Preliminary and Developed Models applied on Deepfake Image Detection Based on Deep Learning

Li Wenxuan

Abstract - This report aims to use machine learning models and deep learning models to detect deepfake images, but the generalization ability of traditional models is not good enough, so we try some new methods and models to see whether they can perform better. We build *SVM* and *CNN* models based on the dataset, whose fake images are generated by *styleGAN*. In addition, we used dataset that contains multiple deepfake methods and the idea of ensemble learning to established the **EfficientNet** model as our improvement and attempt. According to our tests and evaluations, the **SVM** and **CNN** models have good predictive ability for deepfake images generated by *styleGAN*, with an accuracy rate of about **80%**. However, when two models are used to predict the data set specially used to evaluate the prediction ability and generalization ability of the model, the accuracy rate is only about **52%**. **EfficientNet** predicts that this dataset performs slightly better, with an accuracy rate of about **62%**. We think the result should be like this, because the dataset used to train EfficientNet also contains multiple deepfake methods. So, we are not very satisfied with this result. According to our analysis, the reason why the accuracy rate is still not high should be that the test data set contains some fake images generated by Photoshop, which are not deepfake images may affect the performance of the model. In future work, we will try to increase the number of ensemble models in EfficientNet and the number of deepfake methods included in the dataset, which may improve the generalization ability of the model.

Keywords: ***Deepfake image detection, Support Vector Machines(SVM), Convolutional Neural Network(CNN), Ensemble Learning,***

1 Introduction

If someone sends you a photo of him/her, do you really believe the photo is him/her? With the development of technology, we can do things we never imagined with computers. We can use computers to do a huge amount of computation, face and logo recognition, and more. Deep fake images are one of them. With the image fraud technology from Photoshop at the beginning to *GAN*, *FF++*, the detection model has also been developed accordingly. Common machine learning models used for detection include *SVM*, *KNN*, *Random Forest* and *XGBoost*, and deep learning models include *CNN* and other models.

Detection of deepfake images is an emerging field but has been well developed. Our purpose is to try out new models and new methods to achieve better performance in generalization. Our thought process is as follows:

1. We first want to build deepfake images that can be used to detect deepfakes generated by *styleGAN*. We therefore build the *SVM* model.

2. Because the feature extraction process in the *SVM* model uses the *CNN* model. Therefore, we want to try to use the *CNN* model to complete feature extraction and classification in one go, by adding fully connected layers.

3. Because the *SVM* and *CNN* models are both single models and the deepfake image used for training is generated in a single way, which is *styleGAN*. Therefore, the generalization ability of the two models is poor. To improve generalization, we build and train new models by increasing the variety of deepfake image generation methods in the data and experimenting with ensemble learning.

After building and training the three models, we will use a separate test set to evaluate the generalization and prediction ability of the three models.

2 Related Work

Deepfake detection is usually deemed a binary classification problem where classifiers are used to classify between authentic images and tampered ones. This kind of methods requires a large database of real and fake images to train such classifiers. It is still limited in terms of various detection models since the number of fake images is increasingly available.

Early attempts were based on hand-crafted features obtained from artifacts and inconsistencies of the fake image synthesis process. Recent methods have commonly applied deep learning to automatically extract salient and discriminative features to detect deepfakes. When it comes to fake image detection, methods for detecting deepfakes have been proposed as soon as this threat was introduced. Some tried spatiotemporal convolutional networks on this, others might use *LSTM-like* to detect deepfake videos. We also found someone used *SVM* to detect deepfake images by extracting features of 68 landmarks of the face region. In addition to this, we also learned someone had applied *SVM* to this task, by extracting discriminant features using bag of words method and feed these features into the model. As time goes by, someone utilized *CNN* concatenated to *CFFN*, which has two-phase procedure: feature extraction using *SVM* based on the Siamese network architecture and classification using *CNN*. To further dive into this direction, we found new *CNN-based* method named *SCnet*, which can automatically learn high-level forensics features of image data thanks to its hierarchical feature extraction block. This block is formed by stacking four convolutional layers.

3 Data Preparing

3.1 Data Source

We have a total of three datasets to use for training and evaluating the models in out project. We will introduce them separately.

1. Dataset 1

This dataset is used to build and train both the *SVM* and the *CNN* model. There are a total of 4000 images in this dataset, and the ratio of fake and real is 1:1, which means that there are 2000 images in each of the fake and real. It is worth noting that all fake images in the dataset are generated by *styleGAN* method. Figure 1 shows the distribution of 'real' and 'fake' classes in dataset 1.

