Uncovering the Strength of Capsule Networks in Deepfake Detection

Dan-Cristian Stanciu, Bogdan Ionescu
AI Multimedia Lab, Politehnica University of Bucharest, Romania
Email: {dan.stanciu1203, bogdan.ionescu}@upb.ro

***Abstract*—Information is everywhere, and sometimes we have no idea if what we read, watch or listen is accurate, real or authentic. This paper focuses on detecting deep learning generated videos, or deepfakes - a phenomenon which is more and more present in today's society. While there are some very good methods of detecting deepfakes, there are two key elements that should always be considered, i.e. no method is perfect and deepfake generation techniques continue to evolve, sometimes even faster than detection methods. In our proposed architectures, we focus on a family of deep learning methods that is new, has several advantages over traditional convolutional neural networks and has been underutilized in the fight against fake information i.e., the Capsule Networks. We show that (i) state-of-the-art Capsule Network architectures can be improved in the context of deepfake detection, (ii) they can be used to obtain accurate results using a very small number of parameters, and (iii) Capsule Networks are a viable option over deep convolutional models. Experimental validation has been conducted on two publicly available datasets, namely FaceForensics++ and CelebDF, showing very promising results.**

Index terms: **deepfake, capsule networks, deep learning, digital video forensics, face manipulation**

I. Introduction

We live in the era of information. Every day, millions of new articles, videos or social media posts are available to us and only a few clicks away, and because of that, people are more informed and knowledgeable about the past, present and future than ever before. But while technology brings big improvements into our lives, it also comes with some problems, the biggest of which is disinformation. While there are many types of disinformation, in this paper we will outline one in particular i.e. the generation of fake information, mainly in video format - deepfakes. The generation of realistic fake multimedia content became a possibility following the introduction of Generative Adversarial Networks (GAN) [3]. Ideas based on GAN or autoencoder architectures helped create multiple open-source programs that aid the generation of realistic deepfakes, like FaceSwap-GAN [5] or FaceApp[4]. Deepfakes created using these algorithms can be used to impersonate people, for blackmail, to spread fake news or to defame celebrities.

Deepfake detection has been an interesting field of study for a lot of researchers in the last years and as a result, there are many different approaches to deepfake detection. The most popular and successful approaches can be split into 5 main categories, by approach methodology, which are: (i) frequency domain approaches, (ii) approaches based on finding artifacts in the image, (iii) approaches based on finding inconsistencies in physiological signals, (iv) deep convolutional approaches and (v) time domain approaches.

In this paper, we use a new element of Capsule Networks, along with convolutional models, feature extractors and time domain approaches like LSTM. We bring several improvements over the current state of the art architectures and validate our models on two publicly available datasets: FaceForensics++ [1] and CelebDF [2].

The remainder of this article is structured as following: Section II presents an overview of the literature on deepfake detection and Capsule Networks and our contribution beyond the state of the art, Section III describes our Capsule Network models, their architectures, particularities and improvements over existent approaches, Section IV contains our training methodology, models, hyperparameters, datasets, evaluation metrics and results, and a comparison with the state of the art. Conclusions are presented in Section V.

II. Related work

In this section, we will present the following topics related to our contribution: (i) available deepfake detection datasets, (ii) best state-of-the-art approaches for deepfake detection and (iii) the evolution of Capsule Networks, their advantages, disadvantages and their use in the deepfake detection task.

Deepfake detection datasets. Today, deepfakes are more unrecognisable than ever before, and most humans have a hard time differentiating between real and fake videos. Because of that, the interest in detecting them has grown significantly. There is an increasing need of data depicting deepfakes, to help deep learning algorithms learn to discern a fake video from a real one. At the moment, there are multiple open-source datasets available with that exact purpose. The majority of datasets share the following characteristics: videos of short length (10-20 seconds), depicting adult humans filmed at a close distance, the majority of which are talking and remaining fairly stationary. The main type of manipulation used is the identity swap: using the face of another person to replicate the original person's characteristics and facial expressions, by blending a generated face onto the original video. There are 3 generations of deepfake detection datasets, categorised by the realism of the deepfakes, number of videos, quality of videos, number of different people in the videos or the consent of the actors present in the videos. Two of the most popular public datasets for deepfake detection are FaceForensics++ [1]

and CelebDF [2]. FaceForensics++ contains 1000 real videos from YouTube, 1000 deep learning generated identity swap videos [17] and 1000 videos generated with a public FaceSwap software [16]. CelebDF is a 2nd generation dataset, containing 890 real videos and 5639 deepfake videos obtained from identity-swapping between 59 celebrities, with 10 different video scenarios for each celebrity.

Deepfake Detection Approaches. The detection of deepfakes remains an open problem. In spite of the fact that deepfake detection systems continue to evolve and obtain better results, the same thing is happening to deepfake generation techniques. Furthermore, while detection methods are extremely diverse, no method is perfect and many of them suffer from generalization issues.

Convolutional approaches for deepfake detection may be the most reliable, because they are not based on a limited set of features. State-of-the-art architectures like XceptionNet [25] showed success in multiple approaches [1] [6] [29]. One of the most successful models using XceptionNet is presented in the paper "Detecting Deepfakes with Metric Learning" [41], which combined an Xception feature extractor with a triplet approach to obtain a 99.2% AUC on the CelebDF dataset.

Due to the fact that the deepfake detection databases contain video content, leveraging the temporal dimension using recurrent neural networks or LSTM [23] proved to be successful in multiple papers [31], [21], [40] [34]. An approach by De Lima *et al* [40] includes several ways of using and combining 3D and 2D convolutional layers, like R3D [42] or I3D [43] to leverage the temporal dimension. A few notable results are a 97.59% AUC for the I3D architecture and 99.73% AUC for the R3D architecture. I3D is based on InceptionV1 [44] and uses Inception blocks, but uses 3D convolution instead of 2D convolution, applied on groups of frames from the video. The R3D architecture also uses groups of 16 consecutive frames as an input and follows the original 3D Residual Networks [42].

