2024 IEEE AITU: Digital Generation
4-5 April, 2024, Astana, Kazakhstan

Using the CNN architecture based on the EfficientNetB4 model to efficiently detect Deepfake images

Arman Zhalgasbayev
Sofrware Engineering Student
Astana IT University
Astana, Kazakhstan
220650@astanait.edu.kz
https://orcid.org/0000-0001-8507-6845

Tolegen Aiteni
Computer Science Student
Astana IT University
Astana, Kazakhstan
221640@astanait.edu.kz
https://orcid.org/0009-0007-8353-1382

Nursultan Khaimuldin
Master of Technical Sciences, Senior lecturer of Computer Engineering
Educational Program
Astana IT University
Astana, Kazakhstan
n.khaimuldin@astanait.edu.kz
https://orcid.org/0000-0002-3767-4418

Abstract: This study evaluates the effectiveness of using the EfficientNetB4 model architecture for deepfake image detection and identifies the main challenges in developing an accurate model for deepfake image detection. Deepfake image detection is a complex task that requires lots of data, advanced models and new approaches for handling features in face images which shows that image is real or fake. This research is based on the Deepfake Detection Challenge (DFDC), which took place in 2019-2020, where world experts showed new approaches to solving this problem. During the research and development of the model, the best practices and experience of experts were used to identify the deepfake images.

Keywords: *Deepfake, Deep Learning, EfficientNet, CNN, TensorFlow*

I. INTRODUCTION

In today's digital world, the increase in fake images is causing worry, as these images can spread false information and reduce people's trust in the media space. In 2017, a new term "Deepfake" was introduced to denote synthetic generated fake images created by deep learning technology [1]. The technology is widely used for realistic face replacement in photos and videos, known as face-swapping. The use of face-swapping techniques by scammers and detractors has turned into powerful tools for manipulating people, financial fraud, and political subversion [2]. Using another person's face, without special permission, for any personal, business, political or other purposes violates principles of ethics of artificial intelligence [3]. Thus, the rapid development of artificial intelligence technology has given a new boost to improving the processing of images using a machine learning techniques such as Autoencoders and "Generative Adversarial Network" (GAN), which makes it difficult to distinguish real face images from fake ones.

To address this problem, we are using modern deep learning methods utilizing convolutional neural networks (CNN) based on the architecture of the EfficientNetB4 model from TensorFlow, which is one of the best architectures for recognizing features in images [4]. The main purpose of our study is to explore the potential dangers that can be caused by deepfake images to develop a competitive model for identifying fake images from real ones.

understand the originality and the value of the work. Identify applicable sponsor/s here[1]. If no sponsors, delete this footnote.

Technology corporations like Meta, Amazon and Microsoft are making every effort to stop deepfake, to ensure a safe media space on the Internet. In the period from 2019 to 2020, together, these companies held one of the largest Data Science competitions called "Deepfake Detection Challenge" (DFDC). They provided a dataset of more than 100,000 training data so that participants could create a high-quality model for identifying deepfake images. The competition was attended by 2,114 participants who developed a total of more than 35,000 models, the best of which showed 82% accuracy on test data [5]. In the process of research and development, we relied on data obtained from DFDC, analyzed the results of this competition, studied solutions that have shown high performance and used the experience of the best experts in the world.

II. METHODOLOGY

In this research project, we used an iterative research method to cyclically identify inaccuracies and correct them to improve the overall quality of the project. Intermediate experiments with image augmentation, preprocessing of image matrices, changes in features and hyperparameters allowed us to obtain a relatively better result. During the development process, we were based

[1] Microsoft Imagine Cup 2024.

2024 IEEE AITU: Digital Generation | 979-8-3503-6437-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/IEEECONF61558.2024.10585385

on the ML model training pipeline using TensorFlow, shown below in the fig. 1.

