Detection of Visual Deepfakes using Deep Convolutional Networks

Viliam Balara, Kristína Machová, Marián Mach

Department of Cybernetics and Artificial Intelligence,Technical University of Košice, Vysokoškolská 4, 08001 Košice, Slovakia
viliam.balara@tuke.sk, kristina.machova@tuke.sk, marian.mach@tuke.sk

Abstract: With social media being a significant part of everyday life, the possibilities of misinformation spreading in textual, visual or audio form are now significantly in comparison to past. With the onset of widely available generative AI models, the need for effective classification methods for generated or altered content grew even larger. This article focuses on the problem of Deepfake detection, particularly in a domain of artificially generated depictions of human faces. For the detection, we have selected a variety of CNN (Convolutional Neural Networks) based architectures which have recently proven their capabilities in image classification tasks. We have selected five models and performed testing on a two-class dataset which contained Deepfakes created with state-of-the-art StyleGAN. The achieved results of selected models were comparable, each model attaining sufficient classification capability.

Keywords: Deepfake; CNN; deepfake detection;GAN; StyleGAN

1 Introduction

Due to the potential danger that they present, deepfakes pose a considerable social, economic and reputational risk for societies and individuals, especially with the rise of available AI methods such as StyleGAN [1]. Deepfakes are photorealistic images, videos, or voice recordings that have been algorithmically generated or manipulated often with malicious intent. Common cases of misuse include the dissemination of harmful content by perpetrators on socials media with the use of fake videos [2], where they can spread virally, leaving lasting damage to the parties involved even if subsequent legal action is taken (cases of individuals or companies protecting their brand name).

Another case of severe misuse of this technology is the creation of illegal sexually explicit content, either breaking the valid laws of particular country, breaching the terms of technology provider or directly attacking particular person [3]. In most cases, it is almost impossible for human users to adequately distinguish between

authentic content and a carefully constructed deepfake with the naked eye. Even the cases of audio deepfakes have proven to be hardly discernible for human ear [4].

To produce deepfake, a variety of methods such as combination, merging, replacement or superimposing of images or videos are used [5], resulting in a realistic believable content.

However, recently the creators generally employ more advanced AI-based techniques, mainly generative adversarial networks (GANs). Often, one type of deepfake modality is accompanied by another to produce more believable results, as in the case of deepfake videos where the lips of the speaking person are modified to be synchronized to the deepfake audio [6].

Due to the increasing presence of deepfakes in social media, our aim is to assess the possibilities of deepfake detection with the focus on generated images depicting humans. The experiments are aimed at using selected methods and subsequent evaluation with regard to recently generated content.

In this paper, we will focus on CNN-based architectures. In the second section of this paper, we will describe related works in this field. In the subsequent sections, we will describe the issue of deepfakes followed by Section 4 where individual CNN architectures are explained. Section 5 is dedicated to our tests and the last section will be our conclusions.

2 Related Works

With the rise of artificially created content on social media, the importance of deepfake detection becomes apparent with an increasing amount of research being dedicated to it. Generally, deepfake detection is handled as a classification problem [7]. These works tend to focus on single modality, prioritizing the importance of visual [8] and facial features. Such features are facial expressions [9], eye blinking [10] or head movements [11]. However, several studies have performed classification where the focus is put on spotting more elaborate deepfake videos which are accompanied by audio [12]. In recent years, the primary solution to this problem is to leverage the advantages of advanced convolutional neural network architectures. Traditional approaches to face the problem of deepfakes often refer to algorithms developed before the advent of advanced deep learning models. These methods generally include manual inspection, forensic analysis [13] that are used to spot anomalies and inconsistencies in media content. In the area of deepfake classification, a variety of classical machine learning pattern recognition techniques have been employed, those techniques include logistic regression, random forest, decision trees, k-nearest neighbors or support vector machines [14]. [15] summarized deepfake

detection methods such as CNN, LSTM-GAN, and DenseNet. A recent survey by [16] has categorized deepfake detection methods as conventional CNN-based detection, CNN with semi-supervised detection, transformer-based detection, and biological signal detection. The survey compares available deepfake detection datasets and methodologies, providing insight into their respective strengths and weaknesses. Other improvements were recorded with the introduction of attention mechanism [17], while promising outcomes are shown in [18] by using an architecture named capsule-network (CN). The main advantage of capsule networks is the smaller number of parameters that are required for training in comparison to very deep networks. The proposed solutions generally employ models with CNN-based architectures, which is an area that is quickly developing. Therefore, we have decided to follow this approach, implement individual methods and assess whether there are suitable for this task.

