A CNN Based Approach to Detect Deepfake

Pritam Mondal and Arya Panja
Prof. Nivedhitha M
School of Information Technology & Engineering
Department of Computer Application
Vellore Institute of Technology, Vellore – 632014, Tamil Nadu, India.
E-mail: nivedhitha.m@vit.ac.in

Abstract

Deepfakes are an emerging threat to our society as they can be used to manipulate and deceive people with false information. Deepfake detection is a challenging task, and traditional methods are limited in their effectiveness. Therefore, in this study, we propose a deep learning-based approach for Deepfake prediction using convolutional neural networks (CNN). Our proposed method involves training a CNN model on a dataset of real and fake images obtained in Kaggle. Prior to it, we performed transfer learning using EfficientNetB0 which has already been trained on the ImageNet dataset. The model learns to distinguish between real and fake images by identifying patterns and features that are unique to each class. The results show that our proposed CNN-based approach performs decently in predicting fake images. We are aiming forward in achieving better results.

Key Words

Deepfake; Image Detection; Convolutional neural networks; CNN; Deep learning.

Introduction

Deepfake technology's rapid growth has sparked worries about the possible abuse of modified media, especially photos and videos. Deep learning algorithms are used to generate or modify photographs to produce Deepfake images, which may be hard to tell apart from real images. This has important ramifications for a variety of industries, including entertainment, politics, and criminal investigations.

Public personalities, such as celebrities, sports, and politicians, are the most severely affected by Deepfakes because of the abundance of videos and images that are readily available online. Deep fake technologies are mostly used to produce adultery material, despite the fact that they are occasionally used to make fun of other people. These photos, which are freely available on the Internet, include pornographic models with the faces of several celebrities and other well-known people grafted onto their bodies. Deepfake technology may use people's voices and images to produce political, pornographic, or humorous content about well-known individuals without getting their permission. Anyone may create false content that is undetectable to the real content because of how simple various programs are to use. Cyberbullying is affecting a lot of young people.

Our major goal is to correctly distinguish between deepfake and real photos. Researchers have created a variety of methods for spotting deepfake photos to solve this problem. Convolutional neural networks (CNNs), a subset of deep learning algorithms that are particularly effective for image classification tasks, are one such approach. Researchers have shown encouraging results in recognizing deepfake photos with a high degree of accuracy by training CNNs on vast datasets of both real and fake images. In this study, we investigate the use of CNNs for deepfake image recognition, evaluating previous studies and approaches, identifying significant difficulties, and outlining potential future paths.

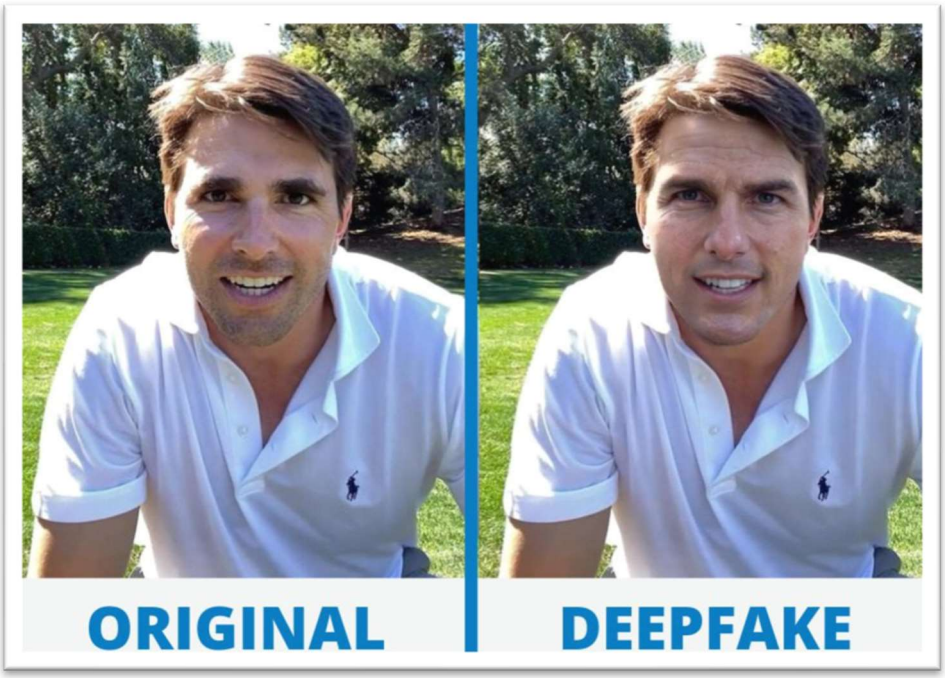


Fig 1. Sample Image

Related Work

1. **D. Güera and E. J. Delp, "Deepfake Video Detection Using Recurrent Neural Networks," 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Auckland, New Zealand, 2018, pp. 1-6, doi: 10.1109/AVSS.2018.8639163.**