Fig. 1. ML Model Training Pipeline with TensorFlow Diagram [Application – 1]

- The used version of TensorFlow: 2.13.0
- Used accelerator: GPU T4 x2
- Used Environment: Kaggle Notebooks

A. Deepfake Images Generation

To effectively combat the problem of synthetic images, we first need to understand the mechanism underlying the creation of deepfake images. This section discusses two key tools used in this area:

1) Autoencoders

An autoencoder is one of the well-known methods for creating deepfake images using a neural network that receives an input image and compresses it into a lower-dimensional representation called a latent code. Then it modifies the latent code and tries to restore the original image from it in such a way that the restored image closely resembles the original [6].

2) Generative Adversarial Network (GAN)

Generative Adversarial Network (GAN) is an advanced deep learning method for generating synthetic images using 2 types of neural networks: a generator and a discriminator. The generator creates images while the discriminator evaluates these images and tries to determine whether this image is real or not. Active cross-training helps the generator to adapt to the real image to the level of generating realistic deepfake images that are difficult to detect [7].

B. Deepfake Detection Contest (DFDC) Key findings

Before starting work on training a new model for identifying deepfake images, we analyzed the top-performing models and approaches from DFDC competition [5].

1. Effective Use of EfficientNet Architecture: most of the top-performing models used the architecture of EfficientNet model in various variants.

2. Absence of Forensic Methods in Top Solutions: the use of digital forensics methods at the pixel level leads to degradation of the effectiveness of the model.

3. Low Average Performance of Models: Although more than 2,000 experts from all over the world participated in the competition, no one was able to achieve 70% accuracy on private test data.

C. Data Preparation

As the main dataset, we used preprocessed data from the general dataset of the DFDC competition. Frames with clear outlines of the face were cut from all the images with a modified size of 224x224 pixels using "OpenCV" image processing tools and "face_recognition" module [8].

Initially, the dataset consisted of 95,634 images of 224x224 sized faces. More detailed metadata summary shown in the table 1.

	dtypes	count	unique	missing#	missing%
videoname	object	95634	95634	0	0.000000
original_width	int64	95634	31	0	0.000000
original_height	int64	95634	34	0	0.000000
label	object	95634	2	0	0.000000
original	object	79341	16955	16293	0.170368

Table 1. Metadata Summary Information.

1) **Distribution of real and fake images:**
   - Fake Images: 79,341
   - Real Images: 16,293

2) **Image size in bytes:** 224 * 224 = 50,176 B (49 KB)

3) To equally distribute images, we choose 16,000 real and 16,000 fake images.
   - **Total images for training** = 16,000 (real) + 16,000 (fake) = 32,000 images.

4) **Total dataset size** = 49 KB * 32,000 = 1,568,000 KB (or 1,531 MB)

It's important to note that the total dataset size is approximately 1,531 MB (~ 1.5 GB), which poses a significant memory challenge especially for resource-constrained environments such as Kaggle Notebooks. Thus, we need to maximize the efficiency of our code for training. This includes optimizing data loading, preprocessing, and model training steps to ensure smooth execution within the limitations of our environment.

D. Exploratory Data Analysis

Splitting data into training, testing and validation sets:

- Training Set: 17,920 (56%) – used to train the machine learning model.
- Testing Set: 6,400 (20%) – used to evaluate the final performance of the trained model.
- Validation Set: 7,680 (24%) – used to tune hyperparameters.

Fig. 2. shows distribution of classes in each data set, while fig. 3. shows random sample of 15 images from the dataset of facial images.

Fig. 2. Distribution of classes in each set.

Fig. 3. Sample face images from Training Set. Credits: DFDC.

E. Data Preprocessing

1) Before data preprocessing sets were splitted into X – features and y – target values.
2) "Random state" parameter was set to a certain value (42) to ensure reproducibility.
3) "Batch size" value was set to 16, which means that in each training iteration 16 samples will be processed.
4) As a preprocessing function EfficientNet model's built-in function was used. This preprocessing function includes normalization and scaling to prepare input images for the model.
5) Convert the training, validation and testing datasets into TensorFlow datasets;
6) Preprocess and batch datasets:

- "Shuffle" function shuffles the training dataset to introduce randomness during training stage.
- "Batch" function batches the dataset into batches of size "Batch size".
- "Prefetch(1)" prefetches one batch of data to improve training performance by overlapping data preprocessing and model execution.

