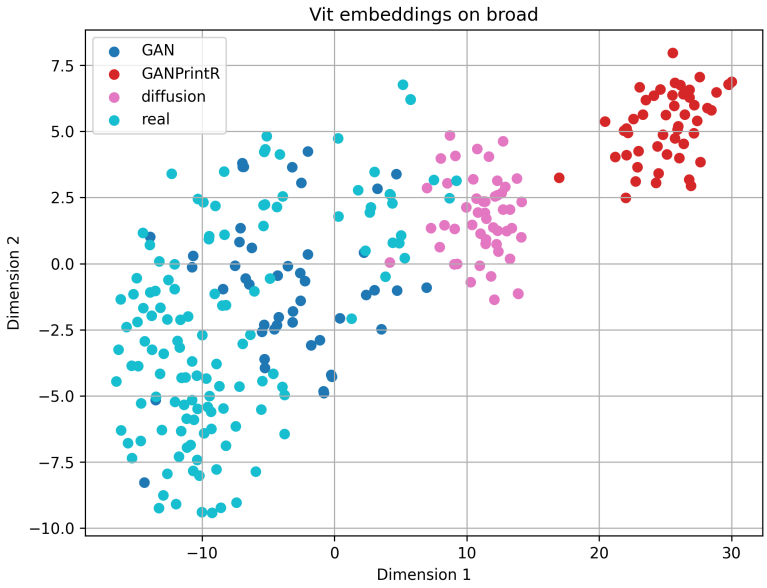

Figure 5: t-SNE plot of VIT model trained on all datasets.

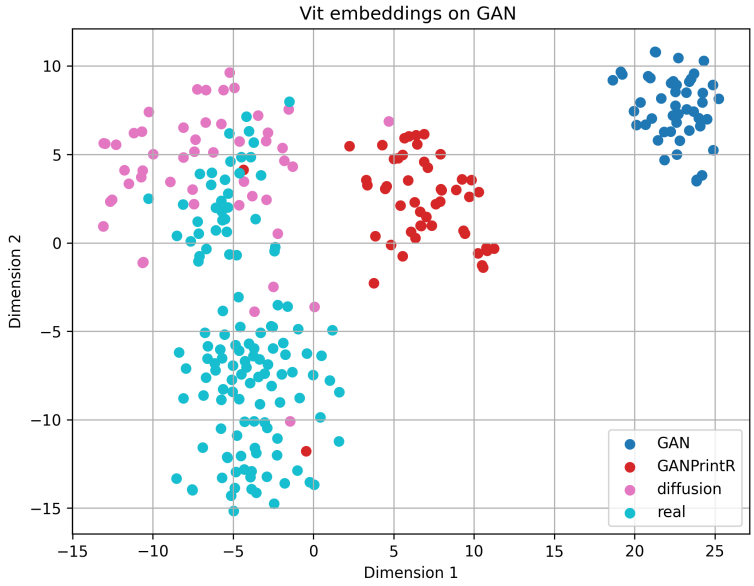

Figure 6: t-SNE plot of VIT model trained on GAN images.

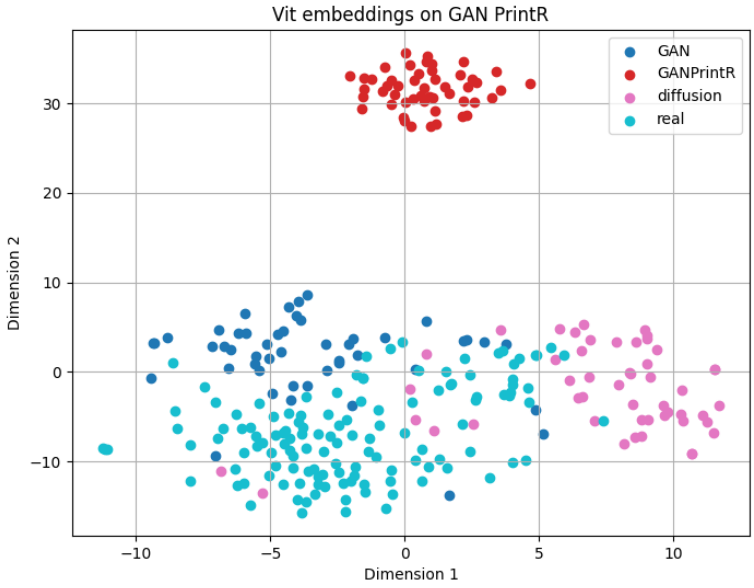

Figure 7: t-SNE plot of VIT model trained on GANprintR images.