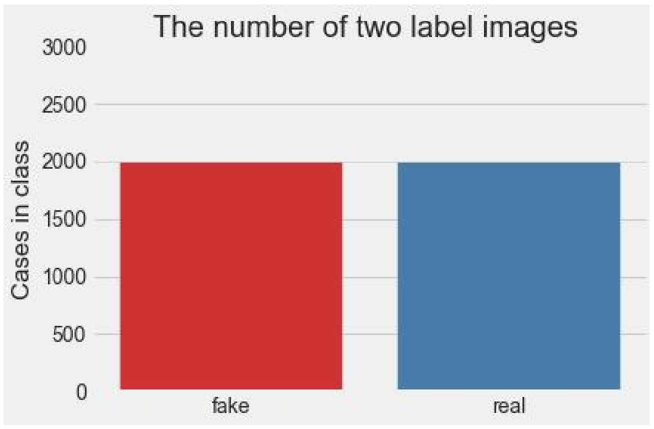

FIGURE 1. The distribution of 'real' and 'fake' classes in Dataset 1

We randomly select 16 images from the two classes of dataset 1 for display.

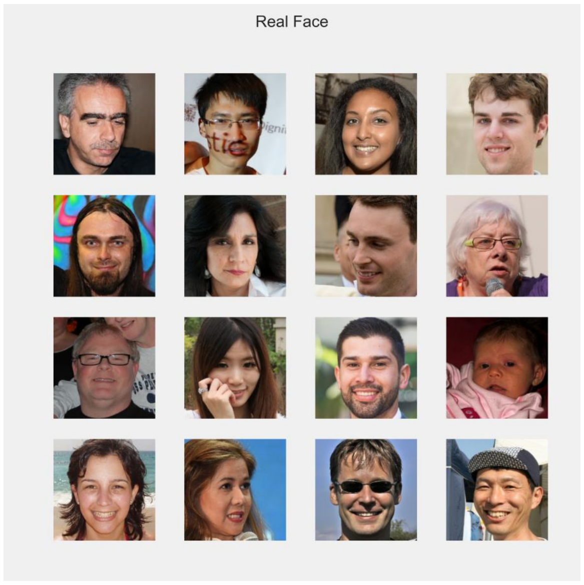

FIGURE 2. Examples of real face in Dataset 1

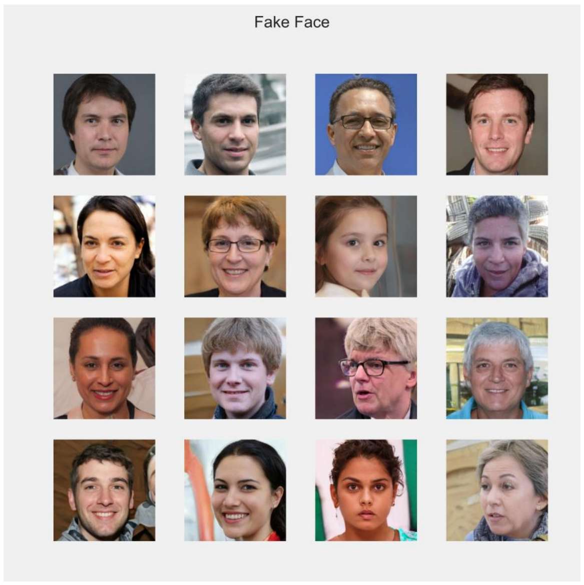

FIGURE 3. Examples of fake face in Dataset 1

2. Dataset 2

This dataset is used to train a detection model with stronger generalization ability based on ensemble learning. The dataset consists of 19,197 real images and 100,000 fake images generated using different deepfake methods based on real images. The deepfake methods for generating fake images are not single, including *styleGAN, FF++, CycleGAN* etc.

3. Dataset 3

This dataset is used to test the generalization ability of the model. The dataset contains a total of 2041 images, including 1081 real images and 960 fake images. The way of generating deepfake images is not unique, including but not limited to the methods mentioned in Dataset 1 and Dataset 2. Figure 4 and 5 shows the examples of dataset 3.

