sciendo

Land Forces Academy Review
Vol. XXX, No. 2(118), 2025

CONVOLUTIONAL NEURAL NETWORK FOR DEEPFAKE CONTENT DETECTION

Annamaria SÂRBU
"Nicolae Bălcescu" Land Forces Academy, Sibiu, Romania
sarbu.annamaria@armyacademy.ro

Andrea ROTONDO
"Nicolae Bălcescu" Land Forces Academy, Sibiu, Romania
s.rotondo.andrea@armyacademy.ro

ABSTRACT

This paper presents the development, training, and performance evaluation of a customized 4-layer Convolutional Neural Network (CNN) for classifying deepfake and real images. The model was trained on Google Colab, using a publicly available dataset. Data augmentation techniques were applied to increase the size and diversity of the training dataset. The resulting accuracy on the test dataset was 97.5 %, indicating strong performance in distinguishing manipulated images from real ones. A comparison with other existing deepfake detection models, including DenseNet, Xception, EfficientNet, and a similar 5-layer CNN, revealed the superior accuracy and runtime efficiency (4m 44s/epoch) of the proposed CNN. Additionally, the proposed model was adapted for video classification, by implementing a custom designed algorithm based on analyzing each 10th frame of a video and averaging the predictions to determine a final verdict on the video's authenticity. The results suggest that the customized 4-layer CNN outperforms predefined models, offering a promising solution for real-time deepfake detection tasks.

KEYWORDS: Convolutional Neural Network, deepfake, image classification

1. Introduction

Deepfake is a form of synthetic media in the format of visual/audio data, created with the use of artificial intelligence (AI). Deepfake technology is based on deep learning techniques, such as generative adversarial networks (Goodfellow et al, 2020), which generate, evaluate and further improve the realism of the desired content (human faces, voices or even full video sequences).

Driven by the advancements in AI, deepfake technology has become more elaborate, producing synthetic media that appears to be original. The presence of fake content in the form of news, images or videos, disseminated mainly on social media platforms has become a significant challenge affecting individuals, states and organizations altogether. On social media platforms, content spreads rapidly, often without proper verification, as users tend to share information that aligns with their beliefs without fact-checking.

Fake content totally changes the way information is created, shared and consumed, as it can influence public opinion, spread harmful chronicles, and even affect political processes (Kharvi, 2024). The consequences related to the spread of deepfake are associated with erosion of trust (Vaccari & Chadwick, 2020), social polarization driven by fuelling conflicts in the online environment or even damage of reputation for

DOI: 10.2478/raft-2025-0029

individuals and organizations (Ghodke, 2024). By blurring the line between authentic and fabricated media, deepfake, a direct application of AI, becomes both an emerging and a disruptive technology affecting communication, media consumption and digital security (Sudhakar & Shanthi, 2023).

The current escalation of deepfake content has led to the development of various detection methods to combat the spread of false information and protect media integrity. According to the published literature, deepfake detection techniques include deep-learning based methods, classical machine learning, statistical techniques, and blockchain-based approaches (Rana, Nobi, Murali & Sung 2022).

Deep learning techniques, particularly those using neural networks, have shown superior performance in detecting deepfakes. Methods such as Xception and MobileNet have achieved high accuracy rates, ranging from 91% to 98% across various datasets (Pan et al., 2020). According to (Heidari, Navimipour, Dag & Unal, 2023) Convolutional Neural Networks (CNNs) are frequently employed due to their effectiveness in video deepfake detection. It was noted that deep learning methods generally outperform alternatives based on classical machine-learning, statistical and blockchain based methods in terms of accuracy and robustness (Rana & Bansal, 2024).

The documented deepfake detection methods still face significant challenges as attackers can evade detection by introducing small perturbations or using novel deepfake generation methods that existing classifiers are not trained to recognize. This highlights the need for detection systems to consider adversarial tactics and improve their robustness (Cao & Gong, 2021). In addition, models trained on specific datasets may not perform well on others, indicating a need for more comprehensive training strategies and diverse datasets to improve generalization (Khan & Dang-Nguyen, 2023).

