Deepfake Image Detection: A VGG-16 based CNN Architecture Approach that Utilizes Data Augmentation

Tingqi (Ting) Wang

Abstract

With deepfakes becoming massively generated and widely distributed in today's society, the devastating impacts on deep-faked victims have never been larger. Further worsening this matter is the increasingly growing power of A.I. that has directly contributed to the quality of deepfakes that have made it extremely difficult to distinguish with the human eyes. In order to tackle the issue of deep-faked images, I have proposed a VGG-16 based CNN model that utilizes data augmentation, which aims to distinguish between deep-faked images to real images. After training on a 35k training set obtained from the "deepfake and real images" dataset, the VGG-16 is tested on two 5k test sets obtained from two different datasets ("deepfake and real images" and "deepfake_faces"). The model was able to achieve a 76.94% accuracy on the "deepfake and real images" test set but performed very poorly on the "deepfake_faces" test set. After analysis, the usage of two poorly chosen datasets seem to be the main culprits for the low performance of the proposed model.

1. Introduction

Deepfakes, synthetic images or videos that are manipulated or generated using A.I. to display falsified information, is an unethical practice that is inflicting high social controversy. Many individuals in the fields of politics, social media, and entertainment have fallen victim of deepfakes. Victims have suffered from appearances on pornography to performances of inappropriate speech and actions, and has resulted in detrimental harm to their public image. With the advanced capabilities of machine learning, specifically Deep Learning, deepfakes have become undetectable by the human eye. As evident in the recent $25 million scam where a multinational firm worker was tricked by scammers imposing as the company's CFO using deepfake technology (Chen & Magramo, 2024). Consequently, there has been a significant rise of social distrust in images and videos on social media and the internet.

The project attempted to address this pressing issue by providing a VGG-16 based Convolutional Neural Network (CNN) model to detect deep-faked images in the attempt to limit the consequences of deepfakes. The model is able to take a digital image input and provide an output of whether the image is deep-faked or real. After testing the model's performance on the testing dataset, the model was able to achieve a 76.94% accuracy (0.5 log loss) on the "deepfake and real images" test set but performed very poorly on the "deepfake_faces" test set with a 49.98% accuracy (2.89 log loss).

The GitHub repository for this project can be found here.

2. Related Work

2.1. Frequency Domain Analysis

Frequency domain analysis focuses on examining images to find inconsistent patterns and anomalies in the frequency domain. Through adopting Fourier transformation the image can be converted from the spatial domain to the frequency domain, then using the Support Vector Machines and Linear Kernels a model can be constructed. Agarwal et al.'s model was able to achieve 99.76% accuracy in distinguishing deepfake images, but one significant issue is that the resolution of images needed to be very high (Agarwal et al., 2021).

Comparing the proposed VGG-16 based CNN model to Agarwal et al.'s model, it can be seen that my model has significantly been outperformed in terms of accuracy. However, similar to Agarwal et al.'s model, my proposed model seem to have suffered from the same challenge of needing high resolution images, along with also needing well selected data samples.

2.2. Convolutional Neural Networks (CNNs)

Shad et al. researched into different machine learning techniques to evaluate the effectiveness on deepfake image detection and their results demonstrated that CNNs were the best fit for this application (Shad et al., 2021). In line with this, Patel et al. were able to produce a dense CNN architecture that achieved 97.2% accuracy when test on 7 different data sources (Patel et al., 2023a). However, there

are challenges regarding the trade-off between accuracy and time latency that Patel et al. faced in their project. In addition, the generalization of deepfake image detection models and the ability to maintain effectiveness following improvements in generated deepfake images are additional challenges for CNN-based deepfake detection models (Patel et al., 2023b). Nevertheless, the incorporation of data augmentation of images - such as changing the contrast, applying rotations, adding noise, and cropping images - in Jellali et al.'s CNN model enabled a close to 99% accuracy with a limited dataset and provides insights into the improved generalization data augmentation may enable (Jellali et al., 2023).

