Detecting Fake and Fraudulent Faces in Images and Videos Using a Convolutional Neural Network and a Django Web Interface

Yeshaswini Bhandari K R[1] Vijay Kumar S[2] Harshitha H B[3]
[1,2,3]Department of Computer Applications
[1,2,3]ATME College of Engineering, Mysore, Karnataka, India

Abstract — Advances in image synthesis now allow highly realistic fake faces to be produced, creating risks of disinformation, identity fraud and loss of trust in online media. This paper presents a prototype system that classifies face images and video as real or fake. A convolutional neural network (CNN) with three convolution–pooling stages and a sigmoid output is trained on a labelled directory of real and fake face images using the Adam optimiser and binary cross-entropy loss. Face detection and a pre-trained face-embedding network with Grad-CAM visualisation support face extraction and interpretation. The trained model is served through a Django web application in which a user uploads a video (MP4, MOV, WMV or AVI) or selects a trending online video and receives a REAL or FAKE verdict. Over eight training epochs the training accuracy rose to roughly 85% while the validation accuracy reached roughly 66%, and the validation loss stopped improving after the fifth epoch, which shows overfitting on the available data. The paper discusses these results honestly, describes the testing approach, and outlines how larger datasets, regularisation and memory-based architectures can improve generalisation to unseen manipulations.

Keywords: Fake Face Detection, Deepfake, Convolutional Neural Network, TensorFlow, Django, Grad-CAM, Image Forensics

I. Introduction

The rapid spread of synthetic faces created with deep generative models, in particular generative adversarial networks (GANs) [1], has led to serious breaches of digital security and trust. Fake faces are used for online scams, fraudulent identity verification, impersonation and the spread of false information. Face-based verification is now common in e-commerce, social networking and digital identity services, so reliable detection of manipulated faces is an urgent practical need.

Detectors based on convolutional neural networks (CNNs) work well on the type of manipulation they were trained on, but often fail on manipulation types they have not seen, which leaves a gap in the line of defence. Large benchmark collections such as Face Forensics [2] have made systematic study possible, yet a deployable tool that an ordinary user can operate from a browser is still uncommon.

This paper describes such a tool. Its contributions are: (i) a compact CNN classifier that labels face images as real or fake; (ii) a processing pipeline that adds face detection, a pre-trained face-embedding network and Grad-CAM interpretation; (iii) a Django web interface for uploading videos and viewing a REAL or FAKE result; and (iv) an honest account of the training behaviour, which shows overfitting and points to the changes needed for better generalisation. The rest of the paper reviews related work, describes the design and implementation, presents results and testing, and concludes with future work.

II. Related work

Early approaches to forgery detection used handcrafted features such as local binary patterns (LBP) and histograms of oriented gradients (HOG) with classifiers such as support vector machines and random forests. Deep learning has largely replaced them. MesoNet [3] is a compact network that targets facial video forgery by analysing mesoscopic image properties. Bayar and Stamm [4] introduce a constrained convolutional layer that learns manipulation traces and supports universal image-manipulation detection. Capsule-forensics [5] uses capsule networks to detect forged images and videos, and ForensicTransfer [6] uses weakly supervised domain adaptation to improve transfer to unseen forgeries.

Face morphing is a related threat to identity documents; Ferrara et al. [7] show that a morphed photograph can pass automatic and human checks at border control. Work on social perception in the human brain, such as that of Allison et al. [8], has also inspired detectors that reason about how a face is perceived rather than only about pixel statistics, for example neural memory networks that predict the future semantic embedding of a face.

Other studies compare feature-based and deep methods and report that deep networks outperform handcrafted features on benchmark data, investigate adversarial perturbations such as the fast gradient sign method against face recognition systems, and combine classical image processing with deep networks in hybrid detectors. A common conclusion is that generalisation to unseen manipulation types remains the main weakness, which motivates larger and more varied training data and architectures with richer representations.

III. Problem statement and objectives

The problem is to decide automatically whether a face in an image or video is genuine or digitally fabricated or altered, and to make this decision available to non-expert users. The objectives are to: (a) build an accurate Python classifier for fake and fraudulent faces using machine learning; (b) assemble real and synthetic face images for training and testing; (c) support rapid, near-real-time analysis; and (d) provide a user-friendly interface that can be integrated into existing security workflows.

IV. System design

A. Overview

The system has three stages (Fig. 1). In the training stage a directory of labelled real and fake face images is loaded, normalised and split into training, validation and test sets, and a CNN is trained and saved. In the inference stage an uploaded video is converted to frames, faces are detected and cropped, the crops are resized and scaled, and the saved network returns a probability p that the face is real. A value

All rights reserved by www.ijsrd.com

below 0.5 is reported as FAKE and any other value as REAL. In the presentation stage a Django view returns the verdict and the video on a result page.