While deep learning models show promising results, there is room for improvement in precision and the ability to detect more sophisticated deepfakes (Rana & Bansal, 2024). As it appears that nowadays, deepfake is easier to create than detect (Hancock & Bailenson, 2021), we propose a solution based on combining AI technology with social science research to counter the impact of deepfake.

To this extent, this paper aims at presenting a CNN network capable of achieving high performances in detecting deepfake images. We selected a CNN because it has proven effective in capturing the distinct features of images, making it an ideal candidate for distinguishing between authentic and manipulated data. The performance of the network is analysed by connection to other prebuilt architectures, including DenseNet and EfficientNet. The classification capability of the designed CNN was also tested in realistic scenarios, including images and videos extracted from the internet, outside the used dataset. The comparison is based on key-performance metrics such as accuracy, precision, recall, F1 score and computational efficiency, providing insights into the strength and weaknesses of each model in identifying manipulated content. Moreover we have adapted and tested the algorithm for various images/video and content extracted from the internet.

2. Development Environment and Dataset Description

For the development and training of the deepfake image detection model, we chose to use Google Colab. Unlike running locally, where hardware resources can be limited, Google Colab provides access to powerful graphical processing units (GPUs), such as the T4 GPU, which are essential for reducing training and image processing time. The Colab platform is flexible, enabling the execution of the Python developed code from any machine without requiring manual setup of a development environment. In addition,

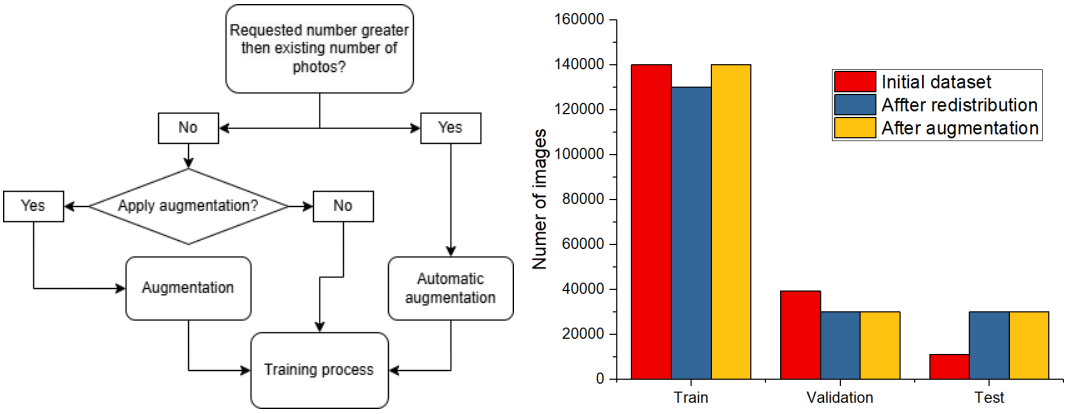
Colab is interoperable with Google Drive which was used for storing the code and model progress.

A pre-procesed openForensics dataset was used to train the model (Le, Nguyen, Yamagishi & Echizen, 2021). The dataset was retrieved from the Kaggle platform (Kaggle, deepfake-and-real-images, 2024) and contains real and manipulated images representing human faces created by various means. By importing an existing dataset directly into Colab, we eliminated the need to manually download, label and organize the images. Our work was focused on preprocessing, training, and model optimization.

The initial dataset was pre-processed to contain 256x256 size jpg images and split into train (140,000 images/75%), validation (39,200 images/20%) and test datasets (10,905 images/5%). Figure no. 1 presents a selection of fake (up) and real (bottom) images in the used dataset.

Figure no. 1: *Real and fake images in the dataset*

The initial classes in the original dataset were merged, then the images were split again into train (70%), validation (15%) and test (15%) for a more balanced distribution. The test and validation datasets were populated first with images while the remaining number of images was used for training. The training process was set to run on a given number of training images. A data augmentation process was configured to reach the required number of training images in the training dataset. In the augmentation process the following image modifications were implemented: horizontal flip, rotation of the image up to 20°, width/height shift of up to 10% of the image dimension, shear up to 10%, zoom (up to 20%), brightness adjustments (between 80% and 120%). The nearest fill mode was used to generate the empty pixels values resulted after the applied modifications. The diagram used to apply data augmentation is presented in Figure no. 2 (a) while Figure no. 2 (b) presents a comparative histogram of the initial and augmented dataset.