Capsule networks in deepfake detection. Capsule Networks are an emerging concept in the field of Machine Learning. Although they were first introduced in 2011 by G. Hinton [45], they did not become a popular method until 2017, when S. Sabour, N. Frosst and G. Hinton introduced the routing by agreement algorithm [46] which would help train the capsules and propagate information through layers of capsules. The idea of a Capsule is based on the fact that every value in a convolutional neural network is a scalar instead of a vector. Therefore, Capsules were introduced as vectorial units which contain information on multiple dimensions. This can help with many shortcomings of traditional Convolutional Neural Networks (CNN).

Although, at the moment, Capsule Networks do not outperform traditional Convolutional Networks in computer vision tasks, they solve some limitations for CNNs that often times could lead to poor performance. One of the aspects that George Hinton criticises regarding Convolutional Neural Networks is the use of Pooling layers. While pooling is one of the key contributors to the success of convolutional neural networks, it has one crucial drawback: information is lost in the pooling process. In some cases, information like the position of an object could be lost while information about the presence of an object goes deeper into the network. While most of the time we only need to determine whether an object is in the image, there are cases where the position of that object is as important as its existence. In the case of detecting human faces, a neural network looks for key elements like eyes, nose or mouth, but may not look for their position because a max pooling layer deletes that information. So, according to the "Picasso problem", if the eyes of a person are located near the mouth, a traditional convolutional network would still detect a person in that image. On the other hand, using a vector to define the features in the eyes could account for elements like position, color, skewness, texture, deformation etc. Another drawback of using the pooling operation is presented in a paper regarding the explainability of Capsule Networks, by Shahroudnejad *et al* [47]. Because the pooling operation discards features, the CNN needs to compensate by requiring more training data. More than that, CNNs require a longer duration of training because they need to be big in depth to perform, whereas Capsule Networks need to be more wide than deep [47].

Many Capsule Network architectures are composed of the same basic elements: a feature extractor CNN, a set of primary capsules and a set of secondary capsules. The data from the CNN is shaped in the form of capsules: vectors of fixed dimension. Those are the primary capsules. The secondary capsules are usually bigger and can represent something tangible (eg: a capsule for every class). The primary capsules are linked to the secondary capsules using the dynamic routing algorithm presented in [46]. The values for every primary capsule are multiplied by a weight matrix so the result has similar length to the secondary capsules, called a prediction vector. Basically, the primary capsules try to guess what values will be in the secondary capsules. The values for every secondary capsule are calculated by using the dynamic routing algorithm, which outputs a weighted sum of the predictions from the primary capsules. To calculate the weights, the algorithm goes through a number of iterations in which capsules with predictions closer to the weighted mean will be assigned a greater weight (starting from weights equally distributed among capsules). Nonlinearity is introduced using the "squash" function, which squashes the vector to a maximum length of 1 while retaining its direction. The weights are bigger for prediction vectors that agree with each other and smaller for those who do not. There are multiple ways to output predictions using secondary capsules. A method presented in [46] is to calculate the length of the secondary capsules and assign that to a probability. Other methods include using the first value in the capsules as a probability for that class or using fully connected layers which result in output probabilities.

Looking at their advantages, we can see that Capsule Networks are a good fit for detecting deepfakes. Their ability to generalize, the small number of parameters used, their invariance to transforms applied to the image like affine transforms, rotation, dimension or perspective changes and a possible resistance to attacks [49] are all desirable qualities

in a deepfake detection model. One disadvantage of capsule networks is that capsules are limited in how much they can learn, so they sometimes perform worse than traditional CNNs on large and diverse datasets like ImageNet [30].

Two papers by Nguyen *et al* [50] [51] highlight the possibility of using Capsule Networks for the task of deepfake detection. The papers present an architecture containing the following elements: a feature extractor (CNN), a number of branches containing convolutional layers which lead to the primary capsules, a stats pooling [52] layer which calculates the mean and variance of each filter, the primary capsules and 2 output capsules for the 2 classes: real and fake. In [50], different Capsule Network architectures were used, the best of which scored a 93.11 binary classification accuracy on the FaceForensics++ dataset using only 3.9 million parameters. Tolosana *et al* [24] used the same Capsule Network architecture, which yielded a 99.52% AUC on FaceForensics++ and a 82.46% AUC performance on the much more difficult CelebDF database. Additionally, the first paper by Nguyen *et al*, Capsule Networks were tested on the Replay-Attack database [53] and yielded perfect results.

III. Proposed Method

In this section, we present our approaches using multiple Capsule Network architectures that we implemented, their components, their improvements over already existent models, and their advantages. In this paper, we advance the state of the art by using our Capsule Network architectures. We have achieved the following feats:

- We uncovered the unexploited power of capsule networks in the deepfake detection task.
- We experiment on 2 real world datasets, achieving performances over the state of the art.
- We are able to maintain state-of-the-art performances while significantly reducing the number the parameters.

Capsule Network models. Our main contribution in this paper is the implementation of several Capsule Network models that are used to detect deepfakes. Capsule Networks bring several big improvements over CNN architectures that can be leveraged for the deepfake detection task.

While using Capsule Networks in the deepfake detection task is not a new idea [50] [51] and it has already been implemented by Nguyen *et al*, we believe that their implementation has a few shortcomings that prevent it from reaching high levels of performance. In our search for the best architectures, we start from this state-of-the-art implementation and adjust or correct the models to increase their performance. The Capsule-Forensics architecture presented in [50] [51] starts with a part of a pretrained VGG-19 model, used to extract features which will be later transformed into primary capsules. Before creating the primary capsules, a stats pooling layer [52] is used. It calculates the mean and standard deviation for the filters. After that, 3 to 10 primary capsules are formed, with 8 features each. The next step is the routing by agreement algorithm, which performs a weighted sum of the prediction vectors obtained from the primary capsules. Two iterations are used in the dynamic routing by agreement algorithm. Nguyen *et al* use 2 output capsules, one for each class (deepfake and real), which are 4x1 vectors. Finally, The resulted values go through a softmax layer and an average function to predict the deepfake probability. Using 3.9 million parameters, the authors of the papers [50] [51] managed to obtain a 93.11% Accuracy on binary classification for the FaceForensics++ database. Using the same architecture, Tolosana *et al* [24] scored 99.52% AUC for FaceForensics++ and 82.46% AUC for CelebDF.