3 Deepfakes

Deepfake content that appears in the form of images, videos and audio is a content that aims to appear authentic, however, it is made with the use of artificial intelligence algorithms. In most cases, deepfakes are created as facial alterations that can be categorized into four groups based on the type of transformation. These are the entire face synthesis, identity swap, change of facial attributes and change of expression. Attribute editing employs minor adjustments to a person's face. Those include change of skin tone, adding wrinkles or skin marks, facial hair, or making the person older or younger. In the case of expression manipulation, the identity of the person remains the same. However, the facial expression is altered to change the sentiment of the picture or video. A wide array of techniques in computer graphics is available. However, since the traditional approaches involve classical image manipulation, which is time consuming, the creators of such content have recently turned their focus to the GANs to alter the facial expression of people [19]. Examples of commonly used GAN architectures for this kind of task include CycleGAN [20] or InfoGAN [21]. Another example of possible architecture is the DCGAN, which adds upsampling convolutional layers in the generator part of GAN, thus yielding the possibility of generation of high-resolution images. MaskGAN [22] enables the user to transfer selected features for particular expression from one photo to another. MaskGAN learns the face manipulation process as traversing on the mask manifold [23], thus producing more diverse results with respect to facial components, shapes, and poses. A significant advantage of MaskGAN is that it enables the users to directly specify the location, shape and facial component categories for interactive editing.

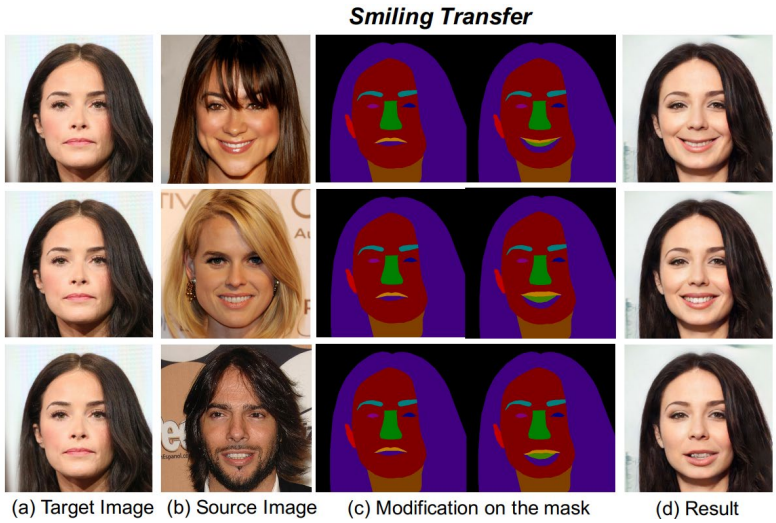

Figure 1
MaskGAN facial Manipulation [22]

There are numerous methods which are used for the creation of facial deepfakes, however, the most currently used methods are autoencoders and GANs. Autoencoder architecture aims to efficiently compress -encode input data down to the essential features and reconstruct or decode the original input from this compressed representation. The Autoencoders consist of encoder and decoder. The role of the encoder is to transform the input data into a reduced-dimensional representation, which is often referred to as "latent space" or "encoding". This representation is used as a basis for the decoder, which uses it to rebuild the initial input. This network can be trained by minimizing the reconstruction error, which serves as the measure for the deviations from the original input and the resulting reconstruction. The bottleneck is a key attribute of the network design, without its presence, the network could tend to simply memorize the input values by passing these values along through the network. A bottleneck constrains the amount of information that can traverse the full network, forcing a learned compression of the input data. Generative Adversarial Networks use a characteristic approach to generating new data by staging two neural networks in a contest against each other. GAN architecture was first described by [24]. DCGAN [25] is a class of GAN that is notable for producing high-quality, high-resolution images, thus it is often used for the creation of deepfakes. The GAN architecture comprises of two constituent sub-models: generator and discriminator. The purpose of the generator is to create new samples which are then fed to the discriminator. The discriminator tries to distinguish whether the provided image is real or created by generator. At the beginning, the generator model receives vector of a fixed length and creates sample in the particular domain. When the training is done, samples in this multidimensional vector space will correspond to samples in the problem domain,

the result being a significantly compressed representation of the data distribution. The aforementioned vector space is referred to as a latent space. The purpose of the Latent space is to compress high-level concepts of the observed input data such as the input data distribution. The discriminator model receives an example from the domain as input, which is either real or generated by generator and attaches real or fake label. The real example is sourced from the training dataset. The generated example is made by the generator model. The discriminator is a standard classification model. When the training process is finished, the discriminator model becomes redundant as it is not necessary for further generative use. Sometimes, the generator can be repurposed in transfer learning due to its ability to extract features. Generator and discriminator are trained together, the role of generator is to create samples that are as close as possible to the real ones, while the role of discriminator is to distinguish whether the provided samples are real or fake. Each round, both models try to improve their respective goals. When discriminator can classify too well, the generator is punished by larger updates to the model parameters. The same applies to the generator, which is penalized when its classifying results are too inaccurate. When the generator succeeds at tricking the discriminator, it receives a reward and the discriminator is penalized, meaning the parameters of the model have to be updated. The end condition occurs when generator creates indistinguishable replicas of the input each time, meaning the discriminator is unable to correctly classify and predict 50% probability for both real and fake classes. With vast amounts of publicly available footage of politicians, actors or private data from social media, it is easy for perpetrators to use this data as a training data for GANs, which may be subsequently used for illicit purposes.