(a) (b)

Figure no. 2: *Diagram for data augmentation (a) and histogram of the dataset (b)*

A larger number of images was allocated to the test dataset, while the number of images in the training dataset was increased using the data augmentation techniques described above.

3. Network Architecture and Training

We have designed a customized CNN model, consisting of 4 convolutional blocks, followed by dense layers for classification. Each Conv2D layer applies 16, 32, 64, and 128 filters, respectively, with a 3x3 kernel and ReLU activation function, so that the model can extract important features. After each convolutional layer, we introduced a Batch Normalization layer. For size reduction and extraction of the most relevant features, each convolutional block is followed by a MaxPooling2D layer sized 2x2. To prevent overfitting, we added Dropout layers with progressive rates (0.1, 0.2, 0.3 and 0.4) so that the model learns generalizable features without over-specializing on the training set. After feature extraction through the convolutional layers, the network is flattened using a Flatten layer having an 3D input tensor of 14x14x128 which is reshaped to a 1D vector sized 25088. The resulting 1D vector is further passed to the dense layers. We used a dense layer of 256 neurons with ReLU activation, followed by Batch Normalization and Dropout (0.5) for generalization. Finally we added an output layer (type Dense (1)) with sigmoid activation, which returns a value between 0 and 1, representing the probability that an image is real or fake. The CNN network architecture and layer dimensions are depicted in Figure no. 3.

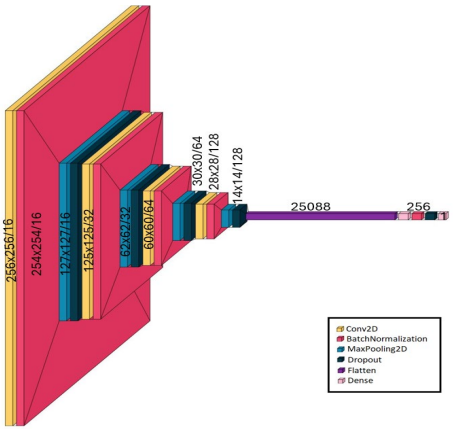


Figure no. 3: *CNN network architecture*

The designed model was compiled using the Adam optimizer, with a learning rate of 0.0005. The binary_crossentropy loss function was used. This is considered optimal for binary classification problems, such as detecting real or fake images. For evaluation, we used accuracy to measure model performance during training and on the validation set.

As a method to prevent overfitting, we implemented an Early Stopping callback, which monitors the loss on the validation set (val_loss). If it does not improve for 3 consecutive epochs, the training stops automatically, and the best epoch weights are restored. The model was set to have a maximum number of 25 epochs, using the model.fit() function, where we used the train dataset and validation data as input. For each epoch, the model displays the losses and accuracy on the training and validation sets, and the setting of the verbose parameter to a value of 1 enables visualization of the progress in the execution console. The evolution of the loss value and accuracy during training is depicted in Figure no. 4.

