Evaluations of Deepfake Detection Tools

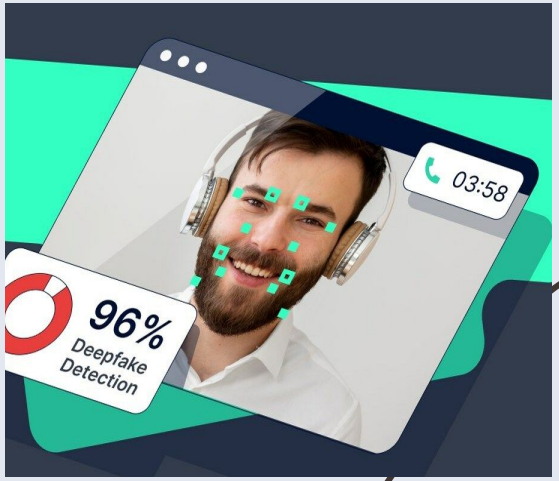
Aden Harris, Sophia Liang, Connie Chiang, Justin Chen

Introduction and Motivation

Goal of the project: test different detection tools and evaluate their accuracy

Importance:

1. Reliable detection tools help identify manipulated content, ensuring the integrity of info
2. Robust detection tools can safeguard personal privacy and security
3. Evaluating deepfake detection tools can identify the most accurate, scalable, and efficient approaches for real-world applications



Related Work

Bishop's 12.1 Principle Component Analysis (PCA)

- PCA extracts and reduces features from image frames, which are the training and testing datasets for deepfake detection tools
- PCA can also highlight irregularities in facial expression

Deepfake detection using deep learning methods: A systematic and comprehensive review (Heidari et al., 2023)

- Paper investigates deepfake detections using deep learning-based algorithms to understand its real-world applications

Deepfake Image Detection using Vision Transformer Models (Ghita et al., 2024)

- Paper provides an overview of popular deepfake detection tools used and introduces Vision Transformer model-based deepfake detection technique

Leveraging Frequency Analysis for Deep Fake Image Recognition (Frank et al., 2020)

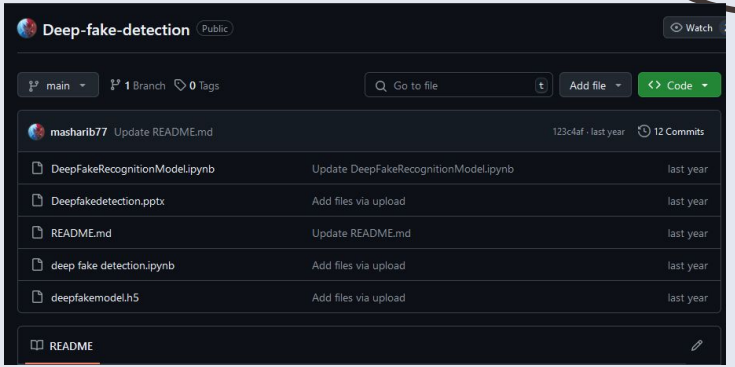
- Paper analyzes the use of frequency representation in identifying Generative Adversarial Networks (GANs)-generated images
- Therefore, deep fake images can be detected in an automatic way

Method

- Attempted to find and test diffusion, transformer, and GAN(CNN) open source deepfake detection models on the task-test2 dataset of real and deepfaked images sourced from - https://iplab.dmi.unict.it/deepfakechallenge/#importantdates
- Attempted to find and use models that were backed and referenced by scientific papers
- Ran and tested each model locally
- Tried to find and use pretrained models if possible
- Applied a classification report at the end of successful testing to show accuracy of models

Deep-Fake-Detection by masharib77

-Used a combination of a Vision transformer and Neural network
-was pretrained by https://www.kaggle.com/datasets/ciplab/real-and-fake-face-detection
-when ran on a subset of our test dataset only achieved a accuracy of .48 and a weighted average of 0.62




```
Overall Metrics:
Accuracy: 58.00%

Detailed Classification Report:
              precision    recall  f1-score   support

        Fake       0.22      0.29      0.25        24
        Real       0.75      0.67      0.71        76

    accuracy                           0.58       100
   macro avg       0.48      0.48      0.48       100
weighted avg       0.62      0.58      0.60       100
```

image.img_to_array:

- Converts the image into a NumPy array.

np.expand_dims(img_array, axis=0):

- Adds a new dimension to represent a batch (since models expect batches of data for prediction).

model.predict(img_array):

a. Feeds the preprocessed image to the model for prediction.
b. Returns an array of prediction probabilities.