   This paper uses two essential components: CNN is used for frame feature extraction and LSTM is used for temporal sequence analysis. To provide a temporal sequence descriptor for the shot frame's picture processing, a convolutional LSTM is used. The high-dimensional LSTM descriptor is mapped to a final detection probability via the integration of fully-connected layers, with the goal of achieving end-to-end learning. To reduce training over-fitting, our shallow network specifically comprises two fully-connected layers and one dropout layer. The dataset used contains 300 deepfake videos from multiple video-hosting websites are obtained. We further incorporate 300 more

videos randomly selected from the HOHA dataset, which leads to a final dataset with 600 videos. The model achieved an accuracy of 97.1% with 80 frames.

2. **X. Chang, J. Wu, T. Yang and G. Feng, "DeepFake Face Image Detection based on Improved VGG Convolutional Neural Network," 2020 39th Chinese Control Conference (CCC), Shenyang, China, 2020, pp. 7252-7256, doi: 10.23919/CCC50068.2020.9189596.**

   The chosen main architecture is VGG16. It is possible to separate the convolution layer and pooling layer of VGG16 into several blocks. 13 volume layers, 3 completely linked layers, and 5 pool layers make up each block, which also has many volume layers and a pool layer. This paper proposes a type of VGG network based on noise and image augmentation (NA-VGG) by adding an SRM filter layer and an image augmentation layer in front of the VGG16 network.
   Celeb-DF dataset is used for training evaluation. The image is extracted from the DeepFake video. 12416 training set images (including 5334 original images and 7082 DeepFake images) are extracted, 1376 verification set images (including 573 original images and 803 DeepFake images) and 1376 test set images (including 552 original images and 552 DeepFake images) are extracted 824 DeepFake images. The model achieved an accuracy of 85.7 %.

3. **Huaxiao Mo, Bolin Chen, and Weiqi Luo. 2018. Fake Faces Identification via Convolutional Neural Network. In Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec '18). Association for Computing Machinery, New York, NY, USA, 43–47. https://doi.org/10.1145/3206004.3206009.**

   The proposed architecture inputs an RGB picture which is transformed into residuals using high-pass filter and fed into three-layer groups containing a convolutional layer fitted with LReLu and a max pooling layer. The output is then fed into two fully-connected layers. SoftMax layer is finally used to produce the output in the end. Dataset is prepared from CELEBAHQ dataset by taking 30000 true face images and furthermore, 30000 fake images are selected from fake face image database.

4. **Deepfake Video Detection through Optical Flow Based CNN**
   Irene Amerini, Leonardo Galteri, Roberto Caldelli, Alberto Del Bimbo; Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 0-0.

   The suggested approach in this study uses optical flow to differentiate between a deepfake and the genuine picture. It is said that optical flow takes use of differences in motion between frames that are artificially made in comparison to those that are naturally generated by a video camera. It is computed on two successive frames. The CNN (pre-trained with VGG-16/ResNet50) is fed optical flows. After that, a final completely linked layer with one output unit is added to the network, and it is followed by a sigmoid activation to determine if a frame is false or real. The dataset utilised is

the FaceForensics++ dataset, which is made up of 1000 original video clips that have undergone three automated face alteration techniques. Training set consists of 720 videos while validation and test set contain 120 videos each. The model gives an accuracy of 81.61% with VGG16 and 75.46% with ResNet50.

5. **Hsu, Chih-Chung, Yi-Xiu Zhuang, and Chia-Yen Lee. 2020. "Deep Fake Image Detection Based on Pairwise Learning" Applied Sciences 10, no. 1: 370. https://doi.org/10.3390/app10010370.**

   The proposed CFFN consists of three dense units that include two, four, and three dense blocks. The parameter θ in the transition layer is 0.5 and the growth rate is 24. Then, a convolution layer with 128 channels and 3 × 3 kernel size is concatenated to the output layer of the last dense unit. Finally, the fully connected layer is added to obtain the discriminative feature representation. To obtain the cross-layer feature representation, they also reshape the last layers of the first and second dense units to aggregate the cross-layer features into the fully connected layer. Therefore, in the final feature representation, there are 128×3=384 neurons.
   The dataset used in the experiments was extracted from CelebA. The images from the CelebA covered large pose variations and background clutter, including 10,177 of identities and 202,599 aligned face images. This method gives a recall value of 0.900.