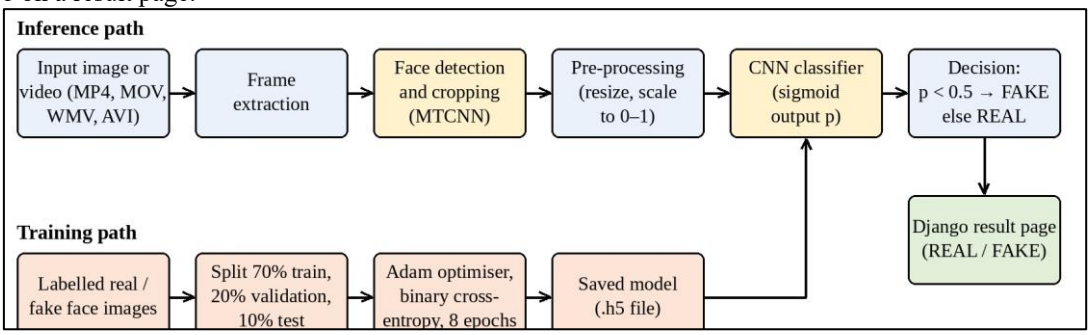


Fig. 1: Training and inference pipeline of the fake face detection system.

B. Data Preparation

Images are read from a directory whose sub-folders name the classes (real and fake) using the Keras image-dataset utility, which batches the data and assigns labels automatically. Fig. 2 shows a sample batch with its labels. Pixel values are divided by 255 so that they lie between 0 and 1. The batches are divided into 70% for training, 20% for validation and 10% for testing. Augmentation by flipping, scaling and rotation can be used to increase diversity, and corrupt or irrelevant images are removed before training.

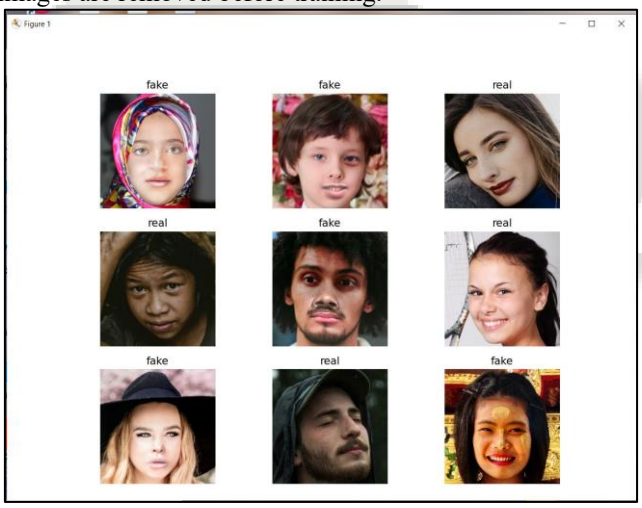


Fig. 2: Sample labelled batch of real and fake faces used for training.

C. Network Architecture

The classifier is a sequential CNN written with TensorFlow [11] and Keras. It has three convolution stages with 16, 32 and 16 filters of size 3 × 3 and ReLU activation, each followed by 2 × 2 max pooling, then a flatten layer, a dense layer of 256 units and a single sigmoid output unit. The layer shapes in Table I imply a 256 × 256 × 3 input. The network has 3,696,625 trainable parameters, most of which are in the first dense layer.

Layer	Output shape	Parameters
Conv2D (16)	254 × 254 × 16	448
MaxPooling2D	127 × 127 × 16	0
Conv2D (32)	125 × 125 × 32	4,640
MaxPooling2D	62 × 62 × 32	0
Conv2D (16)	60 × 60 × 16	4,624
MaxPooling2D	30 × 30 × 16	0
Flatten	14,400	0
Dense (256)	256	3,686,656
Dense (1, sigmoid)	1	257

Table I: Layers of the CNN Classifier

D. Training

The network is compiled with the Adam optimiser and the binary cross-entropy loss, which for N samples with labels y_i and predicted probabilities p_i is

$$L = -(1/N)\ \Sigma\ [\ y_i \log p_i + (1 - y_i) \log (1 - p_i)\ ] \qquad (1.1)$$

Training runs for eight epochs with 44 training batches per epoch, and the loss and accuracy of the training and validation sets are logged with TensorBoard and plotted. The model is then evaluated on the held-out test batches.

E. Face Extraction and Interpretation

A second component uses a multi-task cascaded convolutional network (MTCNN) [9] to locate and crop faces, and an Inception-ResNet network pre-trained on VGGFace2, in the spirit of FaceNet embeddings [10], with a single-output head for classification. Gradient-weighted class activation mapping (Grad-CAM) [12] overlays a heat map on the face to show which regions drove the decision, which helps users judge the result and exposes failure modes.