Our CapsuleNet architectures aim to solve the problems in the Capsule-Forensics architecture [50] [51] and leverage Capsule Networks in multiple sizes and complexities, architectures and configurations. We kept the default elements of a Capsule Network architecture, while correcting some of the shortcomings in the Capsule-Forensics architecture presented above. Our architecture is presented in Fig. 3. All of our Capsule Network components, changes or improvements are listed below:

- We used part of a pretrained VGG19 convolutional network to extract features for the primary capsules. For the bigger models, we used the first 8 convolutional layers, resulting in about 2.3 million parameters total. For other models, we used as few as the first 3 convolutional layers, resulting in under 500,000 trainable parameters. We used finetuning in all of our models.
- While the stats pooling layer used by Nguyen *et al* could be useful and has the big advantage of making the network invariant to input size, it goes against one of the main principles of Capsule Networks: the use of pooling layers eliminates lots of useful information and, although statistics with stats pooling layers might help, we believe that they may have a negative effect on the models, hindering the performance of the capsules overall. Therefore, we decided to not use the stats pooling layer. A next logical step is training a CNN extractor without the use of MaxPooling layers. However, that idea will be used in future improvements.
- The authors of [50] decided to use between 3 and 10 primary capsules. The effect of this choice is one of the following: (1) every primary capsule will have a big importance in the calculation of the values for the secondary capsules, or (2) a small number of capsules will have a role in the calculation of secondary capsules values. Therefore, for this architecture to work, the capsules will need to be error prone. On the other hand, by using a big number of capsules, the routing algorithm will use only use capsules which "agree with each other", eliminating noise and poor predictions. By having a wider capsule network, it matters less that every capsule predicts a correct value, because the algorithm concentrates more on the agreement between primary capsules. For that reason, our architectures use a big number of 8D primary capsules. We also try to use very big primary capsules to determine if they provide an advantage.

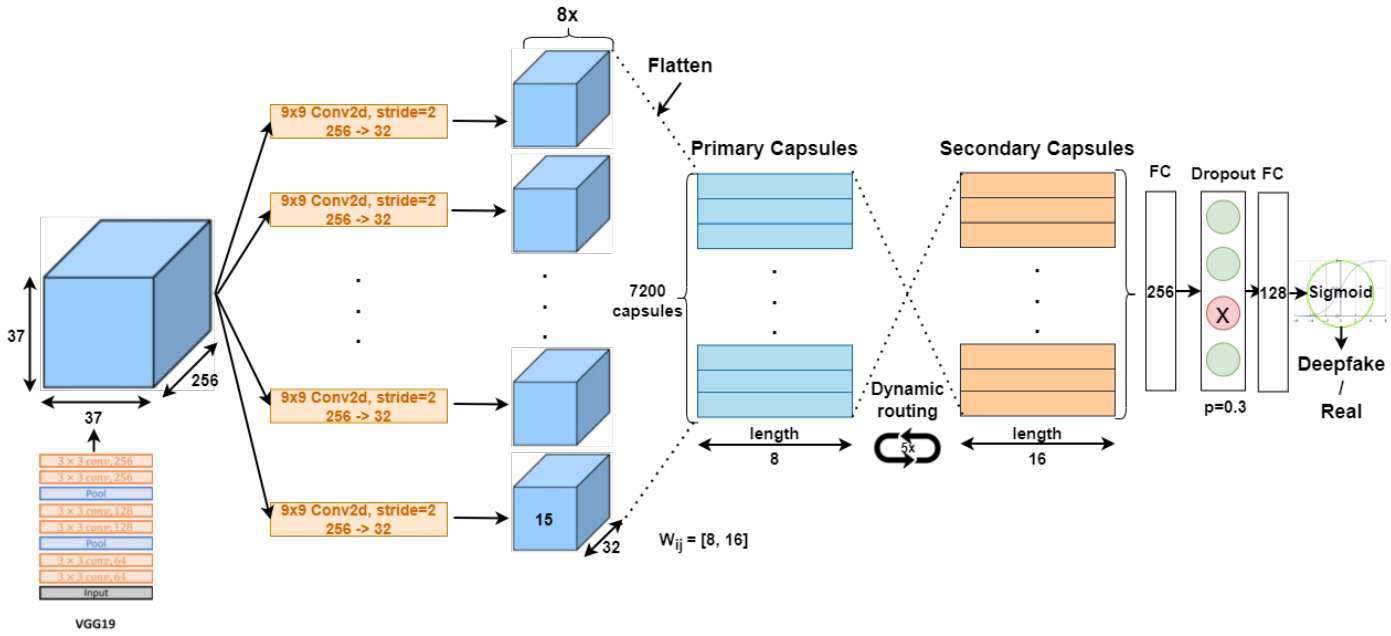

Fig. 1. CapsuleNet architecture for our best model. It is composed of the following building blocks: (1) Part of a VGG19 extractor, (2) Primary capsule layer: 9x9 Conv2d with stride=2, 32 filters, (3) Primary capsules, obtained by flattening the output of the previous block (4) The dynamic routing algorithm, (5) The resulted secondary capsules, (6) Prediction using the features from the secondary capsules in fully connected layers, with dropout enabled

- While we initially tried using one secondary capsule for each class, we also tried using more secondary capsules than available classes. In most of the cases, we used 16 secondary capsules of 16x1 dimension, followed by fully connected decision layers which would determine if the image is deepfake or not. We also used dropout to ensure that no single feature from the secondary capsules would become too important. This approach resulted in better performance on the Celeb-DF dataset.
- Using a big number of loops in the dynamic routing algorithms ensures that the secondary capsules will be obtained using a better set of weights, because the regression to the mean will be more accurate. The authors of [50] and [51] only used 2 loops to determine the weights, while we use a minimum of 3 and a maximum of 5.
- The architecture presented in Fig. 3 can detect deepfakes at the frame level. But, deepfakes are often videos, so we need to consider the evolution in time in these videos. We do that by using LSTM together with features extracted from the Capsule Networks. The CapsNet-LSTM models use the same architecture presented in Fig. 3, but instead of having fully connected layers after the secondary capsules, they use the features in the capsules as descriptors for the images. The CapsNet-LSTM models expect 256 consecutive frames from the video as an input. They use the secondary capsule values from those frames as an input for a 2-layer LSTM, resulting in a temporal descriptor for the whole video. An output layer is used to make the decision between deepfake and real.