4 Convolutional Neural Networks

Convolutional neural networks are a class of neural networks which have left their mark in computer vision, but thanks to their capabilities they have also been widely used in other fields such as time series forecasting [26] or natural language processing [27]. One of the common applications is the field of healthcare [28]. The main advantage of CNNs is that they automatically extract features and capture spatiotemporal dependencies. When it comes to architecture, convolutional layers, activation functions, pooling and fully connected layers are the main components of CNNs. Starting with the convolutional layer followed by either pooling layer or another convolutional layer, the network ends with the last fully connected layer. The complexity of CNN grows with each additional layer, thus being able to identify larger portions of the processed image. First layers concentrate on less complicated features, for example edges. As the image data passes deeper through the layers of the CNN, it starts to distinguish more delicate features and shapes of the object, resulting in the final identification of the object.

4.1 DenseNet

DenseNet, a convolutional neural network with densely connected structure, was proposed by [29] in 2017, which revolutionized the field of computer vision by proposing a novel connectivity pattern within CNNs, addressing challenges such as feature reuse, vanishing gradients, and parameter efficiency. In contrast to the traditional CNN architectures, where connections are only made between one layer and the following one, in DenseNet all layers within a block are connected between each other, hence the term dense connectivity. This feature makes it possible for each layer to receive feature maps from all previous layers, enabling extensive flow of information through the network. Characteristic features of the DenseNet are:

1. **Mitigated Vanishing Gradient Problem**: Dense connections secure the direct flow of gradients to earlier layers, significantly reducing the vanishing gradient issue commonly occurring in deep networks.
3. **Improved Feature Propagation**: All layers have immediate availability of gradients from loss function and original input signal, which results in improved propagation of features.
4. **Feature Reuse**: DenseNet supports feature reuse through concatenation of features sourced from all preceding layers, thus curbing redundancy while increasing efficiency.
5. **Reduced Parameters**: Despite dense connections, DenseNet is parameter efficient. One of the advantages is the elimination of the need to relearn redundant features, thus resulting in fewer parameters in comparison to traditional networks.

DenseNet architectures are composed of building units called Dense blocks. A dense block is constructed from multiple convolutional layers that are paired with batch normalization as well as non-linear activation functions, such as ReLU. Each layer contained within dense block is given feature maps from all previous layers which serve as inputs. This mechanism promotes feature reuse and propagation through the network. To connect individual dense blocks, transition layers are included. They have two distinctive purposes. The first one is the reduction of feature maps count and second one is the downsampling of the spatial dimensions of these feature maps, which proves to be beneficial in maintaining computational efficiency and lightweight structure of the network. A standard transition layer is made from:

- **Batch Normalization**: normalization of the feature maps.
- **1×1 Convolution**: feature maps quantity reduction
- **Average Pooling**: downsampling of the spatial dimensions.

To define a quantity of produced feature maps by individual layers present in dense block, a hyperparameter growth rate (k) is introduced. A larger growth rate implies that more information is added at each layer, however, it also leads to increased computational costs. The choice of k affects the network's capacity and performance.

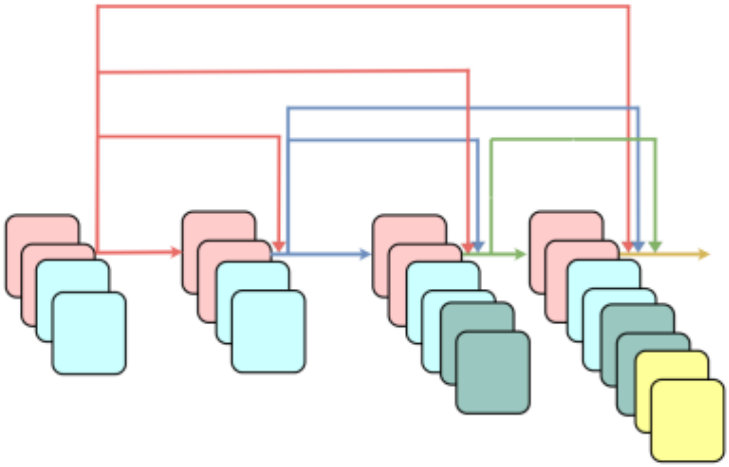

Figure 2
Densenet with growth rate k=2

4.2 VGG-16

Created by Visual Geometry Group in 2014 [30], the characteristic feature of VGG-16 is its depth consisting of 16 layers, 13 of which are convolutional while the three remaining layers are fully connected. The network won the ILSVR (ImageNet) competition in 2014 [31], proving its pioneering capabilities. VGG is one of the pioneers of deep neural net structures, renowned for its simplicity and efficacy in a variety of computer vision tasks. The input of VGG is an RGB image with the size of 224x244. The average RGB value is calculated for all images from the training set image, and afterwards the image is passed as an input to the VGG network. While not having a a large number of hyper-parameters, VGG16 uses convolution layers with a 3x3 filter and a stride 1 that have the same padding and maxpool layer of 2x2 filter of stride 2. Each convolution layer contains 2 to 4 convolution operations. For example, in AlexNet each convolutional layer has only single convolution with the filter size 7x7. The main idea being to use smaller convolutional filter is the reduction of parameters while making the decision more discriminative. This arrangement of convolution and max pool layers reoccurs repeatedly throughout the whole architecture. In the end it has two