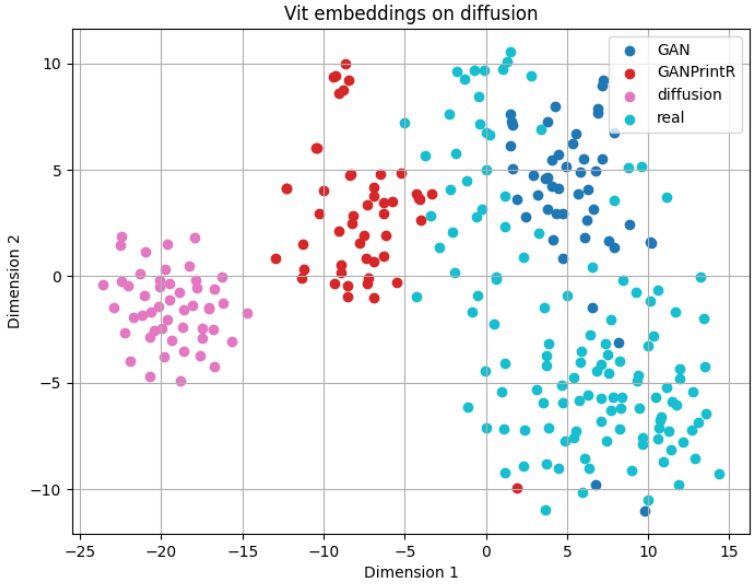

Figure 8: t-SNE plot of VIT model trained on Diffusion images.

5 App

We created an application on streamlit which classifies whether the image is AI generated or real. The user can choose from multiple models to try for the classification. The user just needs to upload an image of any size. Preprocessing of the image takes place internally and the classifier gives the output with the pie chart showing the probability percentage of the image being real or AI generated. Furthermore, the app also indicates how likely it is that the image comes from a specific model used to create fake images.