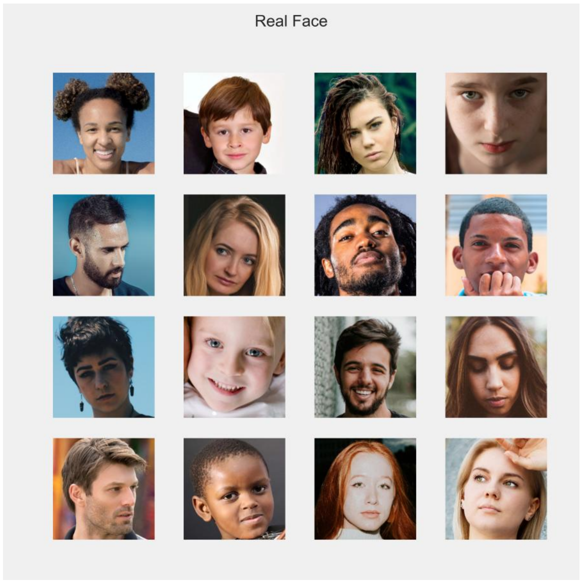

FIGURE 4. Examples of real face in Dataset 3

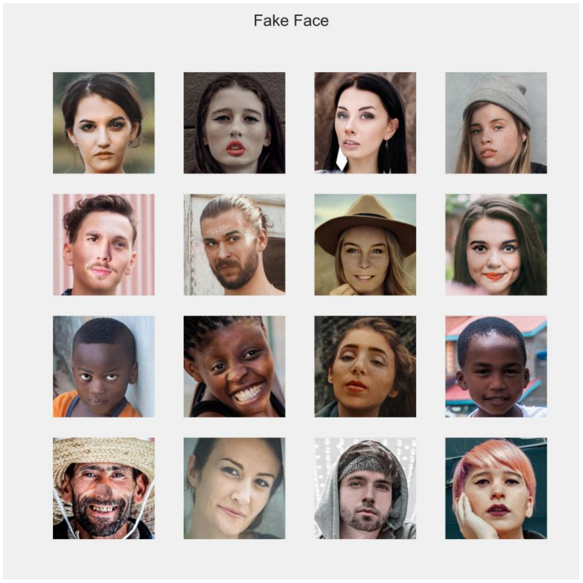

FIGURE 5. Examples of fake face in Dataset 3

3.2 Data preprocessing

1. Image size

After we read in the image, we will resize the image to (224, 224). The reason we choose 224 is that according to our related research, the feature map size generated by feature extraction is 7*7, which has the most balanced performance. If the feature map is too large, the abstraction level of the information is not high enough and the amount of computation is too large; if the feature map is too small, the information loss is severe. So, 7*7 size is the most suitable size. In addition, because the image will go through the max pooling layer in the neural network model, the size of the image is gradually reduced, and generally reduced by half each time. Therefore, the shape of the image input into the model should be $7 * 2^n$, where n is the number of max pooling layers in the model. Because the effect is better when the pixels of the image are around 300, the shape of the input image should be as close to 300 as possible. For the above reasons, we will set 5 max pooling layers in the neural network model and set the image size to (224, 224).

2. Pixel normalization

We read images in color mode, so the information of each picture will be stored in three channels [B, G, R] respectively. Therefore, the shape of the image information is (224, 224, 3). In order to speed up the convergence speed without changing the image information, we normalize the pixels, that is, the value range of the pixels is changed from [0, 255] to [0, 1].

3. Data segmentation

Because data set 1 and data set 2 are used for model establishment and training, we will divide 80% of the data into training sets in a ratio of 8:2 during this process, and the rest are test sets. The selection of data is performed at random. This is as close as possible to the chance of the second model results, but it also causes us to produce different results each time we run the program. Dataset 3 is only to evaluate the predictive ability and generalization ability of the three models. Therefore, we do not split this dataset. When making predictions on dataset 3 using the three models, the entire dataset is predicted.

4. Label encoder

The real image and the fake image are placed in folders with different names. The folder where the real image is stored is called 'real' and the folder where the fake image is stored is called 'fake'. When we read the image, the label of the image is the same as where the image is located. The label of the image is what we want to predict, but the label of string type cannot be input into the model. So, we need to convert the label. Because we want to detect whether the image is fake, we set 'fake' as the positive class and replace it with 1; 'real' set it as the negative class and replace it with 0.

4 Methodology

4.1 SVM

A support vector machine (*SVM*) is a binary classification model whose basic model is a linear classifier with the largest margin defined on the feature space, and the largest margin makes it different from a perceptron; *SVM* also includes kernel tricks, which make it become an essentially non-linear classifier. The learning strategy of *SVM* is to maximize the interval, which can be formalized as a problem of solving convex quadratic programming, which is also equivalent to the problem of minimizing the regularized hinge loss function. The learning algorithm of *SVM* is the optimization algorithm for solving convex quadratic programming.

The basic idea of *SVM* learning is to solve the separating hyperplane that correctly partitions the training dataset and has the largest geometric separation. As shown in the figure below, $\omega * x + b = 0$ is the separation hyperplane. For a linearly separable data set, there are infinitely many such hyperplanes (perceptrons), but the separation hyperplane with the largest geometric interval is unique.

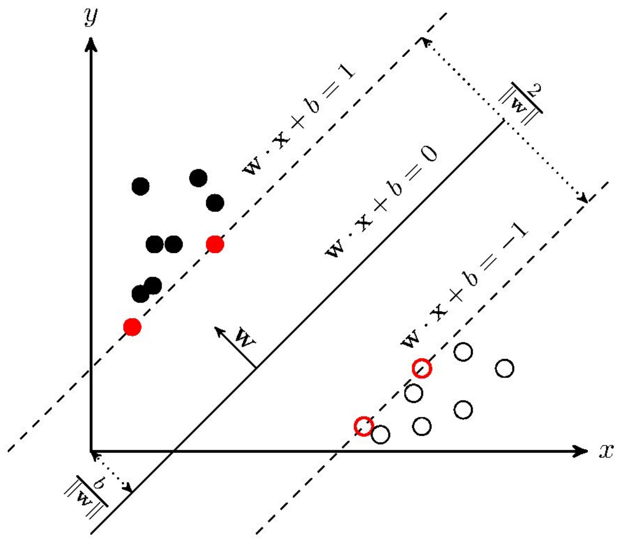
y
x
$\mathbf{w}\cdot\mathbf{x}+b=1$
$\mathbf{w}\cdot\mathbf{x}+b=0$
$\mathbf{w}\cdot\mathbf{x}+b=-1$
$\frac{2}{\|\mathbf{w}\|}$
$\frac{b}{\|\mathbf{w}\|}$
$\mathbf{w}$