My proposed VGG-16 based CNN model is not at all competitive with Patel et al.'s and Jellali et al.'s CNN models in terms of test accuracy. But my VGG-16 model has attempted to incorporated features of both models. Nevertheless, the problem of trade-off between accuracy and latency stated by Patel et al was also evident in my project: hyperparameter tuning took over 3 hours when using 2 Nvidia A100 GPUs.

2.3. VGG-16 and CNN Hybrid Model

Raza et al. proposed a very similar model to the one I have proposed in this project, where they also utilized the pretrained VGG-16 model in creating their hybrid model. They stated that their model was able to achieve 94% accuracy and a loss of just 0.2 (Binary Cross Entropy Loss) (Raza et al., 2022).

Though adopting a very similar approach, my VGG-16 based CNN model performed significantly poorer than the hybrid CNN model. But after examining the dataset (only around 2k images, but very high quality) Raza et al. used to train their model and their model's architecture, I found that there were very minimal differences in the architecture of our models. So I believe that the drastically different datasets were the determining factor in our model's contrasting success.

3. Dataset and Evaluation

3.1. Dataset

In this project, I have decided to utilize two separate datasets. I will be using the "deepfake and real images" dataset to train and test the model, and I will use the "deepfake_faces" dataset to test the generalization of the model. Both datasets are obtained from Kaggle.

3.1.1. "DEEPFAKE AND REAL IMAGES" DATASET (TRAINING AND TESTING)

The dataset contains 256 X 256 images that is labeled as "REAL" or "FAKE" and is pre-split into train (140k - FAKE: 70.0k, REAL: 70.0k), dev (39.4k - FAKE: 19.6k, REAL: 19.8k), and test (10905 - FAKE: 5,492, REAL: 5,413). Although there is a significant amount of available images in this dataset, I decided to use a 50k-subset from the entire dataset due to the limitations in computing resources. The size of train, dev, and, test sets are 35k, 10k, 5k respectively, with a perfect 50/50 split in each set to ensure accuracy being a suitable evaluation metric.

3.1.2. "DEEPFAKE_FACES" DATASET (TESTING ONLY)

The dataset contains 95,634 (FAKE: 79,376, REAL: 16,258) unique 224 X 224 images that are labeled as "REAL" or "FAKE". I took 50k images (25k FAKE and 25k REAL) to create a test set which I used to test the generalization capabilities of the models. To allow the models to correctly classify the test images from the created test dataset, I resized the images from 224 X 224 to 256 X 256.

3.2. Evaluation

After testing on both datasets, I will use accuracy to measure the overall correctness of the model in detecting deepfake images. I will also use log loss to evaluate the model's performance, as it was the evaluation metric used by the Facebook Deepfake Detection Challenge on Kaggle.

The log loss is defined as follow:

$$LogLoss = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)] \quad (1)$$

where

- n is the number of images being predicted
- $\hat{y}_i$ is the predicted probability of the image being FAKE
- y_i is 1 if the image is FAKE, 0 if REAL
- log() is the natural (base e) logarithm

4. Methods

The project includes two different CNN models.

4.1. Baseline CNN

For the baseline of the project, I used a simple CNN model. To train this model, I used the 50k-subset from the "deepfake and real images" dataset. Then tested the model on both datasets. Using the results of this model I can evaluate the

improvement of my proposed VGG-16-based CNN model in detecting deepfake images.

4.1.1. Architecture

The following is the architecture of the CNN model and the corresponding output dimension after each layer:

- Input (3 X 256 X 256)
- Convolution1 + Relu (16 X 256 X 256)
- Max-Pooling (16 X 128 X 128)
- Convolution2 + Relu (32 X 128 X 128)
- Max-Pooling (32 X 64 X 64)
- Convolution3 + Relu (64 X 64 X 64)
- Max-Pooling (64 X 32 X 32)
- Flatten (65536)
- Dropout
- Linear1 (256)
- Dropout
- Linear2 (1)

4.1.2. Hyperparameters

The hyperparameters that are being used for the baseline model utilizes the PyTorch defaults except the following which followed Yamashita et al.'s CNN model (Yamashita et al., 2018):