F. Model Architecture

Architecture of our model called "ReDeepFake" utilizes EfficientNetB4 advanced CNN architecture with a custom-designed additional layers. Model architecture diagram shown below in fig. 4:

Fig. 4. «ReDeepFake» model architecture diagram [Application – 2].

Main model architecture: The model architecture comprises the EfficientNetB4 convolutional neural network, a powerful pre-trained feature extractor designed for image classification tasks [9]. The base model is initialized with weights from the "ImageNet" dataset, providing a strong foundation for learning hierarchical features [10].

Additional layers:

- **CV2 and EfficientNet preprocessing** stages for preparing data for processing in the main model architecture.
- **Global average pooling layer** is employed to condense the spatial dimensions of the extracted features, creating a global context for the subsequent dense layer.
- **Dense layer with a sigmoid activation function** generating a single output representing the probability of the input image being a fake image (1 – fake, 0 – real).

The model is compiled using the stochastic gradient descent (SGD) optimizer, with a learning rate of 0.01 and momentum of 0.9, chosen to strike a balance between rapid convergence and stability.

"Binary_crossentropy" loss function commonly used for binary classification problems due to its simplicity and efficiency. It measures the difference between the predicted probability distribution and the true binary labels, thereby improving the accuracy of the model at each iteration.

"Accuracy" score metric is widely used for evaluating classification models. It measures the proportion of correctly classified samples out of the total number of samples.

G. Model Training

Having prepared the data in the right format, configured the architecture of the model for training, and all additional functions, we launched the training process for 47 epochs. Choosing 47 epochs for training was not a random decision, we checked the accuracy of the model on test sets, using different epochs, the size of the training data and the batch size [11]. Table 2 shows the results for 1-5 model versions for various settings of parameters.

Table 2. Table of results for 1-5 model versions to get the best settings [Application – 3].

H. Model Evaluation

In the figures fig. 5, fig. 6 and fig 7 below, you can see the results of training the model based on historical data on the values of accuracy and loss in training and validation sets.

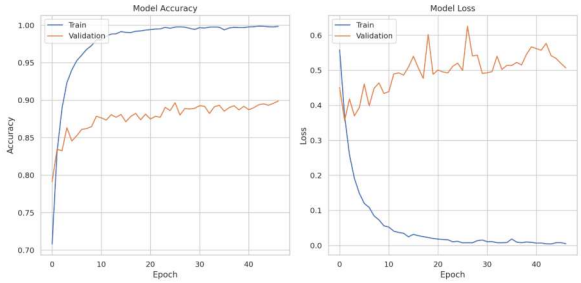

Fig. 5. V1.3 model performance evaluation plot.

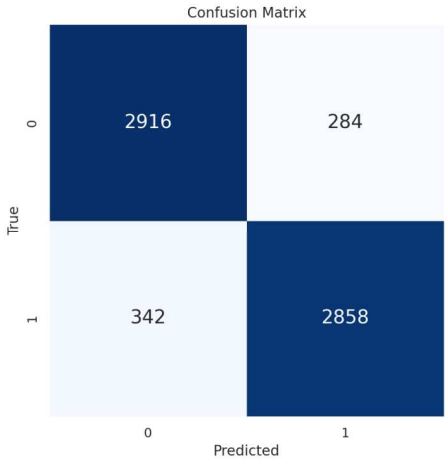

Fig. 6. V1.3 model Confusion Matrix.

Classification Report:

	precision	recall	f1-score	support
0	0.90	0.91	0.90	3200
1	0.91	0.89	0.90	3200
accuracy			0.90	6400
macro avg	0.90	0.90	0.90	6400
weighted avg	0.90	0.90	0.90	6400

Fig. 7. V1.3 model Classification Report.