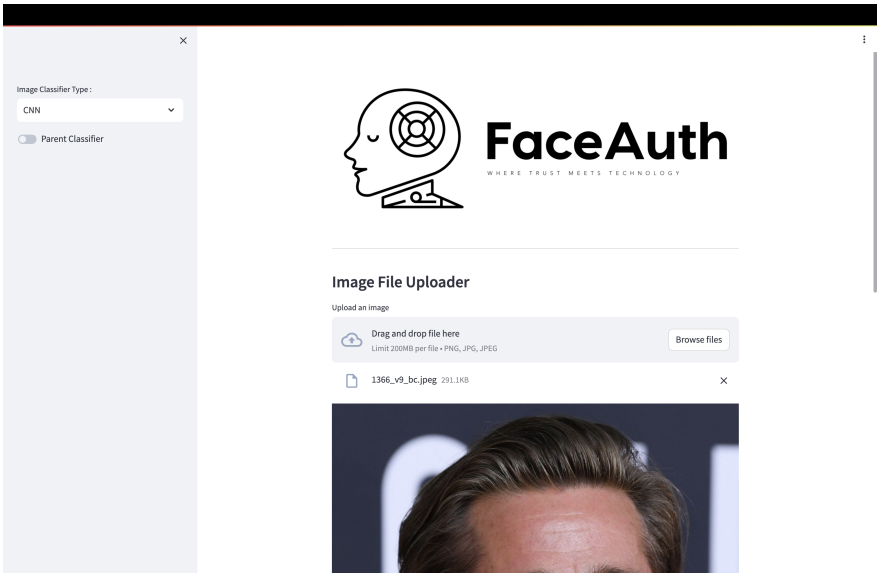

Figure 9: App Screenshot 01

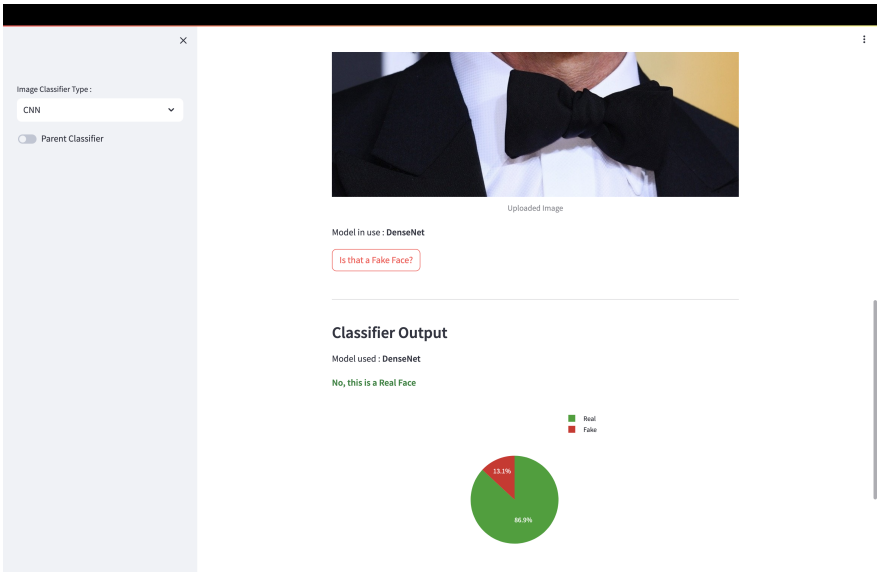

Figure 10: App Screenshot 02

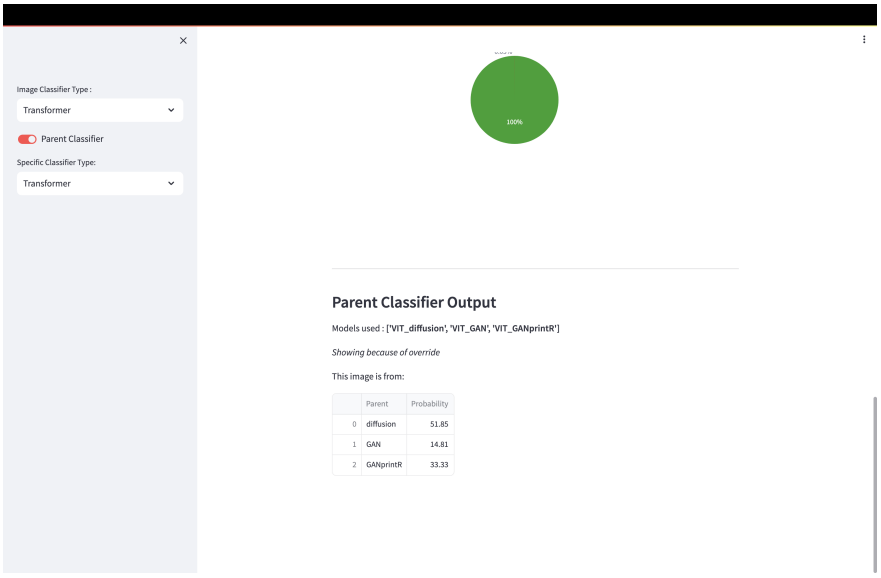

Figure 11: App Screenshot 03

6 Conclusion

In conclusion, the project successfully addresses the challenge of fake face detection by employing a combination of CNN and transformer-based models. The ViT model emerges as the most effective in terms of accuracy and generalization, outperforming other models. The t-SNE plots offer insights into the latent representations of images, and the Streamlit app provides a user-friendly interface for real-time classification. Possible improvements, such as implementing pretrained GAN discriminators and incorporating face cropping in the app, are suggested for future enhancements. Overall, the project contributes to the field of deep learning for face authentication and detection of AI-generated faces.

7 Possible Improvements

- We wanted to classify images using a pretrained GAN discriminator. We could not make it work for this project but we would love to try and implement and see its results.
- We were not able to implement face cropping in the app image preprocessor, a feature that automatically crops the uploaded image to isolate the person's face before classification. This can give better results to the user.
- We could try ensembles of models for improving the classification results.
- The app refreshes every time when a user changes some input parameter, this is very inefficient as the model gets reloaded.

References

[1] Google/vit-base-patch16-224. Hugging Face. n.d.

[2] Openrl/deepfakeface. Hugging Face. n.d.

[3] Papers with code - ifakefacedb dataset. Papers With Code. n.d.

[4] J. C. Neves, R. Tolosana, R. Vera-Rodriguez, V. Lopes, H. Proença, and J. Fierrez. Ganprintr: Improved fakes and evaluation of the state of the art in face manipulation detection. July 1 2020.

[5] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. January 15 2021.

[6] S.-H. Tsang. Review-deit: Data efficient image transformer, August 14 2022.

[7] B. Zhang, S. Gu, B. Zhang, J. Bao, D. Chen, F. Wen, Y. Wang, and B. Guo. Styleswin: Transformer-based gan for high-resolution image generation. July 21 2022.