Although we propose several different Capsule Network architectures, the majority of them follow a similar pattern, shown in Fig. 3: (1) a pretrained CNN feature extractor, which will be fine-tuned during training, (2) a primary capsule layer, which uses parallel convolutional layers on the output of the CNN separately. Each cube from Fig. 3 contains 32x15x15=7200 features, so each of the 8 primary capsule layers is used to determine one of the 8 dimensions of the 7200 primary capsules, (3) the primary capsules, obtained from flattening the results of the parallel convolutional layers and stacking them, (4) the dynamic routing algorithm, which contains a high number of loops to ensure that the weights for the secondary capsules are as accurate as possible and distant predictions will have small weight values, (5) the set of secondary capsules, which contains more capsules than the number of classes, (6) an output part, which uses fully connected layers with the features from the secondary capsules to obtain a prediction (using a dropout layer to prevent overfitting).

IV. Experimental Results

This chapter contains information regarding our experimental setups and results. We will present the datasets used for the experiments, their preprocessing and the evaluation metrics, our convolutional models used as baselines, our Capsule Network models and their hyperparameters and the experimental results obtained using those models, compared to the state of the art.

Datasets and evaluation. We used 2 datasets for training and evaluation: CelebDF [2] and FaceForensics++ [1]. FaceForensics++ is a generation I dataset and contains 1000 real videos, 1000 deepfakes and 1000 identity change videos which

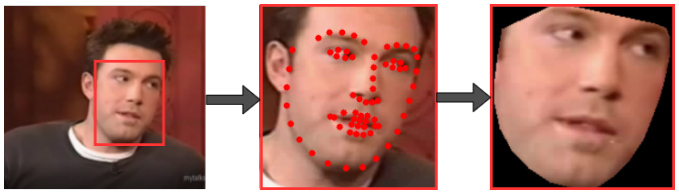

Fig. 2. Preprocessing algorithm, used in our previous work [34]. Preprocessing steps: (1) Face detection, (2) extraction of facial landmarks with OpenFace2 [27], (3) extraction and alignment of the facial region, (4) background elimination, (5) image resizing

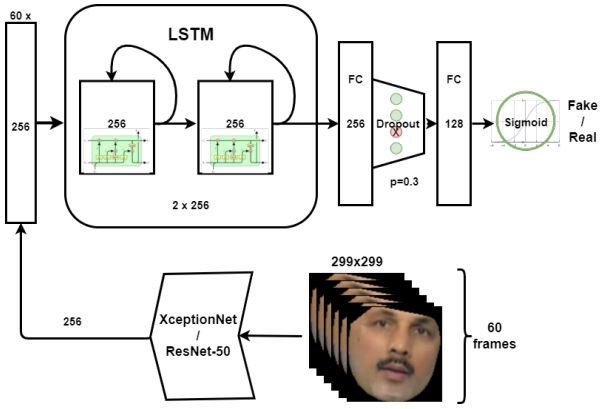

Fig. 3. Baseline CNN-LSTM models. (1) 256 features are extracted for each of the 60 frames, (2) the features are input to a 2-layer LSTM, which outputs a temporal descriptor, (3) fully connected layers are used to predict whether the image is real or deepfake

are not created using deep learning. We decided to use 1000 real videos ans 1000 deepfakes for training and evaluation. We split the dataset into 80% training data and 20% test data. Furthermore, we decided to train and test the algorithms on the uncompressed part of the dataset.

The datasets contain videos of both real humans and deepfakes, their length being, on average, over 10 seconds. The preprocessing algorithm is similar to the one used on our previous work [34] regarding the detection of deepfakes. We know that deepfakes in both datasets are created by using person identity change, so the only region of interest for our algorithms is the face. Therefore, we extract the facial region only from each video frame. Our preprocessing pipeline contains the following steps: (1) face detection, (2) detection of certain facial landmarks using the open-source software OpenFace2 [27], (3) extraction of the facial region using the outer facial landmarks (face contour and eyebrow landmarks), (4) elimination of background, to ensure that no added noise is present in the image, (5) resizing the image to 299x299. This process is applied to the first 256 frames of every video. A diagram depicting this process is shown in Fig. 1.

CelebDF is a harder generation II dataset which contains 5,639 deepfakes of celebrities and 590 real videos of celebrities and 300 YouTube videos. For evaluation purposes, we use the train-test split recommended by the authors [2]: 514 videos are used for evaluation, while the rest are used for training. The evaluation set contains 178 real videos and 336 deepfakes, which is important because the set is fairly balanced, with more than 1/3 of the evaluation data being real videos (compared to the training data, which has a ratio of over 7:1 in favour of the deepfakes). For training, we used 16 random frames for each class, every epoch. We also only use 50% of the training data for each epoch, picked randomly. This, combined with the fact that we pick random video frames each epoch, ensures that it is harder than usual to overfit the model, with the disadvantage that it may take more epochs to obtain the best results. More than that, for CelebDF, we use 7 parts deepfakes and 3 parts real videos for training to ensure more data balance when training. The training data is also slightly augmented by adding noise, changing the brightness or randomly flipping the images. The data is also normalized to fit the ImageNet distributions.

Because the aim of this paper is binary classification, the AUC-ROC score is the best evaluation metric. By using AUC (Area Under Curve), we do not need to use a threshold for decision, because the metric evaluates the area under the True Positive vs False Positive curve at different thresholds. Basically, AUC measures how well a model can separate positive examples from negative examples. Another advantage of AUC is the fact that it is less sensitive when it comes to unbalanced datasets, like CelebDF, compared to Accuracy. Due to the fact that the training data is in video form, the evaluation is done in 2 ways:

- For models which do not contain a temporal neural network, we use the mean deepfake probability for every frame in the video.
- For models with LSTM, we use the features from 256 consecutive frames and LSTM to output a deepfake probability value for the video.