- Convolution layers:

  kernel size = 3, stride = 1, padding = 1

- Max-Pooling layers:

  kernel size = 2, stride = 2, padding = 0

- Hidden layer:

  size = 256

- Dropout layer:

  probability = 0.5

4.2. VGG-16 based CNN

The VGG-16 based CNN architecture that adopts transfer learning is my proposed model for the project. As the VGG-16 is a pre-trained model for image classification, the amount of data needed to train the model can be reasonably reduced and will attempt to improve the accuracy and time consumption trade-off. In addition, VGG-16 has already developed strong feature extraction which can be helpful towards detecting deep-faked faces. I tuned and tweaked the model using the "deepfake and real images" dataset so that the VGG-16 model becomes adapted to detecting deepfake images. I have also incorporated data augmentation when tuning the model to increase the generalization ability of the model and potentially provide more robustness towards newly generated deepfake images.

4.2.1. Architecture

The following is the architecture of the CNN model and the corresponding output dimension after each layer:

- Input (3 X 256 X 256)
- VGG-16
- Linear1 + Relu (512)
- Dropout
- Linear2 + Relu (128)
- Dropout
- Linear3 (1)

4.2.2. Hyperparameters

The hyperparameters that are being used for the VGG-16 based model follows the PyTorch VGG-16 model defaults except the following which followed Yamashita et al.'s CNN model (Yamashita et al., 2018):

- Hidden layers:

  Layer 1: size = 512
  Layer 2: size = 128

- Dropout layer:

  probability = 0.5

I have currently froze all the layers of the VGG-16 model (from the $torchvision$ library) except the last 10 layers so that these layers can be fine-tuned and allow for better specialization in the deepfake detection task. As a result, when training, I am using a smaller learning rate (0.1 times smaller) for the 10 unfrozen pre-trained layers to ensure only small tweaks are being applied.

4.3. Training

To prepare the data for training, I used $DataLoader$ that took a 50/50 split of REAL and FAKE images from the 'deepfake and real images' dataset (35k, 10k, 5k). Using a dataloader for each set, I trained and tested the models. But it should be noted that the train dataloader shuffles the samples whereas the validation and test ones does not. In addition, the dataloaders also incorporates image normalization using the standard weights used for training on ImageNet: $[0.485, 0.456, 0.406]$, $[0.229, 0.224, 0.225]$. The train dataloader has also facilitated the data augmentation techniques: it performs random horizontal and vertical flips, as well as random rotations of the images.

For training both models, I have used the same loss function: $nn.BCEWithLogitsLoss()$ and utilizes early stopping. The hyperparameters used for training was selected via hyperparameter tuning on the VGG-16 based model (except a few that are common practice).

Common practice (Yamashita et al., 2018):

- momentum = 0.9
- num epochs = 10
- patience = 3 (for early stopping)
- lr (for baseline model) = 0.0001
- weight decay (for baseline model) = 0.001
- optimizer (for baseline model) = SGD
- scheduler (for baseline model) = None

Chosen through hyperparameter tuning:

- weight decay = 0.001 (For L2 regularization)
- lr = 0.0001 (for added layers of the VGG-16 based model)
- lr = 0.00001 (for the last 10 pre-trained layers of the VGG-16 model)
- batch size = 64
- optimizer = Adam
- scheduler = $ExponentialLR$ (for rate scheduling in VGG-16 based model)

To accommodate the large number of training samples, I have altered the training function to utilize GPU and CUDA. Using the GPU resources on USC CARC, the program runtime has fallen dramatically, even for the 35k training dataset.

4.4. Hyperparameter tuning

I used the simple practice of grid searching to fine the best hyperparameters values that yielded the highest validation accuracy for the VGG-16 model.

The following is the dictionary of hyperparameters and their potential values:

```
learning_rates: [1e-3, 1e-4],
batch_sizes: [32, 64],
weight_decays: [1e-4, 1e-3],
optimizers: ['SGD','Adam'],
schedulers: ['StepLR','ExponentialLR']
```

4.5. Improvements

The VGG-16 model has been trained using a training dataset that is significantly larger (35k training images) compared to the 70 images used for the midterm report. With more training samples, the model can learn from more data samples and improve its generalization abilities. In addition, the model is also trained with data augmentations. Having exposed the model to images that have been flipped and rotated allows the model to learn about identifying multiple faces at different location of the image. Reflecting on the potential issues of images normalization proposed in the midterm report, the effects of normalization does not cause the image to no long resemble human faces (it is simply the quality and nature of the image). Another feature that was included in the new model is rate scheduling, which can move the model closer to the global minima and hence improve the accuracy.