According to the results of the evaluations, we have obtained one of the most optimal and balanced models for identifying deepfake images with an overall accuracy of 0.90 (approximately) and F1-Score of 0.90 (approximately) on test sets, with minimal value of validation loss in the process of training of 0.4872.

III. Results and Analysis

A. *Comparison and Analysis*

It is difficult to compare the performance of different models for identifying deepfake images, since there is no single system and method for testing the performance of models to adequately compare and identify the best models [12]. Therefore, as a benchmark for training and testing the model, we used data from the DFDC competition. Since we do not have access to hidden tests from Meta, we cannot say how much our model is worse or better than the models that were at the competition. But regardless of this, we achieved excellent results in 90% accuracy and 90% f1-score on test sets isolated from the general dataset consisting of 32,000 images. Fig. 8 shows an example from a random sample of 20 images from the test set and their results in the format "Predicted class | Score | Correct or not".

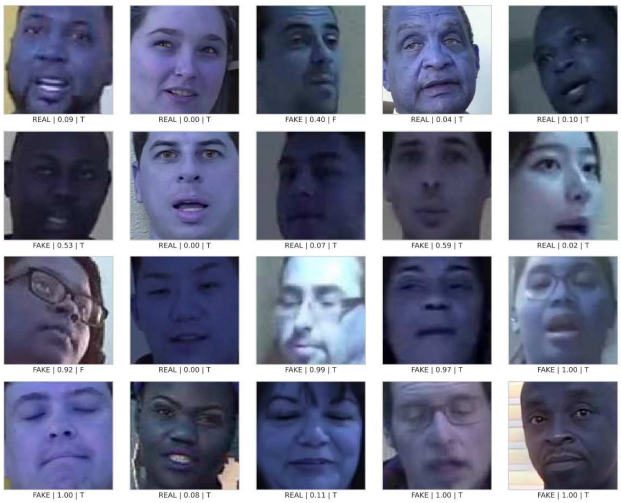

Fig. 8. V1.3 model results on the random 20 test images.

The results are impressive, from a random set of 20 images we got 18 correct and only 2 incorrect predictions. Neural networks derive their own logic and rules for predicting some results based on given data. Therefore, it is difficult to understand by what criteria the model determines that the images are real or fake. But we tried to analyze and find some presumptive features of the predictions using the example of some images from the test set.

B. *Presumptive Features of the prediction*

1) The problem of generating deepfakes on the face with glasses or other objects on the face shown in the fig. 9:

Fig. 9. Example 3 fake images showing the problem of a deepfake with glasses.

2) The obvious disproportionality of a person's face shown in the fig. 10:

Fig. 10. Example 3 fake images showing the problem of a deepfake of disproportionate faces.

3) The quality of the image, the level of illumination [13], the features of the location and facial expressions strongly affect the prediction results [14].

C. *Areas of Usage*

1) **Media Integrity Verification:** Verify the authenticity of images and videos in the media industry to ensure integrity and trustworthiness.
2) **Social Media Content Moderation:** Enhance content moderation on social media platforms by detecting and preventing the spread of deepfake content.
3) **Security and Surveillance:** Strengthen security and surveillance systems by identifying manipulated visuals that could compromise the accuracy of facial recognition and surveillance technologies.

D. *Examples shown in the figures fig. 11 and fig. 12.*

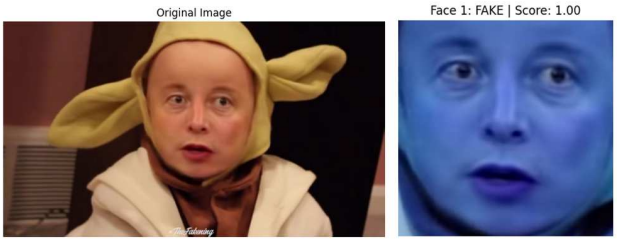

Fig. 11. Example of predicted deepfake image. Credits: Elon Musk (Baby Yoda Deepfake).