Baseline convolutional models. We implemented 2 state-of the art CNN models for comparison: ResNet-50 [54] and XceptionNet [25]. XceptionNet is state-of-the-art and it is used in numerous implementations [1] [6] [29] [34].

To create a basis for comparison, we train the 2 mentioned architectures for the task of deepfake detection on both FaceForensics++ and CelebDF. We started from models pre-trained on the ImageNet [30] dataset and replaced the output layer with fully connected layers and a dropout layer. The models are fine-tuned during training.

We mentioned earlier that many deepfake detection architectures are image-level and do not include the evolution in time. Our CNN models are no exception. Therefore, it is important that we enhance them to include the temporal dimension. With that in mind, we use Long-Short Term Memory (LSTM) [23] cells to ensure that the time evolution of the videos is not omitted. LSTM architectures are state-of-the-art in temporal sequences like video thanks to their ability to handle both long and short term dependencies in the sequence. They are often time used for tasks like video classification, person reidentification, object tracking or generation of video descriptions and understanding video context.

In our latest work [34], we ran experiments which demonstrated the viability and performance increase of CNN-LSTM architectures over image-level CNN architectures. We will be using the same architecture for our experiments, while also adding an additional ResNet-50-LSTM architecture. We also use a similar approach for the input sequences and architectures, which is also presented in Fig. 2: as an input, we use groups of 60 frames, sampled as 6 frames/second (so, every 5^{th} frame of the video). We use pretrained XceptionNet or ResNet-50 models as feature extractors, for every frame. The resulted feature vectors go into a 2-layer LSTM which outputs a temporal descriptor. We use a 2-layer LSTM because our experiments showed a slight performance increase when using more than 1 layer, but a significant training time increase when using 3 or more. The temporal descriptor is fed into a group of decision fully connected layers, resulting in a prediction.

The architectures below are used for the CNN models we trained to help as a baseline for the evaluation of the CapsNet models. All the CNN architectures were pretrained on ImageNet and finetuned.

- ResNet-50 - model trained on both FaceForensics++ and CelebDF on image level.
- XceptionNet - model trained on both FaceForensics++ and CelebDF on image level.
- ResNet-50-LSTM - model trained on both FaceForensics++ and CelebDF on 60 frames 1/5 seconds apart.
- XceptionNet-LSTM - model trained on both FaceForensics++ and CelebDF on 60 frames 1/5 seconds apart.

Capsule Network Models. In this subchapter we will present all the architectures we tested and evaluated.

The models below represent all the Capsule Network architectures we used in the experiments. The VGG19 feature extractors used in the models were pretrained and finetuned while training. The majority of the models use 3 iterations for the dynamic routing algorithm and were trained and evaluated on CelebDF, unless stated otherwise.

- CapsNet-2caps - Capsule Network using 1800 8D primary capsules and 2 16D secondary capsules, one for each class. The loss function used was Capsule loss, identical to the one used in [46]. Used for training only on the FF++ dataset, due to the fact that performance on CelebDF was low (under 80% AUC).
- CapsNet-10caps - Capsule Network using 7200 primary capsules of length 8 and 10 secondary capsules + fully connected layers for decision.
- CapsNet-LSTM-16caps-60frames - Capsule Network using 2048 primary capsules of length 8, 16 secondary capsules, 2-layer LSTM with 256 features + fully connected layers for decision. The LSTM was trained alongside the Capsule Network.
- CapsNet-BigCaps - Capsule Network architecture using 7200 primary capsules of length 8 and 16 secondary capsules of length 16 + fully connected layers for decision. Biggest model, with size approximately equal to XceptionNet. Trained in 2 configurations: using 3 and 5 iterations for the dynamic routing algorithm.
- CapsNet-BigCaps-LSTM - using CapsNet-BigCaps for a feature extractor for every one of the first 256 consecutive frames for videos, then training a 2-layer, 256 features LSTM + fully connected layers for decision. The separate training sessions were due to a hardware limitation.
- CapsNet-BigCaps-GradientClipping - CapsNet-BigCaps architecture, without dropout but using gradient clipping for regularization. Gradient clipping value=1.
- CapsNet-MediumParam - Capsule Network architecture using 1568 primary capsules of length 8 and 16 secondary capsules of length 16 + fully connected layers for decision. Trained in 2 configurations: using 3 and 5 iterations for the dynamic routing algorithm.
- CapsNet-MediumParam-8caps - identical to CapsNet-MediumParam, but using 8 secondary capsules instead of 16
- CapsNet-LowParam - Capsule Network architecture using 784 primary capsules of length 8 and 8 secondary capsules of length 16 + fully connected layers for decision. Also using a smaller part of the VGG19 - only the first 5 convolutional layers.
- CapsNet-LowestParam - Capsule Network architecture using 121 primary capsules of length 8 and 8 secondary capsules of length 16 + fully connected layers for decision. Also using a smaller part of the VGG19 - only the first 3 convolutional layers. More than that, rather than using parallel convolution primary capsule layers to create the elements for the primary capsules, we use 2 convolutional layers with a high stride value to reduce the size of the data and we reshape the resulted data in the form of capsules. The total number of parameters for this model is under half a million.
- CapsNet-Narrow-32 - A more narrow Capsule Network architecture. Uses 32 primary capsules of length 1024 and 16 secondary capsules of length 16 + fully connected layers for decision. Training time for this model is 8-10 times the training time for the wide primary capsules models.

Training and hyparameters. For the CNN and CNN-LSTM models using XceptionNet and ResNet-50, we used an approach similar to our previous work [34]: we trained the models using a $5x10^{-4}$ value for the learning rate with a 0.92 learning rate decay factor, with an Adam optimizer and a batch size of 32. We use weight decay=10^{-5} and dropout=0.2. The chosen loss function was Binary Cross-Entropy. We train the models for 25 epochs at most, using early stopping when needed.