fully connected layers, followed by a softmax for output. It is important to point out that despite this, the network has around 138 million parameters (VGG-16) or 144 million parameters (VGG-19). Due to its performance, VGG-16 is used for image recognition and classification of new images, using the pre-trained version. The pre-trained version of the VGG16 network is trained on over one million images from the ImageNet visual database, with the two classify images into 1,000 different categories with 92.7 percent test accuracy. The included classes encompass animals, plants items or vehicles.

4.3 Resnet 50

ResNet is a type of convolutional neural network (CNN), which was devised with the aim of training extremely deep networks. Introduced by [32], the Resnet proved as a breakthrough in the field of deep learning, especially in computer vision tasks, and they have been widely adopted since their introduction. ResNets function by learning residual functions that reference to the layer inputs, instead of learning unreferenced ones. One of the commonly associated problems with deeper neural networks was the vanishing gradient problem. With the increasing network depths, the gradients got progressively smaller, cutting down the ability of network to learn meaningful representations. Based on a deep residual learning framework, ResNet faces the gradient degradation problem and extracts more information from the original data with the introduction of the skip connections, which supports more efficient flow of the information across layers. To form a network, residual blocks are stacked on top of each other. For instance, a ResNet-50 contains fifty layers made from these blocks. The main building unit of ResNets, the residual block, is composed of multiple layers equipped with the skip connections. A variety of residual block modifications exist, the most common ones are:

1. **Identity Block**: The simplest version of a residual block, the input is added directly to the output without any modifications.
2. **Convolutional Block**: Adds the convolutional layers to the residual block to learn more complex mappings.

The characteristic feature of ResNet, the Skip connections, are the distinguishing part that separates it from traditional neural networks. In a standard deep neural network, each layer feeds the output into the subsequent one. In the case of ResNet, certain layers are equipped with ‘shortcuts’ to non-adjacent layers. The roles of the skip connections are as follows:

- **Encourage Training:** Thanks to the ability of network to circumvent one or multiple layers, skip connections enhance gradient from across the network during backpropagation, thus simplifying the training of deeper networks.

- **Alleviate Vanishing Gradients**: By enabling the gradients to omit layers, they reduce the problem of vanishing gradients in deep networks.
- **Identity Mapping:** Skip connections essentially perform identity mapping, where the output of a layer is added to the output of the following layer, thus encouraging the preservation of information.

The numbers in ResNet variants (34, 50, 101, 152) designate the number of layers found in the network. The architecture for each variant varies slightly:

- **ResNet-34:** Uses basic blocks with two 3x3 convolutional layers each.
- **ResNet-50, 101, and 152:** common feature is the use of bottleneck blocks, which have slightly increased complexity. Each bottleneck block consists of three layers: a 1x1 convolution (dimensional reduction), a 3x3 convolution (the main processing unit), and second 1x1 convolution (dimensional restoration). This design increases the network efficiency and depth.

Each version of ResNet is tailored to diverse needs and computational constraints. The smaller variants ResNet-34 and ResNet-50 are popular for tasks requiring balance between complexity and performance, the larger versions ResNet-101 and ResNet-152 are suitable for cases where model depth is necessary in order to process complex patterns.

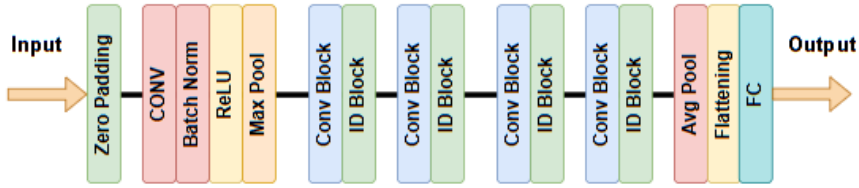

Figure 3
Resnet 50 Architecture

4.4 EfficientNet

Proposed by [33] EfficientNet is a CNN architecture that can is characterized by its use of compound scaling. The main advantage of this concept is its high accuracy accompanied by computational efficiency. It scales uniformly the depth, width and resolution of the neural network. This addresses the problem of increasing accuracy while maintaining reasonable model size. Increasing depth by adding layers, width by adding channels containing convolutional layers considerably increases the number of parameters in the network. Larger number of

parameters leads to more calculations, exacerbating the overall computational burden. Another important aspect of scaling is that it leads to Memory Bottleneck as larger models with more parameters require more memory to store the model weights and activations during processing. To capture large-scale features with more pixels, high-resolution images require deeper networks, however, contrary to these are the finer features that prefer wider networks. To achieve improvements in accuracy and efficiency, all network dimensions need to be balanced during scaling.