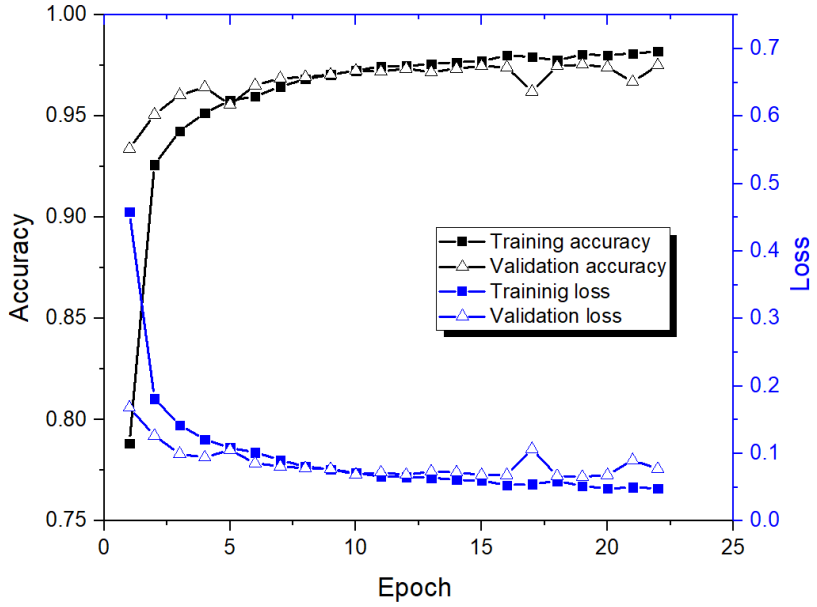
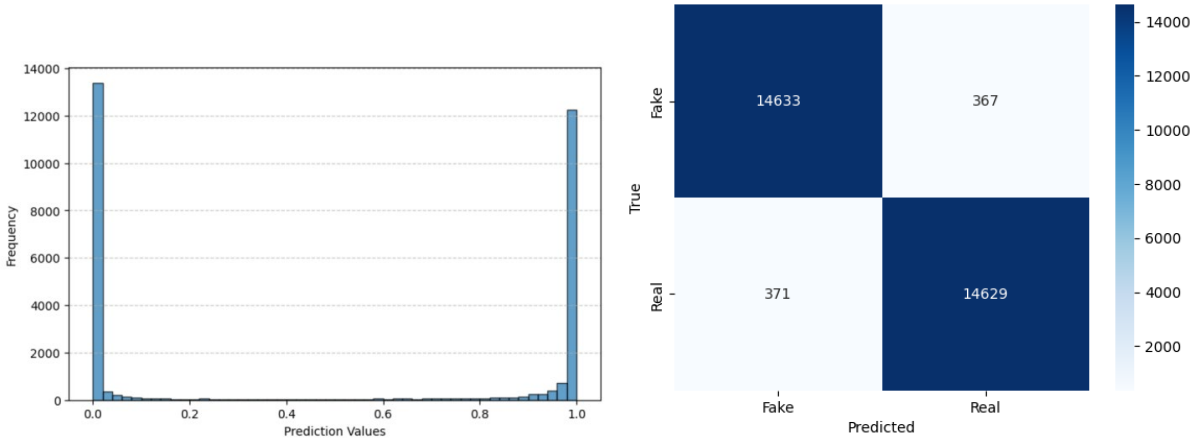


Figure no. 4: *Evolution of the model loss and accuracy during training*

As visible from Figure no. 4 the model accuracy increased during the training process up to a value of 98.2 % for the train accuracy and 97.15 % for the validation accuracy. The training process stopped at epoch no. 22 to prevent overfitting. The total training time was 1h 44m 3s for GPU training, with an average time of 4m 44s per epoch. As a comparison, a similar training performed on Central Processing Unit (CPU) took 5h/epoch. To compare, training on GPU is about 63 times faster per epoch than on CPU, highlighting the major speed advantage of using GPU for deep learning tasks, particularly for models involving CNNs or larger datasets.

4. CNN Performance Evaluation

The model performance on the test dataset was evaluated using the model.predict function. This generated probabilities for each image in the test dataset, indicating how certain the model is that an image belongs to the fake or real class. The probabilities were converted into discrete predictions, by applying a threshold value of 0.5, so that values above this threshold were categorized as Fake (1) and those below the threshold were labeled Real (0). A histogram representing the distribution of the test dataset prediction values is presented in Figure no. 5 (a), while the dataset confusing matrix is visible in Figure no. 5 (b).



(a) (b)

Figure no. 5: *Test dataset prediction values (a) and confusion matrix (b)*

As visible from Figure no. 5, most of the prediction values build up towards the extreme values of 0 and 1, while a smaller number of images are classified with weaker probabilities (around 0.5).

To compute the accuracy on the test dataset, we compared each prediction label with the true label and the test accuracy was determined as the ratio of the total number of correct classifications to the total number of images tested. The calculated test accuracy was 97.57%.