For the Capsule Network models, we use a learning rate or 10^{-3} which decays faster, with learning rate decay factor between 0.8 and 0.5. That is because Capsule Networks had the tendency to learn faster than CNN approaches. We also use an Adam optimizer, without weight decay. The batch size for training is 16. A more aggressive dropout value of 0.3

TABLE I
COMPARISON WITH STATE-OF-THE-ART METHODS FOR FACEFORENSICS++

Author	Model	AUC[%]
Nguyen *et al* [50]	XceptionNet	93.11
Qi *et al* [39]	DeepRythm	98
Tolosana *et al* [24]	XceptionNet	99.4
Tolosana *et al* [24]	CapsNet	99.52
Proposed	XceptionNet	99.63
Proposed	ResNet-50	99.67
Proposed	ResNet-50-LSTM	99.35
Stanciu *et al* [34]	XceptionNet-LSTM	99.95
Proposed	CapsNet-2caps	**99.99**

is used to help with regularization. The models were trained for 15 epochs at most, with early stopping, due to the fact that our Capsule Network models learned faster that the CNN models. Lastly, we tried a variable number of dynamic routing iterations for some architectures, between 3 and 5.

Experimental results. In this part of the paper, we will present our results and findings. We will compare the performance of our models to the state-of-the-art results on both datasets, considering both precision and the number of parameters used. We used a machine with a Nvidia 1070Ti 8GB graphics card and 16GB RAM for all the experiments below.

Table I presents a comparison between our proposed methods and the state of the art on the FaceForensics++ dataset. We can observe from the results that the dataset is fairly easy, given that almost all presented methods achieve an AUC of over 99%. Our CapsNet architecture with 2 class capsules achieved an almost perfect AUC score of 99.99%, higher than all the CNN-based approaches. We can also see from the state-of-the-art results that the Capsule Network also performed better than all the presented convolutional neural networks. One conclusion we can draw from this is that, on easy datasets, Capsule Networks are comparable or even better than other CNN methods.

Table II presents the results for our proposed approaches compared to other state-of-the-art approaches, on the CelebDF dataset. First of all, we can see that approaches based on XceptionNet also achieve very good results on this dataset. The metric-learning XceptionNet proposed by Kumar *et al* [41] achieves impressive results, crossing the 99% AUC mark. For image-level approaches, this approach is superior to our own convolutional approaches, which have an almost 10% lower performance.

Our proposed baseline CNN approaches, using ResNet-50 and XceptionNet were best utilized at video level, using LSTM to handle the time dimension. The best results are obtained using our previous approach in [34], with Xception-LSTM. However, the other state-of-the-art approaches in the state of the art by De Lima *et al* [40] clearly outperform them. By using 3D convolutions to handle groups of frames, these approaches surpass all other convolutional approaches, with AUC values over 99% for 3 of the 4 models.

TABLE II
COMPARISON WITH STATE-OF-THE-ART METHODS FOR CELEBDF

Author	Model	AUC[%]	Parameters
Nguyen *et al* [50][2]	CapsNet	57.5	3.9 M
Tolosana *et al* [24]	CapsuleNet	82.46	3.9 M
Tolosana *et al* [24]	XceptionNet	83.6	22.8 M
Kumar *et al* [41]	Xception metric-learning	99.2	22 M
De Lima *et al* [40]	I3D	97.59	12.29 M
De Lima *et al* [40]	MC3	99.3	11.49 M
De Lima *et al* [40]	R2Plus1D	99.43	31.30 M
De Lima *et al* [40]	R3D	99.73	33.17 M
Proposed	ResNet-50	90.02	23 M
Proposed	XceptionNet	90.68	22.8 M
Proposed	ResNet-50-LSTM	95.49	24 M
Stanciu *et al* [34]	Xception-LSTM	97.06	24 M
Proposed	CapsNet-BigCaps-LSTM	98.85	24 M
Proposed	CapsNet-BigCaps 5 iterations	**99.88**	22.4 M
Proposed	CapsNet-BigCaps 3 iterations	98.95	22.4 M
Proposed	CapsNet-BigCaps GradientClipping	97.21	22.4 M
Proposed	CapsNet-10caps	98.94	16.8 M
Proposed	CapsNet-LSTM-16caps 60frames	99.27	12.9 M
Proposed	CapsNet-MediumParam 5 iterations	99.56	6.4 M
Proposed	CapsNet-MediumParam	97.79	6.4 M
Proposed	CapsNet-MediumParam 8caps	97.65	4.8 M
Proposed	CapsNet-LowParam	97.74	1.9 M
Proposed	CapsNet-LowestParam	63.26	0.47 M
Proposed	CapsNet-Narrow-32	92.47	17.8 M

One of the main goals of this paper is to see if Capsule Networks can outperform traditional CNN approaches. Previous results using Capsule Networks on the CelebDF dataset indicate that this is not the case. In the state of the art, the best performance of 82.46% with CapsNet was achieved in [24], by using the Capsule Networks algorithm proposed by Nguyen *et al* [50].

Our first idea was to evaluate the performance or a CapsNet architecture with the same number of weights as the state-of-the-art XceptionNet architecture. Our "BigCaps" models aim to do exactly that, as they all use approximately the same the number or parameters in the XceptionNet architecture. On frame-level approaches, our "BigCaps" architecture using 3 iterations for the dynamic routing algorithm achieved performances just shy of the ones presented in [41], which use XceptionNet with metric learning. However, our CapsNet-BigCaps architecture with 5 iterations used for the dynamic

routing algorithm achieved better than state-of-the-art results, reaching 99.88% AUC. This model also outperforms the previous state-of-the-art models presented in [40], which use 3D convolutional networks to also tackle the temporal dimension. The following conclusions can be drawn from the experiments using our CapsNet-BigCaps architectures:

- Capsule Networks are capable of outperforming state-of-the-art models in deepfake detection, while using less parameters. Our CapsNet-BigCaps-5iterations model outperformed the best XceptionNet models by using slightly less trainable weights and also outperformed video-level approaches using 3D CNN, using almost 33% less parameters than the best approach (R3D [40]).
- Using more iterations for the dynamic routing algorithm in CapsNet architectures increases performance without the need for bigger architectures. Using 5 iterations instead of 3 for the same architecture increased the performance from 98.95% AUC to 99.88% AUC. While using more loops certainly increases the prediction time, deepfake detection is not usually a real-time task, so this is not a significant problem.
- Using gradient clipping in Capsule Networks leads to poorer performance, due to the fact that exploding gradients are not a concern.
- While leveraging the time component with LSTM did not achieve better results, we must consider the following caveats regarding hardware limitations: (1) CapsNet-LSTM-16caps architecture uses a smaller number of parameters, due to hardware limitations, and therefore should not be directly compared to the BigCaps architectures with almost twice as much parameters. In spite of that, this model still achieves better results that the image-level CapsNet-BigCaps architectures with 3 iterations; (2) The LSTM from CapsNet-BigCaps-LSTM was trained separately, using features extracted from the secondary capsules, due to hardware limitations. So, while the results are good, it makes sense that they do not necessarily outperform other architectures because the CapsNet was not trained alongside the LSTM.

