**REPUBLIC OF TURKEY**
**YILDIZ TECHNICAL UNIVERSITY**
**GRADUATE SCHOOL OF SCIENCE AND ENGINEERING**

# STUDYING DEEP LEARNING MODELS FOR MANIPULATED FACE DETECTION

**İlkin HÜSEYNLİ**

MASTER OF SCIENCE THESIS
Department of Computer Engineering
Program of Computer Engineering

Supervisor
Prof. Dr. Songül VARLI

June, 2021

**REPUBLIC OF TURKEY**
**YILDIZ TECHNICAL UNIVERSITY**
**GRADUATE SCHOOL OF SCIENCE AND ENGINEERING**

# STUDYING DEEP LEARNING MODELS FOR MANIPULATED FACE DETECTION

A thesis submitted by İlkin HÜSEYNLİ in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** is approved by the committee on 30.06.2021 in Department of Computer Engineering, Program of Computer Engineering.

Prof. Dr. Songül VARLI
Yildiz Technical University
Supervisor

**Approved By the Examining Committee**

Prof. Dr. Songül VARLI, Supervisor
Yildiz Technical University _____

Assist. Dr. OĞUZ ALTUN, Member
Yildiz Technical University _____

Assoc. DR. TAMER DAĞ, Member
Kadir Has University _____

I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of STUDYING DEEP LEARNING MODELS FOR MANIPULATED FACE DETECTION supervised by my supervisor, Prof. Dr. Songül VARLI. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

İlkin HÜSEYNLİ

Signature

*Dedicated to my family*

# ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to my family. Without their tremendous support and encouragement, it would be impossible for me to complete my thesis. I am incredibly grateful to my supervisor, Prof. Dr. Songül VARLI, for her directives and academic advice, which kept me on the track of completion. Finally, I'm thankful to my close friends Redzhep M. REDZHEBOV for his precious help on technical subjects and Ayten AHMADLI for motivating me at challenging times.

İlkin HÜSEYNLİ

# TABLE OF CONTENTS

**LIST OF ABBREVIATIONS** vii

**LIST OF FIGURES** viii

**LIST OF TABLES** x

**ABSTRACT** xi

**ÖZET** xiii

**1 INTRODUCTION** 1

1.1 Literature Review . . . . . 1

1.2 Objective of the thesis . . . . . 3

1.3 Hypothesis . . . . . 3

**2 METHODOLOGY AND FAKE FACE DATASETS** 4

2.1 Models . . . . . 4

2.1.1 ResNet . . . . . 4

2.1.2 MobileNet-V3 . . . . . 5

2.1.3 DenseNet . . . . . 6

2.1.4 Xception . . . . . 7

2.1.5 Inception models . . . . . 8

2.1.6 EfficientNet models . . . . . 8

2.2 Dataset . . . . . 11

2.2.1 DFDC . . . . . 11

2.2.2 Celeb-DF-V2 . . . . . 13

2.3 Augmentations . . . . . 13

2.4 Data preparation . . . . . 15

2.4.1 Train data extraction pipeline . . . . . 15

2.4.2 Test data extraction pipeline . . . . . 15

2.5 Training . . . . . 16

2.5.1 Half-precision . . . . . 17

**3 RESULTS AND DISCUSSION** 18

3.1 DFDC results . . . . . 20
3.1.1 Distractor and augmenter analysis . . . . . 22
3.2 Celeb-DF-v2 results . . . . . 29
3.3 Discussion . . . . . 31

**REFERENCES** **33**

**A Environment setup** **36**
A.1 Hardware . . . . . 36
A.2 Software . . . . . 36

**PUBLICATIONS FROM THE THESIS** **37**

# LIST OF ABBREVIATIONS

| | |
|---|---|
| CNN | Convolutional Neural Network |
| DFDC | DeepFake Detection Challenge |
| FP16 | Floating-point 16 (a.k.a. Half Precision) |
| FP32 | Floating-point 32 (a.k.a. Single Precision) |
| GAN | Generative Adversiral Network |
| IoU | Intersection over Union |