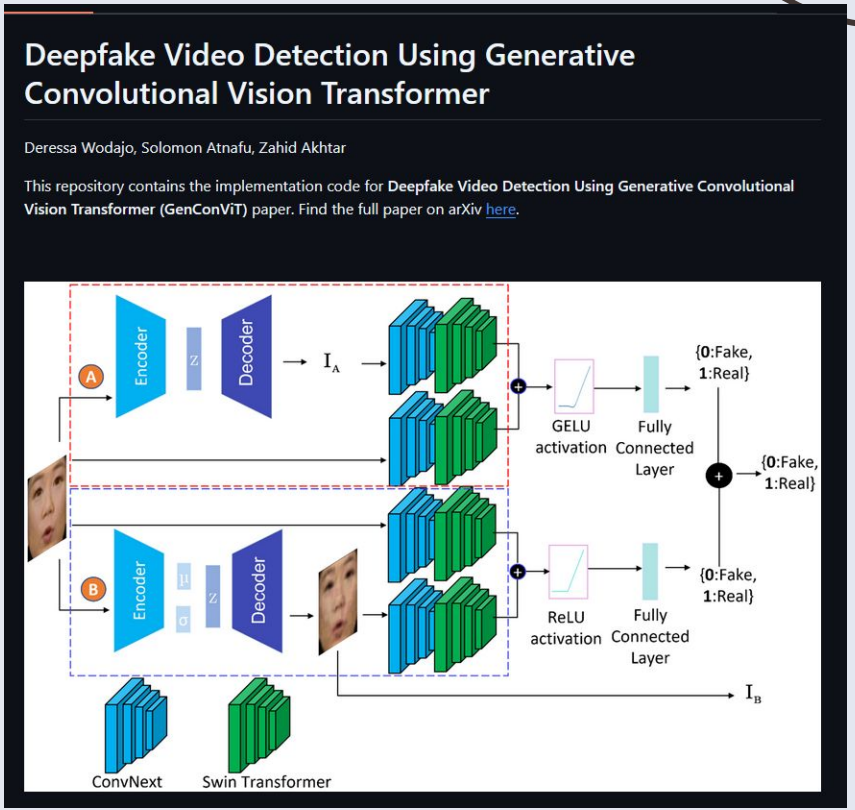
confidence = prediction[0][0]:

c. Extracts the confidence score for the prediction (assuming binary classification).

Also set up if If the confidence score is greater than 0.5, the image is classified as **FAKE**. Otherwise, it is classified as **REAL**. The confidence score is formatted to two decimal places for readability.

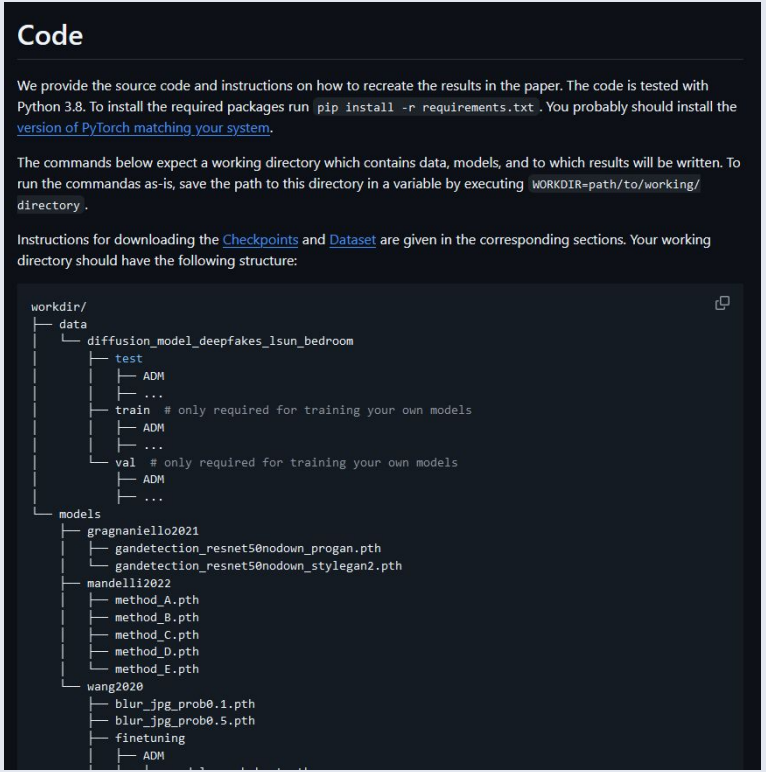
GenConViT Deepfake Detection by erprogs

-uses a generative convolutional vision transformer to detect deep fake frames in videos
-came pre trained as well (from huggingface)
-attempted to modify the config files and the main() functions to allow it to be ran on a file of images instead of taking frames from videos
-ran into multiple problems including environment compatibility issues, including (but definitely not limited to) with tensorflow and the wrong combination of python 3.6 and the pretrained weights
-also used a GPU by default instead of a CPU
-ended up not being able to run it on the test dataset