Fig. 12. Example of predicted real image. Credits: Margulan Seisembayev (Gordon).

IV. CONCLUSION

Having qualitatively completed all the stages of research and development of the deepfake image detection model, we achieved impressive results of 90% accuracy and 90% f1-score on the test set. The result of the 1.3 version of the model is approximately the same as the results of the best models in the DFDC competition, but a direct comparison of the models is not correct since we did not have access to the full amount of data and hidden test data based on which these models were tested. We have created a production-ready model that can be used as a descriptor to identify fake images in various fields such as: media integrity verification, social media content moderation, and security. The development of models based on the GAN architecture makes it very difficult to deal with deepfake content, since they use descriptor models created to fight against them to develop better models for generating deepfake images. Such technologies create images that are difficult to distinguish from a real person so that neither a machine nor a person can determine it. These are the three main reasons why many deepfake image detection models perform poorly at the production level:

The limitations of most models are:

1) The model's performance may be influenced by variations in lighting conditions, image quality, and diverse facial expressions.
2) Model may not be fully robust against emerging deepfake generation techniques: Modern image generation methods (GAN) use advanced descriptors to assess the quality of the image's realism, so photos of such faces are difficult to distinguish from real people.
3) Deepfakes made by using 3D image processing technologies and manually modified images by the authors cannot be recognized correctly by the model.

The results of our research work can help further develop technology in the field of deepfake content detection and improve approaches to the development of new models.

REFERENCES

[1] Paine, L. (2024, Jan 16). Deepfake: AI-generated synthetic media. *The encyclopedia of Britannica.* Retrieved from https://b2a.kz/6mk

[2] Committee On Oversight and Accountability (2023, Nov 8). Mace: Deepfake Technology Can Be Weaponized to Cause Harm. Retrieved from https://b2a.kz/QSn

[3] Office of the Director of National Intelligence (n.d.). Principles of Artificial Intelligence Ethics for The Intelligence Community. Retrieved from https://b2a.kz/86p

[4] Tan, M., & Le, Q. (2019, May 28). Efficientnet: Rethinking model scaling for convolutional neural networks. *International conference on machine learning* (pp. 6105-6114). PMLR. Retrieved from https://b2a.kz/GqC

[5] Meta (2020, Jun 12). Deepfake Detection Challenge Results: An open initiative to advance AI. Retrieved from https://b2a.kz/hLe

[6] Bank, D., Koenigstein, N., & Giryes, R. (2023, Feb 26). Autoencoders. *Machine learning for data science handbook: data mining and knowledge discovery handbook*, 353-374. Retrieved from https://b2a.kz/Gvh

[7] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018, Jan 10). Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1), 53-65. Retrieved from https://b2a.kz/4ML

[8] Dagnelies (2020). "deepfake_faces" dataset. Kaggle Datasets. Retrieved from https://b2a.kz/k24

[9] TensorFlow (n.d.). Module: tf.keras.applications.efficientnet.EfficientNetB4. *TensorFlow API Docs.* Retrieved from https://b2a.kz/A71

[10] Stanford Vision Lab, Stanford University & Princeton University (2020). ImageNet. ImageNet website. Retrieved from https://b2a.kz/TLr

[11] Kirushikesh DB (2022). "Deep Fake Detection on Images and Videos". Kaggle Notebooks. Retrieved from https://b2a.kz/sai

[12] Korshunov, P., & Marcel, S. (2020). Deepfake detection: humans vs. machines. *arXiv preprint arXiv:2009.03155*. Retrieved from https://b2a.kz/ULc

[13] Gerstner, C. R., & Farid, H. (2022). Detecting real-time deep-fake videos using active illumination. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 53-60). Retrieved from https://b2a.kz/Fz5

[14] Mazaheri, G., & Roy-Chowdhury, A. K. (2022). Detection and localization of facial expression manipulations. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (pp. 1035-1045). Retrieved from https://b2a.kz/HOU

[Application – 1]