6. **Hasin Shahed Shad, Md. Mashfiq Rizvee, Nishat Tasnim Roza, S. M. Ahsanul Hoq, Mohammad Monirujjaman Khan, Arjun Singh, Atef Zaguia, Sami Bourouis, "Comparative Analysis of Deepfake Image Detection Method Using Convolutional Neural Network", Computational Intelligence and Neuroscience, vol. 2021, Article ID 3111676, 18 pages, 2021. https://doi.org/10.1155/2021/3111676.**

   In order to improve the findings achieved, the authors of this work employed the CNN basic architecture in conjunction with pre-training the model using several DenseNet and ResNet iterations. Following that, there are layers for pooling and layers for fully connected. Data are supplied into the model, and the matching output is produced. The Flickr dataset, which was compiled by Nvidia Corporation, had 70,000 genuine faces in the dataset that was purchased from Kaggle. In addition, out of the one million phoney faces created by styleGAN, 70,000 were real. Later, the images were downsized to 256 pixels and combined from the two datasets. Lastly, the dataset was divided into three parts, including the train, validation, and test set the architecture achieves an accuracy of 81.6% with ResNet50 which is the highest.

Materials and Methods

Dataset

Our goal is to make our model dataset independent. Hence at the moment we have downloaded two datasets from https://www.kaggle.com/ . The first one contains 70,000 real image and

70,000 fake images. The second one has 1000 real images and 1000 fake images. Every data from both datasets is labeled data.

Data Preprocessing

Data cleaning

The dataset we obtained consisted of a large number of chest X-Ray pictures of various angles. In addition to the PA (posteroanterior) view, there were many side-view images as well, hence we clean our dataset by erasing side-view images and those images that were a bit blurry and low in quality. This prevented our model from getting biased.

Resizing data

Resizing data is a primary procedure in data pre-processing because smaller images are trained more quickly by deep learning models. Every image will use the same amount of processing RAM if they are of the same size. Memory must be restructured if its size varies, which will take time. Insufficient memory may also result in the process failing. Hence each image was resized to the standard 224*224 size.

Data augmentation

Data augmentation is a technique used to artificially increase the size of the training dataset by applying various transformations to the original images. By doing so, it helps to prevent overfitting and improve the model's generalization ability. Here we used some data augmentation technique such as

1. rescale: This rescales the pixel values in the input images to be in the range [0,1] by dividing each pixel value by 255.
2. rotation_range: This randomly rotates the image by a degree between 0 and 10.
3. width_shift_range: This randomly shifts the image horizontally by a fraction of its width between 0 and 0.1.
4. height_shift_range: This randomly shifts the image vertically by a fraction of its height between 0 and 0.1.
5. shear_range: This randomly applies a shear transformation to the image with a shear intensity between 0 and 0.2.
6. zoom_range: This randomly zooms into the image by a factor between 1 and 1.1.
7. horizontal_flip: This randomly flips the image horizontally.
8. fill_mode: This specifies the strategy to use for filling in newly created pixels during the transformation process. In this case, the "nearest" strategy is used, which fills in new pixels with the value of the nearest pixel in the original image.

Proposed Model

In our proposed model, we have used the CNN classifier using the EfficientNetB0 model as a base. We have initialized an instance of the EfficientNetB0 model with pre-trained weights from the ImageNet dataset, excluded the top layer of the model, and used max pooling to reduce the output of the convolutional layers to a single vector for each image. Then, we created a sequential model, and added the EfficientNetB0 model as the first layer of the sequential model, three dense layers with ELU activation function and dropout layers to prevent overfitting. Finally, a dense layer with a sigmoid activation function is applied to output a binary classification result.