Diffusion-model-deepfake-detection by jonasricker

-used a very specific set up of the working directory and test data
-used multiple models to output a confidence on how likely a picture was a deepfake
-when trying to get to run on the test data set continuously ran into issues with the working directory set up
-wouldn't intake differently sized and formatted images than used in the originally trained training and test data



Code

We provide the source code and instructions on how to recreate the results in the paper. The code is tested with Python 3.8. To install the required packages run `pip install -r requirements.txt`. You probably should install the version of PyTorch matching your system.

The commands below expect a working directory which contains data, models, and to which results will be written. To run the commandas as-is, save the path to this directory in a variable by executing `WORKDIR=path/to/working/directory`.

Instructions for downloading the Checkpoints and Dataset are given in the corresponding sections. Your working directory should have the following structure:

```
workdir/
├── data
│   └── diffusion_model_deepfakes_lsun_bedroom
│       ├── test
│       │   ├── ADM
│       │   ├── ...
│       ├── train  # only required for training your own models
│       │   ├── ADM
│       │   ├── ...
│       └── val  # only required for training your own models
│           ├── ADM
│           ├── ...
└── models
    ├── gragnaniello2021
    │   ├── gandetection_resnet50nodown_progan.pth
    │   └── gandetection_resnet50nodown_stylegan2.pth
    ├── mandelli2022
    │   ├── method_A.pth
    │   ├── method_B.pth
    │   ├── method_C.pth
    │   ├── method_D.pth
    │   └── method_E.pth
    └── wang2020
        ├── blur_jpg_prob0.1.pth
        ├── blur_jpg_prob0.5.pth
        ├── finetuning
        │   ├── ADM
```

NPR-deepfakeDetection by chuangchuangtan

- focused on detecting CNNs
-attempted to get to run on the test dataset
-requires GPU to be able to run :/
-tried to modify so that it would use a CPU instead but the modifications just wouldn’t work
-also ran into a bunch of trouble with package compatibility. No matter what version of Torch and TorchVision I tried it just wouldn't work

Rethinking the Up-Sampling Operations in CNN-based Generative Network for Generalizable Deepfake Detection

Beijing Jiaotong University, YanShan University, A*Star

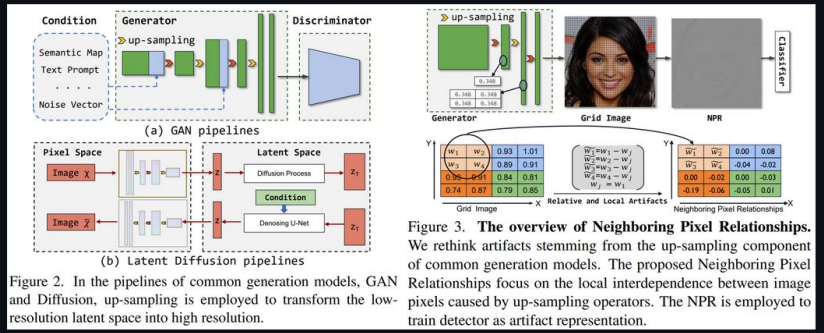


Figure 2. In the pipelines of common generation models, GAN and Diffusion, up-sampling is employed to transform the low-resolution latent space into high resolution.

Figure 3. **The overview of Neighboring Pixel Relationships.** We rethink artifacts stemming from the up-sampling component of common generation models. The proposed Neighboring Pixel Relationships focus on the local interdependence between image pixels caused by up-sampling operators. The NPR is employed to train detector as artifact representation.

Discussions

Lessons learned

- The importance of aligning model architecture, such as input image size, with dataset requirements
- Having enough computing power to run these detecting models

What could have done better

- Allocate more time for data preprocessing
- Test smaller data before full-scale deployment

What didn’t work

- Models failed to run because many pre-trained models were designed for specific datasets
- Possible version mismatch between libraries (TensorFlow and PyTorch)

Improvements

1. Data preprocessing
    a. Standardize the input images with consistent resizing for better model generalization
2. Model training strategy
    a. Freeze lower layers and fine-tune upper layers with a custom dataset in order to retain general feature extraction from pre-trained models while adapting the model to specific dataset
    b. Be able to have self-attention modules to focus on discriminative regions in images