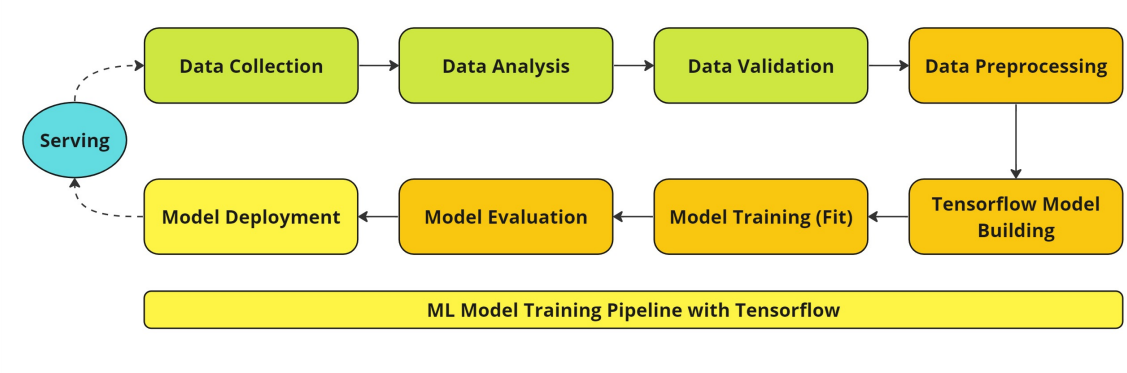

Fig. 1. ML Model Training Pipeline with TensorFlow Diagram

[Application – 2]

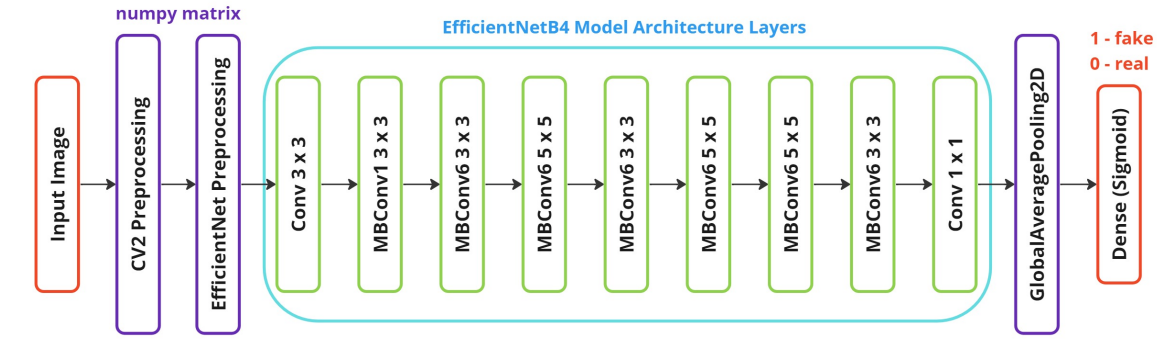

Fig. 4. «ReDeepFake» model architecture diagram.

[Application – 3]

Version	**Dataset** *(images)*	**Epochs**	**Batch Size**	**EarlyStoppage**	**Data Augmentation**	**Accuracy Score** *(Test Set)*	**Loss** *(Test Set)*
1.1	16,000	120	32	No	No	0.8531	0.9521
1.2	16,000	12	32	No	No	0.8739	0.5399
1.3	32,000	47	16	No	No	0.9022	0.4872
1.4	32,000	47	16	Yes	No	0.8395	0.3455
1.5	32,000	47	16	No	Yes	0.8936	0.3751

Table – 2: Table of results for 1-5 model versions to get the best settings.

[Application – 4]

Github Repository: https://github.com/silvermete0r/redeepfake-demo-app

[Application – 5]

Kaggle Notebook: https://www.kaggle.com/code/armanzhalgasbayev/deepfake-detection-efficientnetb4-tf-cnn

[Application – 6]

ReDeepFake Demo Application: https://huggingface.co/spaces/dataflow/redeepfake-demo