FIGURE 6. The basic explanation of SVM

In this project, we just want to find a separating hyperplane that can split 'deepfake image' and 'real image' into two parts as much as possible, in order to achieve the purpose of distinguishing 'fake' and 'real'.

4.2 CNN

Now for image processing, most people tend to use convolutional neural networks (*CNN*) or deep convolutional neural networks (*DCNN*). When extracting image features, compared with common neural networks, the number of parameters that need to be trained in convolutional neural networks will be greatly reduced. Therefore, the convolutional neural network was originally used to deal with the problem of parameter explosion when processing images.

First, let's review the formula for convolution.

$$\int_{-\infty}^{\infty} f(\tau)g(1-\tau)d\tau \tag{1}$$

In simple terms, the meaning of convolution is that the output of the system at a certain time is the result of the joint action (superposition) of multiple inputs. In addition, the convolution operation is inseparable from the convolution kernel. The convolution kernel is a rule we defined. And the whole process for convolution can be understood as the input is superimposed under the action of the convolution kernel to obtain the output. For example, $21 = 1*1+0*0+2*2+2*0+1*2+4*1+2*0+3*3+1*1$.

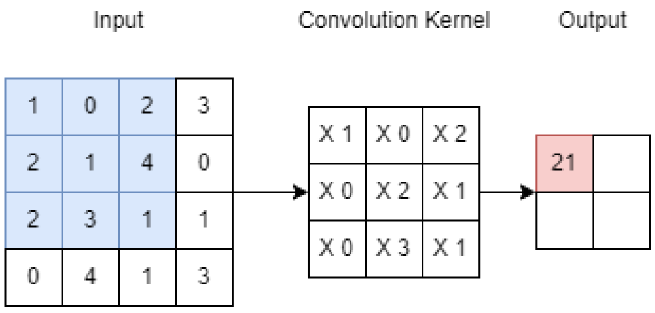

FIGURE 7. Example of convolution kernel operation

So, convolution is the process of linear superposition of the starting point and the convolution kernel. Analogy to the image, after all the action points on the convolution kernel act on the source pixel in turn, the output result of linear superposition is the output of the final convolution, which we call the destination pixel. In addition, a certain feature is also extracted, and the effect after convolution with different convolution kernels is different.

Finally, some specified features of the image can be extracted from the original image through mathematical operations with the convolution kernel.

Let's introduce the three most important layers in the *CNN* model

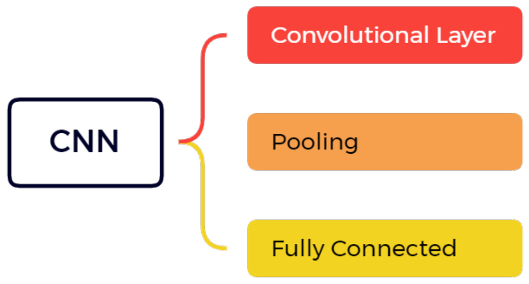

FIGURE 8. Basic structure of CNN

1. Convolutional Layer
The architecture of a convolutional neural network has some important network layers. For Convolutional Layer, it

has two very important properties, namely sparse connectivity and parameter sharing. Sparse connectivity means that the nodes of the convolution layer are only connected to some nodes of the previous layer, and are only used to learn local area features. And parameter sharing means the same convolution kernel will be convolved with different areas of the input image to detect the same features. The weight parameters on the convolution kernel are shared in space, which greatly reduces the number of parameters.

2. Pooling Layer

The pooling layer can reduce the dimension of the extracted feature information. On the one hand, it makes the feature map smaller and simplifies the computational complexity of the network; on the other hand, it performs feature compression, extracts the main features, increases translation invariance, and reduces the risk of overfitting. But in fact, pooling is more of a compromise of computing performance. While compressing features hard, it also loses some information. Polling Layer acts a bit like regularization, reducing overfitting. However, the implementation process is a bit like the Dropout layer, only recording or retaining some important information, in fact, it is also a compromise in computing power.

3. Fully Connected

After several convolutions and pooling, we will finally compress the multi-dimensional image data to "flatten", that is, compress the three-dimensional data into one-dimensional data. Then process the "compressed image information" through the fully connected layer and output the result.

From our understanding, we build a *CNN* model. This *CNN* model is obtained by improving the *Meso4* model. Its architecture diagram is shown in the following figure.