# LIST OF FIGURES

**Figure 2.1** Main building block of the ResNet model [13] . . . . . 5
**Figure 2.2** Architecture of the MobileNet-V3-Large model [14] (SE: Squeeze-And-Excite; NL type of non-linearity: HS - h-swish, RE - ReLU; NBN - no batch normalization) . . . . . 5
**Figure 2.3** Sample Densenet block [15] . . . . . 6
**Figure 2.4** Modified depthwise separable convolutions [15] . . . . . 7
**Figure 2.5** Overall architecture of Xception model [16] . . . . . 8
**Figure 2.6** Architecture of Inception-v4 [18] . . . . . 9
**Figure 2.7** Architecture of Inception-Resnet-v2 [18] . . . . . 9
**Figure 2.8** EfficientNets parameter and ImageNet Top-1 Accuracry comparison with other models . . . . . 10
**Figure 2.9** EfficientNet-B0 architecture . . . . . 10
**Figure 2.10** Examples of the augmenters and distractors applied to the DFDC test set. Left top is the original image.[23] . . . . . 12
**Figure 2.11** Examples from the Celeb-DF-v2 dataset [24]. Left (green) column is real face and other columns (red) are different DeepFake of the real person with various source subjects. . . . . 13
**Figure 2.12** Augmentations: From top left: Source image, compression, Gauss noise, blur, sharpening, horizontal flip, brightness, hue and saturation value adjustment, gray, shift+scale+rotate, Coarse dropout (size=64 pixels), Grid dropout. All images resized to 224 by preserving aspect ratio and padded with 0 value before an augmentation was applied. . . . . 14
**Figure 2.13** Train loss over epochs of the models trained with fixed Learning rate and default augmentation setup. . . . . 16
**Figure 2.14** Validation loss over epochs of the models trained with fixed Learning rate and default augmentation setup. (Note, because the EfficientNet-B4 model overfitted so quickly, validation loss became NaN; thus, it was excluded from this graphic . . . . . 16
**Figure 2.15** Training time per epoch of EfficientNet-B4 with FP16 and FP32 . . 17

**Figure 3.1** Equal Error Rate for the EfficientNet-B4 model trained with Default augmentation. . . . . 19
**Figure 3.2** Calculated threshold values for the best model/augmentation combinations. . . . . 19
**Figure 3.3** Video based log loss of the trained models without any augmentation (lower is better). . . . . 20
**Figure 3.4** Video based log loss of the trained models with default augmentation chain (lower is better). . . . . 21
**Figure 3.5** Video based log loss of the trained models with default augmentation chain and grid dropout (lower is better). . . . . 21
**Figure 3.6** Video based log loss of the trained models with default augmentation chain and coarse dropout (lower is better). . . . . 22
**Figure 3.7** RoC-AuC curves for the best configuration of each model. . . . . 23
**Figure 3.8** Fake face examples from DFDC test set classified as real . . . . . 28
**Figure 3.9** Real face examples from DFDC test set classified as fake . . . . . 28
**Figure 3.10** F1-score for the best configuration for each model. Except for the Xception model, all models have the best score with default + Coarse augmentation. Xception model has the best result with only default augmentation. . . . . 30
**Figure 3.11** Examples of misclassified fake faces from Celeb-DF-v2 dataset . . . 30
**Figure 3.12** Examples of misclassified real faces from Celeb-DF-v2 dataset . . . 31

# LIST OF TABLES

**Table 2.1** Number of trainable parameters . . . . . 4
**Table 2.2** Order and occurrence probability of the applied augmentations. . . 14
**Table 2.3** Training EfficientNet-B4 model with half-precision and single precision 17
**Table 3.1** Log loss results of the DFDC dataset (lower is better). *Italic* values are the best result for the augmentation and **bold** values are the best result of the model . . . . . 22
**Table 3.2** Percentage of misclassified fake videos per distractor for each model's best setup. **Bold** values are the worst results for each model. 24
**Table 3.3** Percentage of misclassified real videos per distractor for each model's best setup. **Bold** values are the worst results for each model. . . . . 24
**Table 3.4** Percentage of the misclassified fake videos for each model's best configuration per augmenter. **Bold** values are the worst result for each model. . . . . 25
**Table 3.5** Percentage of the misclassified real videos for each model's best configuration per augmenter. Bold values are the worst result for each model. . . . . 26
**Table 3.6** Sensitivity and Specificity results of Celeb-DF-v2 dataset for each model and augmentation (higher is better). . . . . 27
**Table 3.7** F1-score of the models tested with Celeb-DF-v2. **Bold** values are the best results for the given model and italic values are the best results for an augmentation. . . . . 29

# ABSTRACT

## STUDYING DEEP LEARNING MODELS FOR MANIPULATED FACE DETECTION

İlkin HÜSEYNLİ

Department of Computer Engineering
Master of Science Thesis

Supervisor: Prof. Dr. Songül VARLI

Deepfakes allow users to manipulate the identity of a person in a video or an image. Previously, special hardware and skill were required to create such fake videos/images. But together with improvements on GAN-based techniques, generating more realistic and hard to detect manipulated faces became easier. This threatens individuals and decreases trust in social media platforms. In this work, our goal is to report eight different models' learning ability on, by far, the largest fake face dataset - DFDC and test the generalization ability of these models with Celeb-DF-v2. Because the training dataset consists of high-quality videos, we started detecting and extracting faces from them. Next, we sampled data to have balanced classes and a feasible amount of data to train with limited resources. We started training with no extra augmentation because the dataset was big enough, and faces were already modified. Next, we added our default augmentation chain, inspired by other works and increased strength with Coarse-Dropout and Grid Mask augmentations.