1. **Width**: The terms width in this context is applied to the number of channels present in each layer. With increasing width, complex features can be captured which improves the overall accuracy. On the other hand, the reduction of width leads to a more lightweight model.
2. **Depth**: Depth scaling refers to the number of layers found in the network. Deeper models can capture more intricate representations of data; however, they require increased number of computational resources. Contrary to this, shallower models are computationally efficient with the significant possibility of the accuracy reduction.
3. **Resolution**: The resolution scaling modifies the size of the input image. High resolution images contain more detailed information, potentially yielding superior performance. Their main drawback is that they require more computational power and resources. The lower resolution images are more resource-efficient; however, they may lead to a loss of information contained in finer details of a picture.

To balance the three dimensions, EfficientNet uses a principled approach. This is initiated with a baseline model, used as a launching point. Generally, the baseline model is a reasonably sized model that has proven track record on a given task but offers a space for optimization in terms of accuracy or computational efficiency.

A compound coefficient is a user-defined parameter that determines the amount of scaling for the dimensions. It is represented as a single scale value, scaling the depth, width and resolution of model uniformly. By adjusting the value of this coefficient, the overall complexity of the model and associated resource requirements can be adjusted.

The main idea of compound scaling is to scale the dimensions of the baseline model (width, depth, and resolution) in a balanced and coordinated manner. The scaling factors for each dimension are derived from the compound coefficient labeled φ.

- **Width Scaling**: The width scaled proportionally by raising φ to the power of another exponent (denoted as α).
- **Depth Scaling**: the depth of the network is scaled by raising φ to another exponent (denoted as β).

- **Resolution Scaling**: The input image is scaled by multiplying the original resolution (r) by φ raised to a different exponent (often denoted as γ).

Another important component of EfficientNet are Mobile Inverted Bottleneck (MBConv) layers. These are made as a combination of inverted residual blocks and depth-wise separable convolutions. The MBConv layers are a characteristic feature of the EfficientNet architecture. They are inspired by the inverted residual blocks sourced from MobileNetV2; however, they are modified. A standard MBConv layer starts with a depth-wise convolution, which is followed by a point-wise convolution (1x1 convolution) that increases the number of channels, and finally, another 1x1 convolution that reduces the channels back to the original number. The bottleneck design is key to efficient learning of the model. The EfficientNet model architecture also uses the Squeeze-and-Excitation (SE) optimization to further enhance performance. This is incorporated in the SE block, which is designed to aid models in learning by suppressing less relevant features and shifting focus on the essential ones. To reduce dimensions of feature maps to single channel, the SE block employs global average pooling and subsequently two fully connected layers. This allows the model to learn feature dependencies and create attention weights, which are then combined with original feature map by multiplication, thus emphasizing relevant information. EfficientNet is available in a variety of different variants, for illustration EfficientNet-B0, EfficientNet-B1, with different scaling coefficients. For every variant, there is a different ratio of model size and accuracy balance, thus leaving it to the user to select correspondent one to the particular task and its respective requirements.

4.5 ConvNext

The ConvNeXT model was proposed in [34]. It is a pure convolutional model inspired by the design of Vision Transformers, with the aim to improve their performance. One of the features of ConvNets are their built-in inductive biases that make them well suited to a wide range of computer vision applications. In the field of computer vision, the stem of a network typically refers to the initial layer where input images will go through. This layer usually involves downsampling of the image and increasing feature map count (i.e. number of filter channels). In the case of traditional ResNet architectures, the stem consists of a 7x7 convolution with a stride of 2, followed by max pooling. In a convolution, when kernel size is larger than stride (when padding is 0), the result will overlap, meaning that the kernel will convolve over the input. In this way the adjacent information is captured resulting in preservation of relational data.

ConvNext and of vision transformers use a “patchify stem”, which is done by a 4x4 convolution a value of stride being 4. This is followed by layer normalization. Due to the stride and kernel sharing of the same size, the result is non-overlapping

convolution, meaning that kernel convolves over the input, no information sharing occurs between kernel and input, thus unique patches are produced. Ultimately, each of these receptive fields will be merged into the final layers of the network. In a classic residual block, which can be found in ResNet models, the input passes through multiple convolutional layers. There, the number of feature maps is down sampled after the entry and eventually upsampled when exiting the layer. Simultaneously, a skip connection enables the direct addition of original input to the output of the last convolutional layer. This novel modification has resulted in the possibility of achieving increased depth of neural networks while preserving training stability, resulting in the increasing ability to learn complex and abstract representations. Inverted bottleneck block, a component commonly associated with transformer architecture, is a specialized type of residual block. The Inverted Bottleneck starts with a depth-wise separable convolution, which is the combination of two convolutional layers: a depthwise convolution followed by a 1x1 pointwise convolution where the number of feature maps is upsampled by a factor of 4. Subsequently, another 1x1 convolution follows, downsampling the feature map count back to original state that was before entering the block, allowing for a skip connection to be made at the end. ConvNeXt uses an Inverted Bottleneck block design that is similar to the transformers, paired with the layer normalization and GeLU activation. This block design is extremely computationally efficient compared to a vanilla residual block.