The performance of the designed CNN was evaluated in comparison to existing methodologies for deepfake image detection using the same dataset. Specifically, the classification accuracy of the designed CNN was assessed against DenseNet/Xception model (Kaggle, Densenet, 2025), EfficientNet (Kaggle, Efficientnet, 2025) and a comparable 5 layer CNN (Kaggle, CNN_5layers, 2025) to determine its effectiveness in identifying deepfake images. The models were assessed based on accuracy, loss, precision, recall, F1-score, and training runtime and the results are presented in Table no. 1.

Table no. 1

Comparison of models for deepfake image classification

Model	Parameter								
	Accuracy (%)			Loss		Precision	Recall	F1-score	Runtime
	Train	Valid.	Test	Train	Valid.				
DenseNet	87.93	89.74	83.1	0.28	0.24	0.84	0.81	0.82	7h19m GPU T4 x2
Xception model (transfer learning)	69.46	77.09	73	0.75	0.48	0.74	0.73	0.73	NA
EfficientNet	76.24	79.7	70.9	0.48	0.43	NA	NA	NA	1h11m GPU T4 x2
CNN_5 layers (10 epochs)	95.8	94.03	91.9	0.1058	0.1558	0.92	0.91	0.91	50m GPU P100
CNN_4 layers (22 epochs)	98.2	97.15	97.5	0.04	0.07	0.97	0.97	0.97	1h44m GPU T4

Among the evaluated models, the designed CNN achieved the highest test (97.5%) and validation accuracy (97.15%), suggesting superior generalization performance. The five-layer CNN also demonstrated strong performance (91.9% test accuracy) with a relatively low training loss (0.1058), indicating efficient learning.

Conversely, DenseNet, while achieving high validation accuracy (89.74%), exhibited a drop in test accuracy (83.1%), suggesting potential overfitting. Xception (transfer learning) and EfficientNet achieved moderate performance, with test accuracies of 73% and 70.9%, respectively, but required significantly less training time compared to DenseNet.

The computational efficiency varied significantly among models, with DenseNet requiring the longest runtime (7h19m on GPU T4 ×2), while the CNN (5 layers) completed training in only 50 minutes on GPU P100.

The model classification capability was evaluated using a set of images that were not extracted from the dataset. The selected images were resized to a dimension of 256x256, converted to a numerical matrix and normalized to match the model. We have adapted the algorithm for video classification. We have segmented the video into its frames and analyzed each 10^{th} frame. The model makes the prediction for each frame, and the results are stored in a list of predictions. As the video is analyzed, each processed frame is displayed along with the predicted label and the model's confidence level. After all relevant frames have been analyzed, the predictions are averaged to determine a final verdict on the entire video. If the average of the probabilities exceeds 0.5, the video is classified as fake, otherwise it is considered as real. Figure no. 6 presents the result of the classification for a selection of images/frames.

Figure no. 6: *Classification of images/frames using the designed CNN*

Future research is aimed at testing the model in various realistic cases, including variations in lightning scenarios, image resolution and compression artifacts, ensuring its applicability to real-world deepfake detection tasks.

5. Conclusion

This paper presents the development, training, and performance evaluation of a customized 4-layer CNN for the classification of deepfake (manipulated) and real images. The model was implemented and trained on a publicly available dataset using Google Colab, with data augmentation applied to expand the size and diversity of the training data.

The results demonstrated the superior performance of the proposed 4-layer CNN compared to the 5-layer CNN, primarily attributed to the combination of an increased number of training epochs and the effective use of data augmentation techniques. The application of these strategies significantly enhanced the model's ability to generalize, resulting in an impressive accuracy of 97.5% and efficient runtime performance, with an average of 4 minutes and 44 seconds per epoch. The deeper 5-layer CNN, despite its greater complexity, did not yield comparable

performance improvements, highlighting the critical role of training duration and data augmentation in optimizing model performance.

Furthermore, the proposed custom-designed model outperformed predefined and transfer learning models, such as Xception and DenseNet, in terms of both accuracy and computational efficiency. This emphasizes the advantage of tailored architecture, which can be fine-tuned to the specific requirements of deepfake detection tasks, as opposed to relying on general-purpose models that may not be as effective for specialized tasks like image manipulation detection.