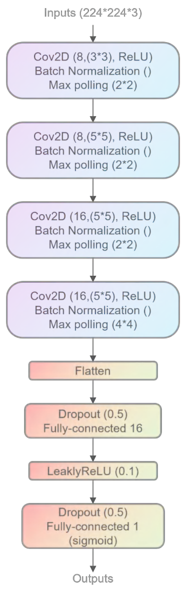

FIGURE 9. The architecture of CNN model

Initially, the shape of the image is 256*256*3, we resize the image to 224*224*3 as the input of the network. The convolutional layer is to extract the feature information of the image. The settings of the convolutional layer are the number of convolution kernels, the size of the convolution kernels and the activation function. After passing through the convolution layer, the information is passed to the BN layer, which can effectively improve the stability of training and reduce the risk of overfitting to a certain extent. The last is the pooling layer. The parameter of the pooling layer here is the size of the pooling layer, and the largest feature of each field is taken to represent the feature of the field. At the end of the model, the weight for each category is obtained through the fully connected layer and the activation function, and finally the image is classified by the weight.

4.3 EfficientNet

After searching surveys regarding deepfake detection, an ensemble *CNN* model was utilized to complete the task. Each model is trained and tested separately over both the considered datasets, *DFDC* and *FF++*. *EfficientNetB4* and *EfficientNetB4Att*, which are deep neural networks designed by M. Tan and Q. V. Le, were trained with classical end-to-end approach, together with *EfficientNetB4ST* and *EfficientNetB4AttST*, trained using the siamese strategy. And all these models derived from *EfficientNetB4* contribute to the final ensemble model. And the graph below clearly shows what *EfficientNetB4* and *EfficientNetB4Att* are.

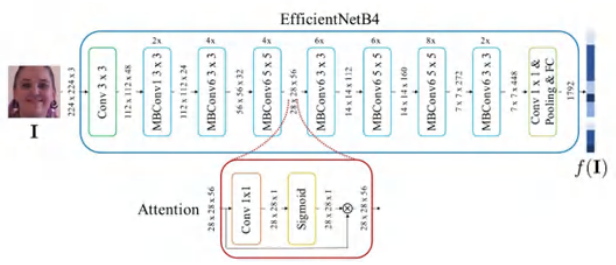

FIGURE 10. The architecture of EfficientNetB4

Then, we would explain and describe what exactly is siamese training strategy. For easy comprehension, I would say, siamese means to connect two neural networks parallelly. Not sequentially, because that's boosting. This Siamese training strategy which extracts deep features from data help the model achieve better classification performances. One can understand this structure more easily with the following figure.

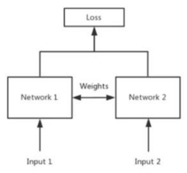

FIGURE 11. The structure of Siamese training

There are two inputs of the similar pattern, like image, text, or even audio. This method aims at comparing the similarity between two input vectors. As two similar input vectors are passed into these two models separately, representation of input vector regarding new basis are then generated by mapping these two vectors into another vector space. Thus, these two models can contribute to the final result simultaneously.

As mentioned previously, deepfake detection aims at classifying those images manipulated by some deep learning models. Basically, it is a binary classification problem. The final output of the model is either 0, or 1, which stands for real image and synthesis image, respectively. Before classifying the image into two 'real' or 'fake' classes, the model will give the score in terms of this image. A score close to 0 predicts REAL. A score close to 1 predicts FAKE. Furthermore, the final prediction score generated by these models are calculated averagely.

5 Experiment

5.1 SVM

However, because the image information does not meet the input requirements of the *SVM* function and in order to filter out valid data, we choose to perform feature extraction on the data before building the *SVM* model. Therefore, we designed a new algorithm architecture, as shown in Figure 12.

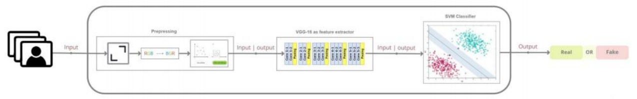

FIGURE 12. The structure of CNN+SVM

We divided the whole process into 4 parts: preprocessing, feature extraction, *SVM* establishment and parameter tuning, and prediction. We will introduce them one by one.

1. Data preprocessing

We read the image in color mode and resize the image to (224, 224). Because we complete the color reading of the image through opencv, the color information of the image will be stored in three channels [B, G, R]. So, the shape of each image is (224, 224, 3). In addition, we normalize each pixel. Then we set the label of the fake image to 1, which represents the positive class; and the label of the real image is set to 0, which represents the negative class.

2. Feature extraction

We chose *VGG16* to help with image feature extraction, a pre-trained *CNN* model. The reason for using *CNN* to accomplish feature extraction has been explained in the introduction of our second method. Following is the architecture of *VGG16*.