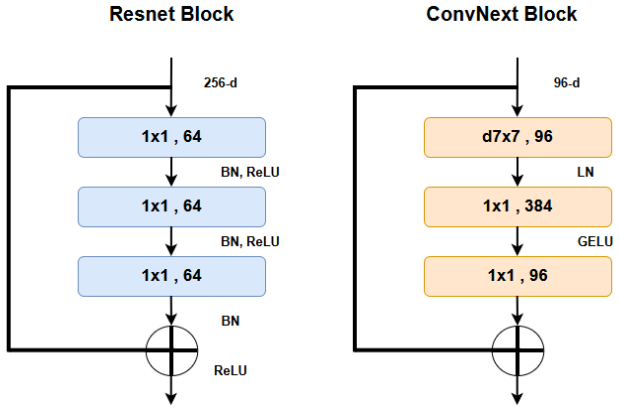

Figure 4
ConvNext Block

The structure of the ConvNext model is complex, involving a wide variety of features, some of which are:

- **Original Architecture** uses small size filters, which renders network lighter and less parameterized while having smaller computational requirements.

- **Long Distance Connections:** ConvNeXt directly connects the outputs of previous layers to the next layers using long distance connections. These connections improve network learning capabilities while increasing accuracy rates.
- **Group Normalization:** Instead of layer normalization, ConvNeXt uses a normalization method called Group Normalization (GN), which enables the network to learn faster and results in training of models with fewer parameters.
- **Activation Function:** ConvNeXt uses an activation function called GELU. GELU provides a linear output for values that are larger than zero. Contrary to the ReLU activation function, it has a non-linear output for negative values, which improves the performance of the network.
- **Global Average Pooling (GAP):** Used to flatten the output of the last convolution layer.

The aim of ConvNeXt authors was to improve the CNN architecture of ResNet with considerable adjustments making it competitive against transformer architectures or architectures that contain attention mechanisms. These adjustments in the case of training follow these principles:

- **Mixup**: a training technique where the network is trained on a convex combination of examples and their labels
- **Cutmix**: a segment is cut from one image and pasted into another image. The label of this new compounded image is adjusted so it reflects the respective image labels in proportion to the image and pasted segments.
- **RandAugmen**t: an automatic image augmentation that can be used out of the box. It works with a reduced search space, showing satisfactory performance on large datasets
- **Random Erasing**: erases random sections of an image.
- **Stochastic Depth**: training technique that shortens the network by randomly dropping layers during training. This allows better information and gradient flow.
- **Label Smoothing**: takes marginalized effect of label dropping into account

4.6 MobileNet

As the name suggests, MobileNet was developed by [35] with the aim of creating a CNN that would be suitable for mobile and embedded devices. The first version of the MobileNet, MobileNet v1, trained 10 times faster in comparison to VGG16.

Not only is it significantly faster, but its size is also considerably smaller while remaining at par with benchmark models in terms of performance. The main features of the MobilNet architecture are depth-wise separable convolution replacing standard convolution, ReLU 6 replacing standard ReLU and width and resolution hyperparameters. There are three main versions of MobileNet available, each of them improving upon the previous one. The MobileNet V2, like the first version, uses depth-wise separable convolutions. These convolutions perform standard convolution in a two-way manner. First, is standard convolution, second is pointwise convolution. This separation leads to a significant reduction in a number of parameters and computations, increasing the networks efficiency. Instead of non-linear bottlenecks found in the first version, the V2 uses linear bottlenecks which secure a more optimal compression rate, which in turn prevents the loss of information while simultaneously increasing the accuracy of the model. The linear bottleneck layer uses the pattern of 1×1 convolution for expansion, depthwise convolution for spatial filtering, and another 1×1 convolution for projection. MobileNet V2 uses the modification of the ReLU activation functions, the ReLU6. The main aspect of ReLU6 is the restriction of the activation values to a range of [0, 6], which results in improved quantization properties for efficient computation on mobile devices. The use of ReLU6 helps in achieving a balance between accuracy and efficiency. The MobileNet V2 architecture is composed of several key building blocks, including the inverted residual block, which is the core component of the network. The architecture of MobileNet V2 consists of:

1. **Initial Convolution Layer**: A standard convolution layer that contains 32 filters and a has stride with the value of 2.
2. **Series of Inverted Residual Blocks**: The network contains several stages, each with a specific number of inverted residual blocks. The expansion factors, output channels, and strides vary across individual stages to manage the computational complexity and receptive field.
3. **Final Convolution Layer**: A 1×1 convolution layer with 1280 filters, followed by a global average pooling layer.
4. **Fully connected Layer**: softmax activation for classification tasks

The third version of MobileNet, V3, introduces the use of hard swish activation and squeeze-and-excitation modules that are found in MBConv blocks. It is two times faster, 30% smaller than the second version, however, the accuracy has decreased by 2%.