The implications of this research are significant in the context of combating online manipulation. With the rise of deepfake technology, there is an increasing need for robust and efficient deepfake detection systems.

The development of specialized, efficient models like the proposed 4-layer CNN is crucial for the real-time detection of manipulated media, which has profound consequences for digital security, online information integrity, and the prevention of misinformation. The findings of this paper reinforce the necessity for continued investment in the development of reliable, scalable deepfake detection tools that can keep pace with the rapid advancements in AI-driven image manipulation techniques.

REFERENCES

Cao, X., & Gong, N. (2021). Understanding the Security of Deepfake Detection. *Digital Forensics and Cyber Crime, ICDF2C 2021, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 441*, 360-378. Springer, Cham. Available at: https://doi.org/10.1007/978-3-031-06365-7_22.

Ghodke, S. (2024). Ethical Implications of Deepfake Technology. *International Journal for Multidisciplinary Research*. Available at: https://doi.org/10.36948/ijfmr.2024.v06i05.28312.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*, 139-144. Available at: https://doi.org/10.1145/3422622.

Hancock, J., & Bailenson, J. (2021). The Social Impact of Deepfakes. *Cyberpsychology, behavior and social networking*, *24(3),* 149-152. Available at: https://doi.org/10.1089/cyber.2021.29208.jth.

Heidari, A., Navimipour, N., Dag, H., & Unal, M. (2023). Deepfake detection using deep learning methods: A systematic and comprehensive review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *14*. Available at: https://doi.org/10.1002/widm.1520.

Khan, S., & Dang-Nguyen, D. (2023). Deepfake Detection: A Comparative Analysis. *ArXiv*, abs/2308.03471. Available at: https://doi.org/10.48550/arXiv.2308.03471.

Kharvi, P. (2024). Understanding the Impact of AI-Generated Deepfakes on Public Opinion, Political Discourse, and Personal Security in Social Media. *IEEE Security & Privacy*, *22*, 115-122. Available at: https://doi.org/10.1109/MSEC.2024.3405963.

Le, T.-N., Nguyen, H.H., Yamagishi, J., & Echizen, I. (2021). OpenForensics: Multi-Face Forgery Detection and Segmentation In-The-Wild Dataset [V.1.0.0] (1.0.0) [Data set]. *International Conference on Computer Vision (ICCV)*, Zenodo. Available at: https://doi.org/10.5281/zenodo.5528418

Pan, D., Sun, L., Wang, R., Zhang, X., & Sinnott, R. (2020). Deepfake Detection through Deep Learning. *2020 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*, 134-143. Available at: https://doi.org/10.1109/BDCAT50828.2020.00001.

Rana, M., Nobi, M., Murali, B., & Sung, A. (2022). Deepfake Detection: A Systematic Literature Review. *IEEE Access*, *10*, 25494-25513. Available at: https://doi.org/10.1109/access.2022.3154404.

Sudhakar, K., & Shanthi, M. (2023). Deepfake: An Endanger to Cyber Security. *2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS)*, 1542-1548. Available at: https://doi.org/10.1109/ICSCSS57650.2023.10169246.

Vaccari, C., & Chadwick, A. (2020). Deepfakes and Disinformation: Exploring the Impact of Synthetic Political Video on Deception, Uncertainty, and Trust in News. *Social Media + Society*, *6*. Available at: https://doi.org/10.1177/2056305120903408.

Rana, P., & Bansal, S. (2024). Exploring Deepfake Detection: Techniques, Datasets and Challenges. *International Journal of Computing and Digital Systems*. Available at: https://doi.org/10.12785/ijcds/160156.

www.kaggle.com/datasets/manjilkarki/deepfake-and-real-images, accessed on November 12, 2024.

www.kaggle.com/code/mdsifatullahsheikh/deepfake-detection-with-densenet, accessed on November 12, 2024.

www.kaggle.com/code/kameshrasu/deep-fake-detection-with-efficientnet, accessed on November 12, 2024.

www.kaggle.com/code/seanstepanek/deepfake-detection-cnn-5-layers, accessed on November 12, 2024.