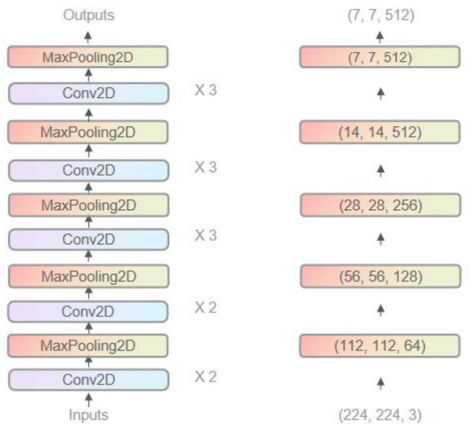

FIGURE 13. Architecture of VGG16

Because we only use *VGG16* to extract the features of the image to obtain the feature map, we remove its fully connected layer and set the model state to untrainable, and then directly use the parameters in the pre-trained model in the subsequent feature extraction work. From the left part of Figure 13, we can see that *VGG16* consists of 5 convolutional layers and the layers are separated by max-pooling. The activation units of all hidden layers use the *ReLU* function. The right side of Figure 13 shows the shape after each max-pooling. After feature extraction by *VGG16*, the shape of the data changes from (224, 224, 3) to (7, 7, 512).

We can see that the shape of the feature extraction data has changed. However, the existing data still does not meet the input requirements of *SVM*. Therefore, we stretch (7, 7, 512) into a one-dimensional tensor, and the shape will become (25088). The data at this time can be directly input into the *SVM* model for training or testing.

3. SVM establishment and parameter tuning

We first built the *SVM* model with default parameter settings. Then use it to make predictions on the training set as well as the test set. Figure 14 shows the confusion matrix of the preliminarily established *SVM* model for the prediction results of the training set and the test set. Table 1 shows the evaluation indicators of the model's prediction results for the corresponding data set.

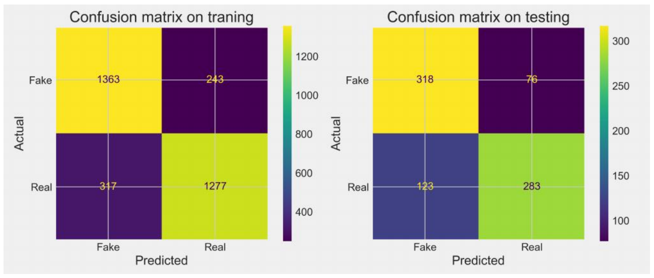

FIGURE 14. Confusion Matrix of SVM model without tunning

TABLE I. Statistics for SVM model without tunning

Statistic	Prediction on Training	Prediction on Test
Accuracy	0.8250	0.7513
Precision	0.8113	0.7210
Recall	0.8487	0.8071
F1 score	0.8296	0.7617

We can see that the *SVM* built with the default parameter settings performs well without overfitting or underfitting. But we want to further improve the predictive power of the model without overfitting. For this purpose, we tried to tune the hyperparameters using grid search and cross validation. During the adjustment process, we mainly adjusted three parameters: ‘C’: Degree of correct classification. (higher C value means smaller margins = fewer misclassifications); ‘kernel’: The type of hyperplane used to separate the data. (’linear’ = linear hyperplane, ’rbf’ = nonlinear hyperplane); ‘gamma’: A parameter of ’rbf’ indicating how well the model fits the training data. (If the gamma value is too high, it is easy to cause the model to overfit).

In the GridSearchCV() function, we set the value range of C to [0.01, 0.1, 1, 10, 100]; kernal chooses from [’linear’, ’rbf’]. When kernal is selected as ’rbf’, the parameter ’gamma’ exists, and the value range is [1, 0.1, 0.01, 0.001, 0.0001]. In addition, our selection criterion is the accuracy rate, which means that after gridsearch will get the highest accuracy parameter grid. The result of tuning as shown in Table 2.

TABLE II. The hyperparameters of SVM Model with tunning

Hyperparameter	Value
C	10
Kernel	rbf
gamma	0.0001

We rebuild the *SVM* model and re-predict the training set and test set using the parameter grid determined after parameter tuning. The confusion matrix is shown in Figure 15. The statistics used to evaluate predictive performance are shown in Table 3.

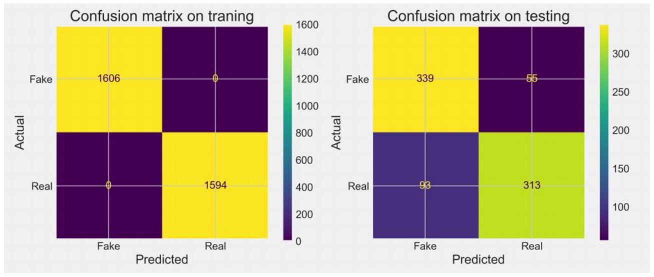

FIGURE 15. Confusion Matrix of SVM model with tunning

TABLE III. Statistics for SVM model with tunning

Statistic	Prediction on Training	Prediction on Test
Accuracy	1.0	0.8150
Precision	1.0	0.7847
Recall	1.0	0.8604
F1 score	1.0	0.8208

We can see that the accuracy, recall, precision, and F1 score of the *SVM* model after parameter adjustment for the prediction results of the training set are all 1. However, after the parameter adjustment, the prediction effect of the *SVM* model for the test set has also been compared. big boost. We believe that the model has an overfitting problem, but it does not affect the predictive ability of the model for the test set.