4.6. Testing

To prepare the data for testing, I used $DataLoader$ that took a 50/50 split of REAL and FAKE images from the 'deepfake and real images' dataset (5k) and the "deepfake_faces" (5k). The dataloader only includes images normalization and no data augmentations.

5. Experiments

5.1. Baseline CNN

The train and validation loss and accuracy can be seen in figure 1. The testing result on the each test set is displayed below:

"deepfake and real images" test set:

```
Test Loss: 0.6104820647239685
Accuracy:  65.9%
```

"deepfake faces" test set:

```
Test Loss: 0.7370538729667664
```

```
Accuracy: 49.5%
```

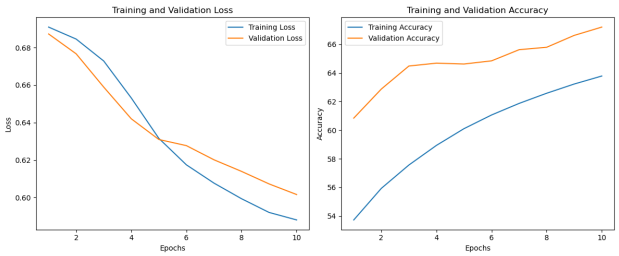

Figure 1. Plot of Training and Validation loss and Accuracy for the Baseline CNN model

5.2. VGG-16 based CNN

The train and validation loss and accuracy can be seen in figure 2. The testing result on the each test set is displayed below:

"deepfake and real images" test set:

```
Test Loss: 0.5009441809654236
Accuracy: 76.94%
```

"deepfake faces" test set:

```
Test Loss: 2.8924816952466963
Accuracy: 49.98%
```

Comparing these results with the baseline CNN model, it is evident that the VGG-16 based CNN model is performing better overall in terms of smaller losses and higher accuracy on the testset obtained from the same dataset as the train and validation sets. The reason being the VGG-16 based model being more complex and better at feature identification compared to the baseline model.

However, the two models have very close test accuracy for the test set obtained from a different dataset ("deepfake faces"). Surprisingly, the loss of the baseline model actually beats the VGG-16 based model in terms of the loss on the new test set by more than 4 times. Although it may appear at first that both models are models that lack generalization abilities on the unseen data, the VGG-16 based model being especially poor. After observing some of the misclassified images in both tests (in discussion section) as well as the "deepfake faces" in general, it can be seen that the new test set contains many bad images- pictures that are mostly black or faces are barely visible- and hence the models may perform very poorly. In addition, the reason for a lower loss in the baseline model could be its higher bias and lower variance. The simpler baseline model means that it fits less closely to the training data and they are more consistent on poor quality, unseen data.

It is also important to note that the VGG-16 based model takes slightly longer to train and test compared to the baseline model (225.60s vs. 380.19s on A100 GPUs) due to the higher complexity and number of layers.

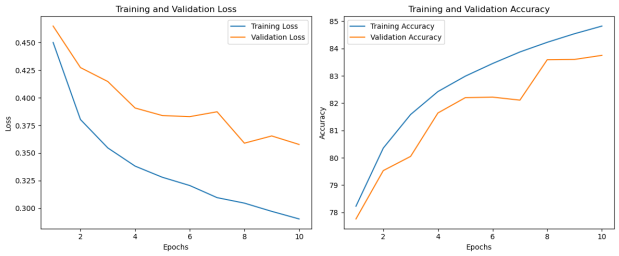

Figure 2. Plot of Training and Validation loss and Accuracy for the VGG-16-based CNN model

5.3. Hyperparameter Tuning

The results of hyperparameter tuning are displayed in a table(figure 3). Examining the table, I found that the best hyperparamters are as follow:

```
learning_rate: 0.0001
batch_size: 64
weight_decay: 0.0001
optimizer: Adam
scheduler: ExponentialLR

Validation Accuracy = 83.84
```

However, it should be noted that there are more hyperparameters (including hidden layer size and number of epochs) that can be tuned to potentially achieve even higher validation accuracy. But given the timing constraint, I have only tuned the above parameters (which took more than 4 hrs using 2 Nvidia A100 GPUs).