F. Web Application

The Django project exposes an upload view that accepts MP4, MOV, WMV and AVI files, plus an option to analyse a trending online video. The view loads the saved model, applies the pipeline of Fig. 1, and renders a result page that plays the video and shows REAL or FAKE with a thumbs-up or thumbs-down icon. Django's forms, templates and built-in protections handle uploads and cross-site request forgery safely. Table II lists the environment.

Item	Specification
Hardware	Intel Core i3 or above, 4 GB RAM, 500 GB disk
Operating system	Windows 10 or later
Language	Python (Anaconda, Spyder)
Libraries	TensorFlow, Keras, OpenCV, NumPy, PyTorch, facenet-pytorch
Web framework	Django 3.0.2
Front end	HTML, CSS, Bootstrap

Table II: Implementation Environment

All rights reserved by www.ijsrd.com

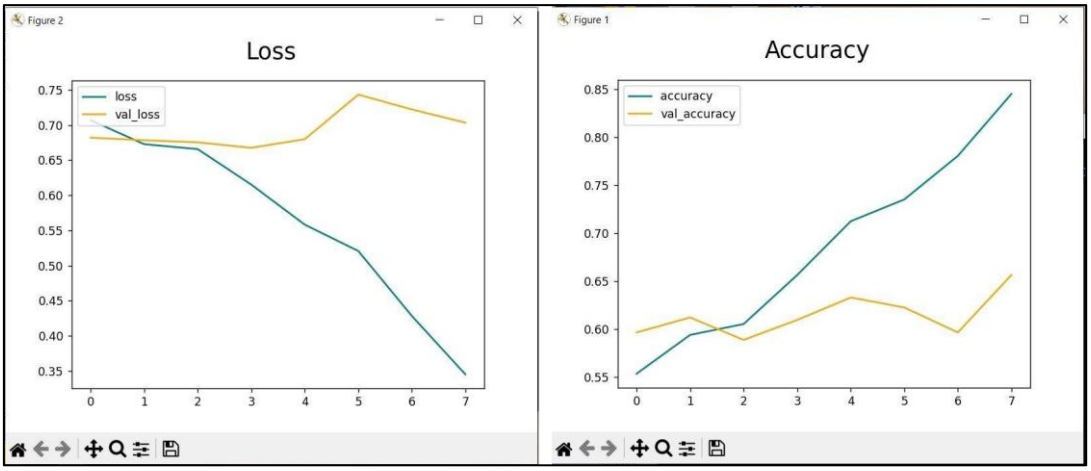


Fig. 3: Training and validation loss (left) and accuracy (right) over eight epochs.

V. Results and Discussion

A. Training Behaviour

Fig. 3 shows the learning curves. The training loss falls steadily from about 0.73 in the first epoch to about 0.35 in the eighth, and the training accuracy rises from about 0.55 to about 0.85. The validation accuracy improves more slowly, from about 0.60 to about 0.66, and the validation loss stays near 0.68, rising to about 0.74 at the sixth epoch and finishing near 0.70. The widening gap between training and validation curves is the typical sign of overfitting: the network begins to memorise the training images instead of learning features that generalise. These values are read from the plotted curves and the training log, and the report does not state a final test accuracy, so none is claimed here.

B. Web Interface

Fig. 4 shows the web interface running on a local Django server. The upload page accepts a video and offers analysis of a trending online video. For a ten-second clip of a natural speaker the result page displays REAL, while for a nine-second clip containing a distorted face it displays FAKE. These two examples demonstrate the complete path from upload to verdict; they are illustrations and not a statistical evaluation.