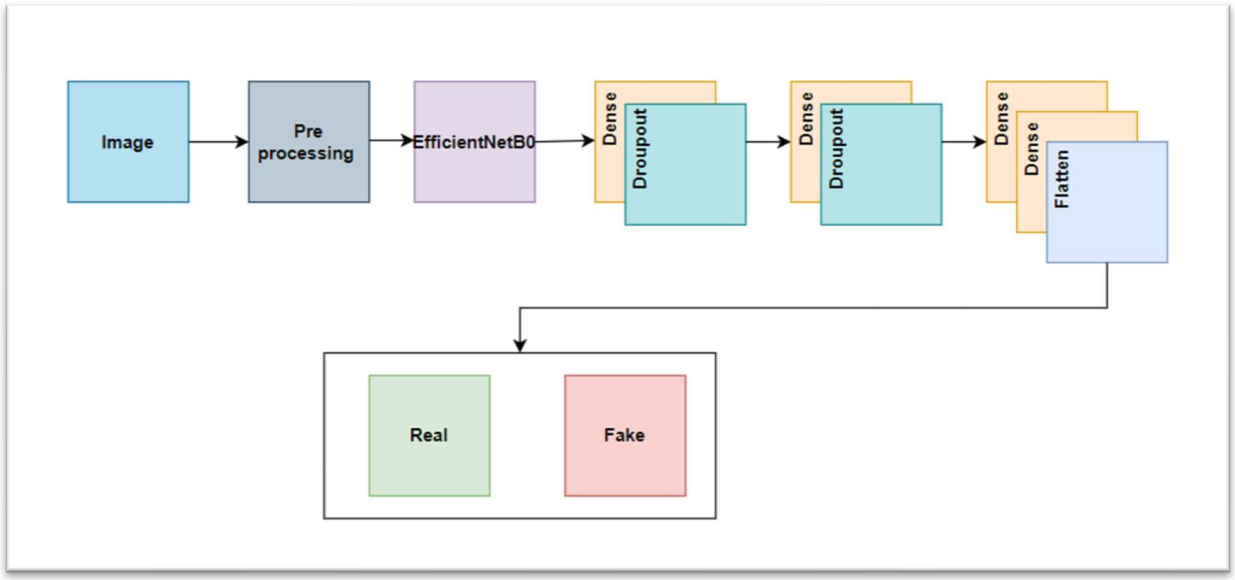


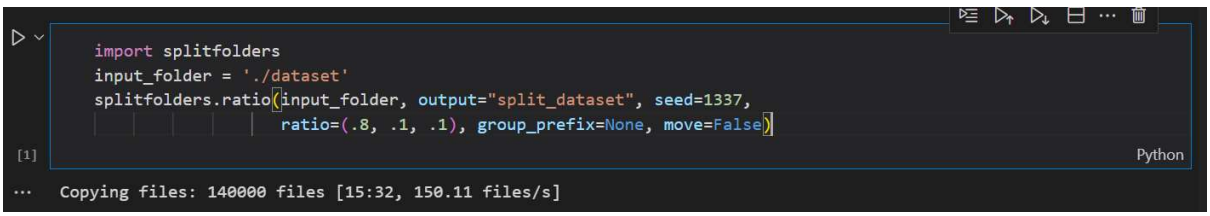
Fig 2. Proposed Architecture of our model

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnet-b0 (Functional  (None, 1280)             4049564
 )

 dense (Dense)               (None, 512)               655872

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 128)               65664

 dropout_1 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 64)                8256

 dense_3 (Dense)             (None, 1)                 65

 flatten (Flatten)           (None, 1)                 0

=================================================================
Total params: 4,779,421
Trainable params: 4,737,405
Non-trainable params: 42,016
_________________________________________________________________
```

Fig 3. Model Summary

Implementation

1. prepare_dataset.ipynb



2. train_CNN.ipynb

```
# Train a CNN classifier
efficient_net = EfficientNetB0(
    weights = 'imagenet',
    input_shape = (input_size, input_size, 3),
    include_top = False,
    pooling = 'max'
)



model = Sequential()
model.add(efficient_net)
model.add(Dense(units = 512, activation = 'ELU'))
model.add(Dropout(0.5))
model.add(Dense(units = 128, activation = 'ELU'))
model.add(Dropout(0.5))
model.add(Dense(units = 64,activation='ELU'))
model.add(Dense(units = 1, activation = 'sigmoid'))
model.add(Flatten())

model.summary()
```

```
# Compile model
model.compile(optimizer = Adam(learning_rate=0.0001),
loss='binary_crossentropy', metrics=['accuracy'])

checkpoint_filepath = '.\\tmp_checkpoint'
print('Creating Directory: ' + checkpoint_filepath)
os.makedirs(checkpoint_filepath, exist_ok=True)

custom_callbacks = [
    EarlyStopping(
        monitor = 'val_loss',
        mode = 'min',
        patience = 5,
        verbose = 1
    ),
    ModelCheckpoint(
        filepath = os.path.join(checkpoint_filepath, 'best_model.h5'),
        monitor = 'val_loss',
        mode = 'min',
        verbose = 1,
        save_best_only = True
    )
]
```

Signature Student

1. Pritam Mondal [22MCA0129]

2. Arya Panja [22MCA0355]

Guide Signature

1. Nivedhitha M