6. Discussion

Taking 5 random misclassified test samples of the VGG-16 based CNN model from both test sets (displayed in figure 4 and 5) I will analyse why the model misclassified and see determine further steps to improve the model.

The image samples are of poor quality and is faulty. A large proportion of these images simply do not demonstrate anything (just almost entirely black images center and right-most images from figure 5) or they are far from resembling human faces (center and right-most images from figure 4). Hence given these images, it is very hard for the model to identify any features that could be used to distinguish between real or fake human faces. And the model would likely be just guessing.

learning_rate	batch_size	weight_decay	optimizer	scheduler	val_accuracy
0.001	32	0.0001	SGD	StepLR	81.77
0.001	32	0.0001	SGD	ExponentialLR	83.18
0.001	32	0.0001	Adam	StepLR	82.27
0.001	32	0.0001	Adam	ExponentialLR	82.07
0.001	32	0.001	SGD	StepLR	82.41
0.001	32	0.001	SGD	ExponentialLR	83.06
0.001	32	0.001	Adam	StepLR	77.22
0.001	32	0.001	Adam	ExponentialLR	78.52
0.001	64	0.0001	SGD	StepLR	81.12
0.001	64	0.0001	SGD	ExponentialLR	80.64
0.001	64	0.0001	Adam	StepLR	82.5
0.001	64	0.0001	Adam	ExponentialLR	82.45
0.001	64	0.001	SGD	StepLR	81.14
0.001	64	0.001	SGD	ExponentialLR	82.57
0.001	64	0.001	Adam	StepLR	81.45
0.001	64	0.001	Adam	ExponentialLR	78.82
0.0001	32	0.0001	SGD	StepLR	77.27
0.0001	32	0.0001	SGD	ExponentialLR	78.86
0.0001	32	0.0001	Adam	StepLR	83.29
0.0001	32	0.0001	Adam	ExponentialLR	83.66
0.0001	32	0.001	SGD	StepLR	77.51
0.0001	32	0.001	SGD	ExponentialLR	79.02
0.0001	32	0.001	Adam	StepLR	82.75
0.0001	32	0.001	Adam	ExponentialLR	83.75
0.0001	64	0.0001	SGD	StepLR	74.54
0.0001	64	0.0001	SGD	ExponentialLR	76.02
0.0001	64	0.0001	Adam	StepLR	83.57
0.0001	64	0.0001	Adam	ExponentialLR	83.84
0.0001	64	0.001	SGD	StepLR	74.32
0.0001	64	0.001	SGD	ExponentialLR	76.27
0.0001	64	0.001	Adam	StepLR	83.12
0.0001	64	0.001	Adam	ExponentialLR	83.13

Figure 3. Table of hyperparameter-tuning validation set accuracys of the VGG-16-based CNN model

The noises that are present in the left-second image in figure 2, may also be challenging for the model. Despite there being quite a clear human face, the images has alterations that make it fake (label 0) and the model was unable to pick up those alterations.

The large areas of shading have resulted in many real images to be classified as fake (false negative). And on the contrary, the dis-coloring of fake images have resulted in them to be classified as real (false positives). The likely reason is that despite the discoloring, the model is still able to identify characteristics of faces, whereas when there are large ares of dark shading, the model can no longer identify features of faces, hence predicting that it is fake.

It should be noted that after looking through both datasets in depth, it was evident that the two datasets contains many poor image samples that barely resembles, or even not resembling, any features humans faces (sometimes even of nothing like demonstrated in figure 5). With this observation in mind, I feel that the model is performing as expected given the quality of the datasets. But I did expect the model to be able to better classify fake images (especially ones of poor quality) as these samples have no relations to real pictures of humans.