5 Experiments

For each individual method, we have used the same approach. For experiments, we have used the Google Colab environment with T4 GPU. The work was conducted in Python programming language, for DL models we have used Keras library. For our purposes, we have used VGG16, ResNet50, DenseNet121, MobileNet, EfficientNet and ConvNext implementations from *Keras.applications* repository. In our experiments, we have modified the parameters of learning-rate and number of epochs. We have also used the early stopping in order to avoid overfitting. The input shape for each network was 256x256x3, the values were normalized.

5.1 Dataset

The used dataset was created as a combination of existing Flickr real face dataset (Flickr-Faces-HQ) provided by Nvidia that were collected for the StyleGAN paper. It also has good coverage of accessories such as eyeglasses, sunglasses, hats, etc. The images are split into two classes, fake and real, each of them containing 70000 samples. The images were crawled from Flickr, thus inheriting all the biases of that website, and automatically aligned and cropped using dlib. Only images under permissive licenses were collected. Various automatic filters were used to prune the set, and finally Amazon Mechanical Turk was used to remove the occasional statues, paintings, or photos of photos. The part of dataset containing fake faces was created with the use of StyleGAN. The dataset pictures are labeled as real or fake, therefore, it is a binary classification task. The whole dataset is publicly available at [https://www.kaggle.com/code/nicoladisabato/fake-face-detection-with-keras-accuracy-0-987]. The dataset is split into train, test and validation parts. The train contains 50000 samples, test 10000 and validation also 10000, meaning that both classes have equal representation in each subdataset. The images are colored, have dimensions of 256 x 256 pixels and are focused on the faces. There are no multiple individuals depicted, only single face per picture.

Figure 5
Dataset example

5.2 Test Results

In this chapter, we have selected the best-performing variants of individual methods, which are summarized in Table 1. The selection criterium was the accuracy achieved on the validation set. Overall, all models achieved validation accuracy above 90%.

The best results were achieved by the EfficientNet architecture with the validation accuracy of 96.76%, followed by ConvNext with 95. 72%. The worst performance was attained by the oldest architecture of VGG16, which scored 92.25%.

Table 1
Results

Model Name	Validation accuracy
VGG16	0.9225
DenseNet121	0.9528
ResNet50	0.9364
MobileNet	0.9448
EfficientNet	0.9676
ConvNext	0.9572

Conclusions

Our conclusion is that with rapidly increasing capabilities of DL models and with accessible computing power in combination with the societal impact that the

social media have it is of a high importance to focus on the development of tools to detect artificially generated content. As the models that are used for generation of content constantly improve, the same needs to apply for the detection tools to prevent harmful conduct. The achieved performance of our selected architecture was comparable, with the best results achieved by EfficientNet. The accuracy provided by those models was satisfactory, each of them scoring above 90%, which would provide the potential user with a viable chance to detect false content including human faces. The experiments were conducted successfully considering the initial goal of this work, we do believe that the rapid development of large language models along with security measures implemented by social media will force the quality of generated content to increase which will in turn create a demand for improved detection models.

It is important to point out that due to data availability we have performed experiments on a single dataset where deepfakes were created with single generative model, thus leaving certain characteristic detection marks. Implementing different generative model or overall settings (such as filters, video frames or photos with multiple people) will result in the inefficiency of the trained models. Therefore, we aim to focus on these shortcomings in our upcoming research by adding more recently generated samples from multiple models.

Acknowledgement

This work was supported by the Scientific Grant Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic, and the Slovak Academy of Sciences under grant no. 1/0685/21 and by the Slovak Research and Development Agency under Contract no. APVV–22–0414.