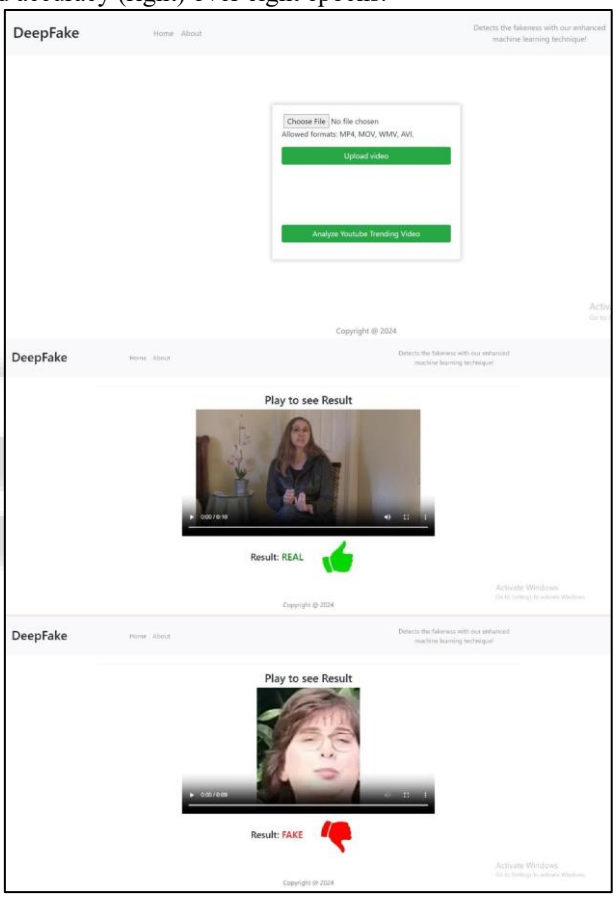


Fig. 4: Upload page and REAL / FAKE result pages.

C. Testing

Testing of the learning component is divided into pre-training and post-training checks. Pre-training tests need no learned parameters and catch implementation errors early: they verify that the training and test sets contain no duplicate images (test-set leakage), that labels are correct, and that the output stays within the range 0 to 1. Post-training tests examine learned behaviour: invariance tests confirm that irrelevant changes, such as small brightness or crop changes, leave the prediction unchanged, and directional tests confirm that stronger manipulation raises the fake score. Integration testing checks the Django view, the model loader and the result page together, including error messages for invalid file

All rights reserved by www.ijsrd.com

types and failed processing. The server was started successfully with no system-check issues on Django 3.0.2.

D. Limitations

The main limitation is generalisation. With a validation accuracy near 66%, the present model is a baseline and not a dependable detector. The dataset is small, the network is evaluated at frame level without temporal information, and it has not been tested against unseen manipulation types or adversarial attacks. Detection of fakes also raises privacy and ethical questions, since face data is sensitive and a wrong verdict can harm a person.

VI. Conclusion and Future Work

This paper presented a prototype that classifies faces as real or fake with a CNN and delivers the result through a Django web application. The system shows that such a tool can be built with open-source components, and its learning curves show clearly where improvement is needed. The work provides a base for further research on robust face forensics.

Future work includes training on larger and more varied datasets such as FaceForensics [2], applying data augmentation, dropout and early stopping to reduce overfitting, using transfer learning from deeper networks, and adding temporal models across video frames. A hierarchical attention memory network that predicts the future semantic embedding of a face is a promising direction for detecting unseen manipulations. Further extensions are detailed reports and analytics in the web interface, e-mail or text notifications for suspect faces, and secure handling of user data in line with privacy regulations.

References

[1] I. Goodfellow et al., "Generative adversarial nets," in Proc. Adv. Neural Inf. Process. Syst. (NIPS), 2014, pp. 2672–2680.
[2] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "FaceForensics: A large-scale video dataset for forgery detection in human faces," arXiv:1803.09179, 2018.
[3] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, "MesoNet: A compact facial video forgery detection network," in Proc. IEEE Int. Workshop Inf. Forensics and Security (WIFS), Dec. 2018, pp. 1–7.
[4] B. Bayar and M. C. Stamm, "A deep learning approach to universal image manipulation detection using a new convolutional layer," in Proc. 4th ACM Workshop Inf. Hiding Multimedia Secur., Jun. 2016, pp. 5–10.
[5] H. H. Nguyen, J. Yamagishi, and I. Echizen, "Capsule-forensics: Using capsule networks to detect forged images and videos," arXiv:1810.11215, 2018.
[6] D. Cozzolino, J. Thies, A. Rössler, C. Riess, M. Nießner, and L. Verdoliva, "ForensicTransfer: Weakly-supervised domain adaptation for forgery detection," arXiv:1812.02510, 2018.
[7] M. Ferrara, A. Franco, and D. Maltoni, "The magic passport," in Proc. IEEE Int. Joint Conf. Biometrics, Sep. 2014, pp. 1–7.
[8] T. Allison, A. Puce, and G. McCarthy, "Social perception from visual cues: Role of the STS region," Trends Cogn. Sci., vol. 4, no. 7, pp. 267–278, Jul. 2000.
[9] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," IEEE Signal Process. Lett., vol. 23, no. 10, pp. 1499–1503, 2016.
[10] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2015, pp. 815–823.
[11] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in Proc. 12th USENIX Symp. Operating Systems Design and Implementation (OSDI), 2016, pp. 265–283.
[12] R. R. Selvaraju et al., "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), 2017, pp. 618–626.

All rights reserved by www.ijsrd.com