4. Prediction

After testing, although our *SVM* model has the problem of overfitting, it still has a good predictive ability for the test part of this dataset. The problem of overfitting is common for deepfake image detection models, but this does not mean that the prediction ability of the model is poor, which means that the model has better prediction performance for the same type of fake images as the images in the dataset. Here, we believe that the adjusted *SVM* model has a good prediction effect on the deepfake image generated by *styleGAN*.

5.2 CNN

After building the network, we first randomly select 80% of the data for training and 20% for testing. In our CNN model, we train it with MSE loss and choose Adam as the optimizer. In addition to this, we set the values of batch size and epochs to 10 and 80, respectively. The training loss is shown in Figure 16. The statistics used to evaluate forecast performance are shown in Table 4.

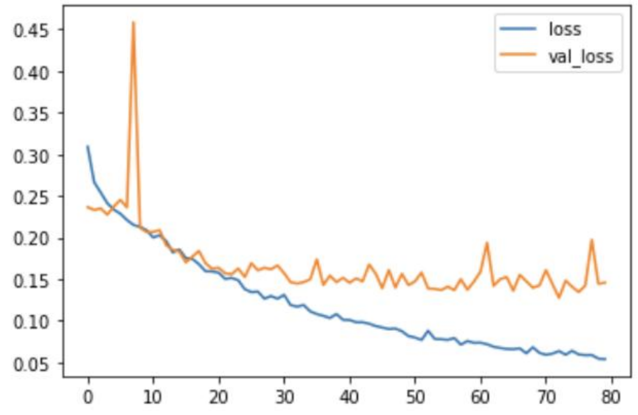

FIGURE 16. The relationship between epoch and loss in CNN

TABLE IV. Statistics for evaluation CNN

Statistic	Prediction on Test
Accuracy	0.7975
Precision	0.7724
Recall	0.8589
F1 score	0.8208

5.3 EfficientNet

At the initial stage, we assume that, given four CNN models, the performance of the combined models is improved as more models are combined. Here is the experimental result on the test dataset that also belongs to DFDC.

TABLE V. The AUC of different combination

B4	B4Att	B4St	B4AttSt	AUC
✓				0.8766
	✓			0.8658
		✓		0.8642
			✓	0.8360
✓	✓			0.8800
✓		✓		0.8785
✓			✓	0.8729
	✓	✓		0.8760
	✓		✓	0.8642
		✓	✓	0.8625
✓	✓	✓		0.8813
✓	✓		✓	0.8769
✓		✓	✓	0.8751
	✓	✓	✓	0.8719
✓	✓	✓	✓	0.8782

It is worth noting that the strategy of model ensembling generally awards in terms of performances. As somehow expected, best top-3 results are always reached by a combination of 2 or more networks, meaning that network fusion helps both the accuracy of the deepfake detection and the quality of detection.

Because the EfficientNet model is a pretrained model, we no longer use dataset 2 to train and test it. We use it directly to make predictions on dataset 3. Yet the accuracy increased when we classify deepfake images which belongs to the same dataset that trained by these four models, after being tested on an unseen dataset (Dataset 3), we learned that four models that trained separately perform similarly. Here is the performance comparison graph of four models.

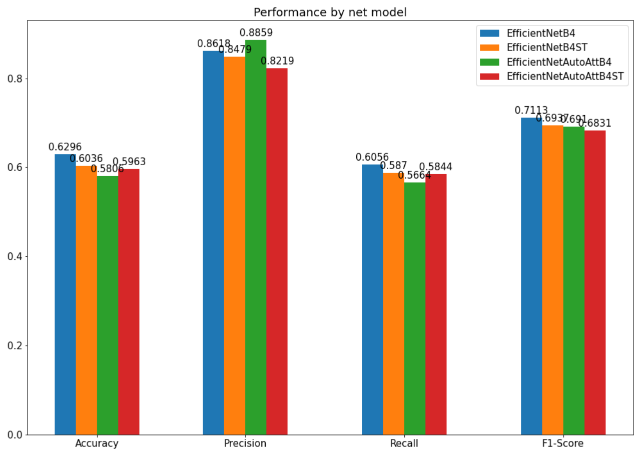

FIGURE 17. The performance of different net models inEfficientNet

Because we do not have that much resources to replicate the experiment details provided by the author, we could only use the pre-trained models to do a simple comparison. Thus we can say that generalization capability of each efficient net are quite similar.

6 Result

6.1 Evaluation

After being trained on the dataset 1, these two models *SVM* and *CNN* gave their prediction with respect to dataset 3. To make a comparison on their performances, we used four measurements, which are accuracy, precision, recall, f1-score to evaluate these two. As shown in Table 6.