References

[1] 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)

[2] Ajao, O., Bhowmik, D., & Zargari, S. (2019, May) Sentiment aware fake news detection on online social networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2507-2511) IEEE

[3] Hameleers, M., van der Meer, T. G., & Dobber, T. (2024) Distorting the truth versus blatant lies: The effects of different degrees of deception in domestic and foreign political deepfakes. *Computers in Human Behavior*, *152*, 108096

[4] Dixit, A., Kaur, N., & Kingra, S. (2023) Review of audio deepfake detection techniques: Issues and prospects. *Expert Systems*, *40*(8), e13322

[5] Naitali, A., Ridouani, M., Salahdine, F., & Kaabouch, N. (2023) Deepfake attacks: Generation, detection, datasets, challenges, and research directions. *Computers*, *12*(10), 216

[6] Bohacek, M., & Farid, H. (2024) Lost in Translation: Lip-Sync Deepfake Detection from Audio-Video Mismatch. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (pp. 4315-4323)

[7] Dobeš, M., & Sabolová, N. (2023) Emotion Recognition Using Pretrained Deep Neural Networks. Acta Polytechnica Hungarica, 20(4)

[8] Subramaniam, B., Krishnan, S. V., Radhakrishnan, S., & Balas, V. E. (2024) Fusion of finger vein images, at score level, for personal authentication. Acta Polytechnica Hungarica, 21(6)

[9] Agarwal, S., Farid, H., El-Gaaly, T., & Lim, S. N. (2020, December) Detecting deep-fake videos from appearance and behavior. In *2020 IEEE international workshop on information forensics and security (WIFS)* (pp. 1-6) IEEE

[10] Jung, T., Kim, S., & Kim, K. (2020) Deepvision: Deepfakes detection using human eye blinking pattern. *IEEE Access*, *8*, 83144-83154

[11] Becattini, F., Bisogni, C., Loia, V., Pero, C., & Hao, F. (2023) Head pose estimation patterns as deepfake detectors. *ACM Transactions on Multimedia Computing, Communications and Applications*

[12] Muppalla, S., Jia, S., & Lyu, S. (2023, October) Integrating audio-visual features for multimodal deepfake detection. In *2023 IEEE MIT Undergraduate Research Technology Conference (URTC)* (pp. 1-5) IEEE

[13] Jafar, M. T., Ababneh, M., Al-Zoube, M., & Elhassan, A. (2020, April) Forensics and analysis of deepfake videos. In *2020 11th international conference on information and communication systems (ICICS)* (pp. 053-058) IEEE

[14] Yi, J., Wang, C., Tao, J., Zhang, X., Zhang, C. Y., & Zhao, Y. (2023) Audio deepfake detection: A survey. *arXiv preprint arXiv:2308.14970*

[15] Abbas, F., & Taeihagh, A. (2024) Unmasking deepfakes: A systematic review of deepfake detection and generation techniques using artificial intelligence. *Expert Systems with Applications*, 124260

[16] Gong, L. Y., & Li, X. J. (2024) A contemporary survey on deepfake detection: datasets, algorithms, and challenges. *Electronics*, *13*(3), 585

[17] Dang, H., Liu, F., Stehouwer, J., Liu, X., & Jain, A. K. (2020) On the detection of digital face manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern recognition* (pp. 5781-5790)

[18] Nguyen, H. H., Yamagishi, J., & Echizen, I. (2022) Capsule-forensics networks for deepfake detection. In *Handbook of Digital Face Manipulation and Detection: From DeepFakes to Morphing Attacks* (pp. 275-301) Cham: Springer International Publishing

[19] Hu, X., Aldausari, N., & Mohammadi, G. (2023, October) 2CET-GAN: Pixel-Level GAN Model for Human Facial Expression Transfer. In *Proceedings of the 1st International Workshop on Multimedia Content Generation and Evaluation: New Methods and Practice* (pp. 49-56)

[20] Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017) Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 2223-2232)

[21] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016) Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, *29*

[22] Lee, C. H., Liu, Z., Wu, L., & Luo, P. (2020) Maskgan: Towards diverse and interactive facial image manipulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 5549-5558)

[23] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NIPS*, 2017

[24] Goodfellow, Ian & Pouget-Abadie, Jean & Mirza, Mehdi & Xu, Bing & Warde-Farley, David & Ozair, Sherjil & Courville, Aaron & Bengio, Y. (2014) Generative Adversarial Networks. Advances in Neural Information Processing Systems. 3. 10.1145/3422622

[25] Radford, A., Metz, L., & Chintala, S. (2015) Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*

[26] Zeng, Z., Kaur, R., Siddagangappa, S., Rahimi, S., Balch, T., & Veloso, M. (2023) Financial time series forecasting using cnn and transformer. *arXiv preprint arXiv:2304.04912*

[27] Wang, W., & Gang, J. (2018, July) Application of convolutional neural network in natural language processing. In *2018 international conference on information Systems and computer aided education (ICISCAE)* (pp. 64-70) IEEE

[28] Abdelrahman, L., Al Ghamdi, M., Collado-Mesa, F., & Abdel-Mottaleb, M. (2021) Convolutional neural networks for breast cancer detection in mammography: A survey. *Computers in biology and medicine*, *131*, 104248

[29] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017) Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4700-4708)

[30] Simonyan, K., & Zisserman, A. (2014) Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*

[31] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015) Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*, 211-252

[32] He, K., Zhang, X., Ren, S., & Sun, J. (2016) Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778)

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

[34] Liu, Z., Mao, H., Wu, C. Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022) A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11976-11986)

[35] Howard, A. G. (2017) Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*