We also demonstrated that Capsule Networks outperform convolutional approaches when using a small number of parameters. For this we tested progressively smaller architectures, from CapsNet-MediumParam which uses 6.4 million parameters to a very small CapsNet architecture, with only 470,000 paramaters. The following conclusions can be drawn from the experiments:

- While the number of parameters stayed over 1.9 M, smaller Capsule Networks perform better than the majority of CNN architectures. They achieve results over 97% AUC, which could be enough to consider the performance-size trade-off as being favorable.
- There is not a big difference between architectures using double the amount of primary capsules (LowParam vs MediumParam models with the same number of iterations).
- The majority of parameter cuts came from using less primary capsules. However, in the CapsNet-LowestParam we also lowered number of the convolutional layers from VGG19 feature extractor. This, combined with an even smaller number of primary and secondary capsules, resulted in a drastic drop in performance, to 63.26% AUC.
- Using more iterations in the dynamic routing algorithm increases performance even in models with a smaller number of parameters. In this situation, we can see that the performance of the CapsNet-MediumParam model rose significantly by just changing the number of iterations, while the number of parameters remained the same. The model with 5 iterations achieved a 99.56% AUC performance, which is very close to the best results for convolutional models in the state of the art, while only using 6.4 million parameters - just over half of the parameters in the I3D architecture in [40] or under 20% of the parameters in the R3D architecture [40].

Lastly, we used a very narrow model with large capsules to see if a big number of features in a primary capsule vector would increase performance. While the results are less impressive compared to the other wide architectures, a 92% AUC is better than many other results in our CNN models or in the state of the art. Therefore, we can hypothesise that there is an optimum configuration, given the trade-off between the length of the capsule vectors and the width of the model.

V. Conclusion

In this paper, we proposed several Capsule Network architectures to aid the task of deepfake detection. We bring several improvements to the Capsule Network models beyond the state of the art for deepfake detection. We compere our model's performances with other proposed convolutional and Capsule Network architectures. The models were trained on 2 state-of-the-art datasets: FaceForensics++ and CelebDF. For Celeb-DF, our best model achieved 99.88% AUC, a better performance than the best state-of-the-art models, while using 33% less trainable parameters. We also demonstrated that Capsule Networks can be effective in the deepfake detection task without using a large number of parameters, as several of our low-parameter approaches achieved an AUC greater than 97% for CelebDF, with one CapsNet architecture reaching 99.56% AUC. We concluded that using CapsNet architectures with an increased number of iterations for the dynamic routing algorithm increases performance, while keeping the number of parameters constant. For future improvements, we plan to use other novel Capsule Network architectures on bigger and more complex datasets. Also, we want to study the length-width trade-off for the primary capsules of a Capsule Network.

Acknowledgment

This work was supported under project AI4Media, A European Excellence Centre for Media, Society and Democracy, H2020 ICT-48-2020, grant #951911.

REFERENCES

[1] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, Matthias Nießner, "FaceForensics++: Learning to Detect Manipulated Facial Images," in 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019 pp. 1-11.

[2] Y. Li, X. Yang, P. Sun, H. Qi and S. Lyu, "Celeb-DF: A Large-Scale Challenging Dataset for DeepFake Forensics," in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020 pp. 3204-3213.

[3] Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A. Bengio, "Generative Adversarial Networks", 2014.

[4] Faceapp, https://www.faceapp.com, (Accessed on 16/04/2021).

[5] Faceswap-GAN, https://github.com/shaoanlu/faceswap-GAN, (Accessed on 16/04/2021).

[6] Brian Dolhansky, Joanna Bitton, Ben Pflaum, Jikuo Lu, Russ Howes, Menglin Wang, Cristian Canton Ferrer, "The DeepFake Detection Challenge (DFDC) Dataset", 2020.

[7] DeepFake Detection Challenge, https://www.kaggle.com/c/deepfake-detection-challenge, (Accessed on 16/04/2021).

[8] Wang, W.; Dong, J.; Tan, T., "Tampered Region Localization of Digital Color Images". Digital Watermarking: 9th International Workshop, IWDW 2010. Seoul, Korea: Springer. pp. 120–133.

[9] This Person Does Not Exist, https://thispersondoesnotexist.com, (Accessed on 16/04/2021).

[10] J. Zhu, T. Park, P. Isola, A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," in 2017 IEEE International Conference on Computer Vision, Venice, Italy, pp. 2242-2251.

[11] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt and M. Niessner, "Face2Face: Real-Time Face Capture and Reenactment of RGB Videos," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016 pp. 2387-2395.

[12] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, Jaegul Choo, "StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation", 2018.

[13] Ivan Perov, Daiheng Gao, Nikolay Chervoniy, Kunlin Liu, Sugasa Marangonda, Chris Umé, Mr. Dpfks, Carl Shift Facenheim, Luis RP, Jian Jiang, Sheng Zhang, Pingyu Wu, Bo Zhou, Weiming Zhang, "DeepFaceLab: A simple, flexible and extensible face swapping framework", 2020.

[14] faceswap-GAN github. https://github.com/shaoanlu/faceswap-GAN, (Accessed on 16/04/2021).

[15] DeepFake TIMIT Dataset, https://www.idiap.ch/dataset/deepfaketimit, (Accessed on 16/04/2021).

[16] FaceSwap , https://github.com/MarekKowalski/FaceSwap, (Accessed on 16/04/2021).

[17] DeepFake FaceSwap, https://github.com/deepfakes/faceswap, (Accessed on 16/04/2021).

[18] B. Dolhansky, R. Howes, B. Pflaum, N. Baram, and C. Ferrer, "The Deepfake Detection Challenge (DFDC) Preview Dataset", 2019.