TABLE VI. SVM and CNN predict performance on dataset 3

Statistic	SVM	CNN
Accuracy	0.5284	0.5031
Precision	0.4919	0.3532
Recall	0.0635	0.0677
F1 score	0.1125	0.1136

However, the results obtained were under expectation. Their generalization capabilities are not acceptable, but reasonable. Because both of these models were trained on one dataset which was generated by only one deepfake method, *StyleGAN*. Namely, both of them haven't seen any other deepfake methods, or, images manipulated by human. Besides, it would be easier for artificial model to learn synthesis images generated by other designed model, and it might be a little bit hard for a model to extract tampered features emerged in those pictures. Thus, here we proposed to use another pre-trained ensemble model, *EfficientNetB4*, to finish this task. As mentioned before, it had been trained on a dataset, whose deepfake methods consist of *FSGAN*, *DFAE*, *NTH*, *StyleGAN* etc. And you can see the comparison below, the generalization ability of the *EfficientNetB4* is the best among the three models. And this is telling us that, generalization capability of the model can definitely be improved with averaging results calculated from more models as they might extract various features. Furthermore, dataset is also crucial for training a model as it is impossible for model to predict what deepfake images it will meet in the future. Thus, dataset should include more different deepfake methods.

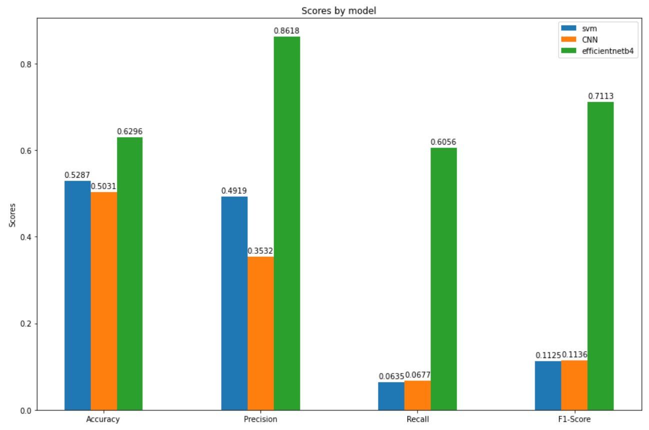

FIGURE 18. 3 models predict performance on dataset 3

6.2 Conclusion

After the previous part and the evaluation of the generalization ability, we can see that the *SVM* and *CNN* models have a good ability to discriminate the deepfake images generated by the *styleGAN* method. But both models performed poorly on datasets containing multiple methods for generating deepfake images. The generalization ability of *EfficientNet* has been enhanced compared to *SVM* and *CNN*, but it has not yet reached our expectations. After our analysis, we believe that there are two important reasons:

1. The dataset used to evaluate the generalization ability contains fake images generated using Photoshop, which are not deepfake images. In addition, the deepfake method contained in the dataset is not comprehensive enough.

2. There can be problems with the architecture of the ensemble learning model.

7 Future Work

We will collect more image data, including more deepfake images generated by different deepfake methods. In addition to this, we will try to tune the architecture of the ensemble learning model and the models used in it. Maybe using a single *CNN* model is not the best choice.

8 Reference

1. Tan, M., Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning (pp. 6105-6114). PMLR.

2. Bonettini, N., Cannas, E. D., Mandelli, S., Bondi, L., Bestagini, P., Tubaro, S. (2021, January). Video face manipulation detection through ensemble of cnns. In 2020 25th International Conference on Pattern Recognition (ICPR) (pp. 5012-5019). IEEE.

3. Karras, T., Laine, S., Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 4401-4410).

4. Chang, X., Wu, J., Yang, T., Feng, G. (2020, July). DeepFake face image detection based on improved VGG convolutional neural network. In 2020 39th chinese control conference (CCC) (pp. 7252-7256). IEEE. IEEE.

5. Rana, M. S., Sung, A. H. (2020, August). Deepfakestack: A deep ensemble-based learning technique for deepfake detection. In 2020 7th IEEE international conference on cyber security and cloud computing (CSCloud)/2020 6th IEEE international conference on edge computing and scalable cloud (EdgeCom) (pp. 70-75). IEEE.

6. Amerini, I., Galteri, L., Caldelli, R., Del Bimbo, A. (2019). Deepfake video detection through optical flow based cnn. In Proceedings of the IEEE/CVF international conference on computer vision workshops (pp. 0-0).

7. Güera, D., Delp, E. J. (2018, November). Deepfake video detection using recurrent neural networks. In 2018 15th IEEE international conference on advanced video and signal based surveillance (AVSS) (pp. 1-6). IEEE.

8. Zhao, H., Zhou, W., Chen, D., Wei, T., Zhang, W., Yu, N. (2021). Multi-attentional deepfake detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 2185-2194).