Figure 4. 5 examples of VGG-16 based CNN model misclassifications in the ”deepfake and real images” test set

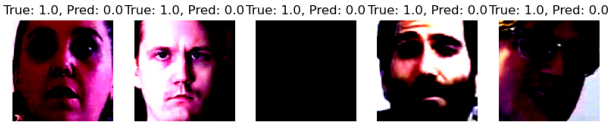

Figure 5. 5 examples of VGG-16 based CNN model misclassifications in the ”deepfake_faces” test set

7. Conclusion

In this project, I have provided an implementation of a VGG-16 based CNN model that is able to outperform the baseline CNN model in one of the two datasets (”deepfake and real images”) and achieved a accuracy, in a perfectly balanced dataset, of 76.94% and log loss of 0.5. The lack of performance in the other testing dataset (”deepfake faces”) can be mainly concluded as a poor choice in dataset selection. Though my proposed model's performance has not been competitive with other deepfake detection methods proposed by other researchers, my model still has many areas of potential improvements. These include tuning additional hyperparameter such as hidden layer sizes and number of epochs, incorporating noises during the training process, and selecting a training dataset with overall higher quality image samples. In addition, my model has integrated transfer learning in CNNs and incorporated data augmentations which may provide a potential direction for similar future research.

From this project, I have witnessed the difficulty in building a model to very accurately and effectively detect deepfake images. As deepfakes become even better, I believe that the detection process will be continuously become even more challenging.

On the topic of ML methods, I have clearly acknowledged the importance of training and testing data for a ML model. Having a reliable and high quality dataset would be highly beneficial towards the performance of a model. I have also experienced the need for high-power computing resources for ML due to the vast number of training examples and number of iterations needed to train a model (that may not even be very complex. Furthermore, the tuning and tweaking of ML models are also very time consuming yet extraordinarily beneficial to the performance of a model.

References

Agarwal, H., Singh, A., and D, R. Deepfake detection using svm. *2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pp. 1245–1249, 2021. URL https://api.semanticscholar.org/CorpusID:

237620126.

Chen, H. and Magramo, K. Finance worker pays out $25 million after video call with deepfake "chief financial officer". https://www.cnn.com/2024/02/04/asia/deepfake-cfo-scam-hong-kong-intl-hnk/index.html, Feb 2024. Accessed: 2024-02-13.

Jellali, A., Fredj, I. B., and Ouni, K. Data augmentation for convolutional neural network deepfake image detection. *2023 IEEE International Conference on Advanced Systems and Emergent Technologies (IC_ASET)*, pp. 01–05, 2023. URL https://api.semanticscholar.org/CorpusID:259217030.

Patel, Y., Tanwar, S., Bhattacharya, P., Gupta, R., Alsuwian, T., Davidson, I. E., and Mazibuko, T. F. An improved dense cnn architecture for deepfake image detection. *IEEE Access*, 11:22081–22095, 2023a. URL https://api.semanticscholar.org/CorpusID:257325844.

Patel, Y., Tanwar, S., Gupta, R., Bhattacharya, P., Davidson, I. E., Nyameko, R., Aluvala, S., and Vimal, V. Deepfake generation and detection: Case study and challenges. *IEEE Access*, 11:143296–143323, 2023b. URL https://api.semanticscholar.org/CorpusID:266268949.

Raza, A., Munir, K., and Almutairi, M. A novel deep learning approach for deepfake image detection. *Applied Sciences*, 12(19), 2022. ISSN 2076-3417. doi: 10.3390/app12199820. URL https://www.mdpi.com/2076-3417/12/19/9820.

Shad, H. S., Rizvee, M. M., Roza, N. T., Hoq, S. M. A., Khan, M. M., Singh, A., Zaguia, A., and Bourouis, S. Comparative analysis of deepfake image detection method using convolutional neural network. *Computational Intelligence and Neuroscience*, 2021, 2021. URL https://api.semanticscholar.org/CorpusID:245325392.

Yamashita, R., Nishio, M., Do, R. K. G., et al. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9:611–629, 2018. doi: 10.1007/s13244-018-0639-9. URL https://doi.org/10.1007/s13244-018-0639-9.