[19] H.H. Nguyen, J. Yamagishi and I. Echizen, "Use of a Capsule Network to Detect Fake Images and Videos", 2019.

[20] Y. Li and S. Lyu, "Exposing DeepFake Videos By Detecting Face Warping Artifacts," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2019.

[21] E. Sabir, J. Cheng, A. Jaiswal, W. AbdAlmageed, I. Masi, and P. Natarajan, "Recurrent Convolutional Strategies for Face Manipulation Detection in Videos," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2019.

[22] D. Güera and E. J. Delp, "Deepfake Video Detection Using Recurrent Neural Networks," 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Auckland, New Zealand, 2018, pp. 1-6.

[23] Shahroz Tariq, Sangyup Lee, Simon S. Woo, "A Convolutional LSTM based Residual Network for Deepfake Video Detection", 2020.

[24] Ruben Tolosana, Sergio Romero-Tapiador, Julian Fierrez and Ruben Vera-Rodriguez, "DeepFakes Evolution: Analysis of Facial Regions and Fake Detection Performance", 2020.

[25] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017 pp. 1800-1807.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory. Neural Computation", 9(8):1735–1780, Nov. 1997.

[27] T. Baltrusaitis, A. Zadeh, Y. Lim, and L. Morency, "OpenFace 2.0: Facial Behavior Analysis Toolkit," in Proc. International Conference on Automatic Face and Gesture Recognition, 2018.

[28] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016 pp. 2818-2826.

[29] H. Dang, F. Liu, J. Stehouwer, X. Liu, and A. Jain, "On the Detection of Digital Face Manipulation," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020.

[30] J. Deng, W. Dong, R. Socher, L. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255

[31] D. Guera and E. Delp, "Deepfake Video Detection Using Recurrent Neural Networks," in 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Auckland, New Zealand, pp. 1-6.

[32] DeeperForensics-1.0: A Large-Scale Dataset for Real-World Face Forgery Detection Liming Jiang, Ren Li, Wayne Wu, Chen Qian, Chen Change Loy

[33] S. Suwajanakorn, S. Seitz, and I. Kemelmacher-Shlizerman, "Synthesizing Obama: Learning Lip Sync From Audio," ACM Transactions on Graphics, vol. 36, no. 4, pp. 1–13, 2017.

[34] Deepfake Video Detection with Facial Features and Long-Short Term Memory Deep Networks Dan-Cristian Stanciu, Bogdan Ionescu

[35] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In ICLR, 2014.

[36] Leveraging Frequency Analysis for Deep Fake Image Recognition Joel Frank 1 Thorsten Eisenhofer 1 Lea Sch¨onherr 1 Asja Fischer 1 Dorothea Kolossa 1 Thorsten Holz

[37] Two-Stream Neural Networks for Tampered Face Detection Peng Zhou, Xintong Han, Vlad I. Morariu, Larry S. Davis

[38] In Ictu Oculi: Exposing AI Generated Fake Face Videos by Detecting Eye Blinking Yuezun Li, Ming-Ching Chang and Siwei Lyu Computer Science Department, University at Albany, SUNY

[39] DeepRhythm: Exposing DeepFakes with Attentional Visual Heartbeat Rhythms, Hua Qi1, Qing Guo2, Felix Juefei-Xu3, Xiaofei Xie2, Lei Ma1, Wei Feng4, Yang Liu2, Jianjun Zhao1

[40] Deepfake Detection using Spatiotemporal Convolutional Networks Oscar de Lima, Sean Franklin, Shreshtha Basu, Blake Karwoski, Annet George

[41] Akash Kumar and Arnav Bhavsar. Detecting deepfakes with metric learning. arXiv preprint arXiv:2003.08645, 2020.

[42] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In Proceedings of the IEEE International Conference on Computer Vision Workshops, pages 3154– 3160, 2017.

[43] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6299–6308, 2017.

[44] Going Deeper with Convolutions Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich

[45] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In International Conference on Artificial Neural Networks, pages 44–51. Springer, 2011.

[46] Dynamic Routing Between Capsules Sara Sabour Nicholas Frosst Geoffrey E. Hinton

[47] IMPROVED EXPLAINABILITY OF CAPSULE NETWORKS: RELEVANCE PATH BY AGREEMENT Atefeh Shahroudnejad† , Arash Mohammadi , and Konstantinos N. Plataniotis

[48] J. Su, D.V. Vargas, K. Sakurai One pixel attack for fooling deep neural networks IEEE Trans. Evol. Comput., 1–15 (2019),

[49] EFFECTIVE AND EFFICIENT VOTE ATTACK ON CAPSULE NETWORKS Jindong Gu1,3 , Baoyuan Wu2 , Volker Tresp1,

[50] USE OF A CAPSULE NETWORK TO DETECT FAKE IMAGES AND VIDEOS Huy H. Nguyen? , Junichi Yamagishi?†‡, and Isao Echizen?†§

[51] Nguyen H.H., Yamagishi J., Echizen I. (2022) Capsule-Forensics Networks for Deepfake Detection. In: Rathgeb C., Tolosana R., Vera-Rodriguez R., Busch C. (eds) Handbook of Digital Face Manipulation and Detection. Advances in Computer Vision and Pattern Recognition. Springer, Cham.

[52] Rahmouni N, Nozick V, Yamagishi J, Echizen I (2017) Distinguishing computer graphics from natural images using convolution neural networks. In: International workshop on information forensics and security (WIFS). IEEE
[53] I. Chingovska, A. Anjos, and S. Marcel, "On the effectiveness of local binary patterns in face anti-spoofing," in International Conference of the Biometrics Special Interest Group (BIOSIG), 2012.
[54] Deep Residual Learning for Image Recognition Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
[55] Rethinking the Inception Architecture for Computer Vision Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna
[56] Very Deep Convolutional Networks for Large-Scale Image Recognition Karen Simonyan, Andrew Zisserman
[57] Capsule Networks – A survey Author links open overlay panelMensahKwabena PatrickaAdebayoFelix AdekoyaaAyidzoeAbra MightybBaagyire Y.Edwardc