A separate test set from the DFDC dataset, which has unseen augmentations and distractors and a completely different Celeb-DF-v2 dataset, was used to evaluate results. As distinct from the train set, we followed different face extraction flow for the test sets. We issued face tracking by using simple Intersection over the Union and sampled faces that only tracked over a certain number of consecutive faces. For each video in the test set, the confidence of the sampled faces averaged, and a single confidence value was generated. To calculate video-based log loss values, we used this confidence values. For the Celeb-DF-v2 dataset, we also calculated Sensitivity and Specificity values. For these metrics, the optimal threshold was decided by

using Equal Error Rate. We concluded that despite the relatively smaller size input EfficientNet-B4 model has the best learning and generalization ability. Training models with half-precision may speed up training time up to 2 times with very few losses. Finally, Coarse Dropout helped models to generalize better.

**Keywords:** digital video forensics, face manipulation, deepfake, face swap

# ÖZET

## SAHTE YÜZ TESPİTİ İÇİN DERİN ÖĞRENME MODELLERİNİN ARAŞTIRILMASI

İlkin HÜSEYNLİ

Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Danışman: Prof. Dr. Songül VARLI

Deepfakes, kullanıcıların bir video veya görüntüdeki bir kişinin kimliğini değiştirmesine olanak tanır. Önceden, bu tür sahte videolar/görüntüler oluşturmak için özel donanım ve beceri gerekiyordu. Ancak GAN tabanlı tekniklerdeki iyileştirmelerle birlikte, daha gerçekçi ve algılaması zor manipüle edilmiş yüzler oluşturmak daha kolay hale geldi. Bu da bireyleri tehdit etmekte ve sosyal medya platformlarına olan güveni azaltmaktadır. Bu çalışmada amacımız, bugüne kadarki en büyük sahte yüz veri seti olan DFDC üzerinde sekiz farklı modelin öğrenme kabiliyetini rapor etmek ve bu modellerin genelleme kabiliyetini Celeb-DF-v2 veri kümesi ile test etmektir. Eğitim veri seti yüksek kaliteli videolardan oluştuğu için ilk olarak yüzleri tespit etmeye ve çıkarmaya başladık. Daha sonra, dengeli sınıflara sahip olmak, sınırlı kaynaklarla makul sürede eğitmek için verileri örnekledik. Veri kümesi yeterince büyük olduğundan ve yüzler zaten değiştirilmiş olduğundan, ekstra yapay artırmak olmadan eğitime başladık. Ardından, diğer çalışmalardan ilham alan varsayılan büyütme zincirimizi ekledik ve "Coarse Dropout" ve "Grid Mask" güçlendirmeleriyle gücü artırdık.

Sonuçları değerlendirmek için DFDC veri setinden, görünmeyen yapay artırmak ve çeldiricilere sahip ayrı bir test seti ve tamamen farklı bir Celeb-DF-v2 veri seti kullanıldı. Eğitim veri setinden farklı olarak, test setleri için farklı yüz çıkarma akışları izledik. Videodakı yüzleri basit Birleşimlerin Kesişimi (Intersection over Union) yöntemi ile takip ederek, belirli miktardakı ardışık sahnelerde yer alan yüzlerden örnekleme yaptık. Test setindeki her video için örneklenen yüzlerin güvenirliğinin

ortalaması alındı ve tek bir güven değeri üretildi. Video tabanlı log kaybı değerlerini hesaplamak için bu güven değerlerini kullandık. Celeb-DF-v2 veri setindeki sınıflar dengesiz olduğu için Duyarlılık ve Özgüllük değerlerini de hesapladık. Bu metrikler için, Eşit Hata Oranı kullanılarak optimal eşiğe karar verildi. Nispeten daha küçük boyutlu girdiye rağmen EfficientNet-B4 modelinin en iyi öğrenme ve genelleme yeteneğine sahip olduğu sonucuna vardık. Yarı hassas (half-precision) eğitim modelleri, çok az kayıpla eğitim süresini 2 kata kadar hızlandırabilir. Son olarak, Coarse Dropout, modellerin daha iyi genelleştirilmesine yardımcı oldu.

**Anahtar Kelimeler:** video adli bilişim, yüz manipülasyonu, deepfake, yüz değiştirme

# 1
# INTRODUCTION

## 1.1 Literature Review

Tolosana R. et al. [1] did a comprehensive review of fake face generation methods and methods to detect them. They divided manipulations into 4 categories: $i$) entire face synthesis, $ii$) identity swap, $iii$) attribute manipulation and $iv$) expression swap. The DFDC dataset lays in the identity swap category. Thus we are mainly focused on the methods for identity swapping. They grouped identity swap datasets under two categories according to the visual quality of the data.

Bonettini, N. et al. [2] trained four different models and evaluated them on 2 different datasets: Face Forensics++ [3] and the preview version of DFDC dataset. Considering the model size, run time, and classification performance, they decided to choose the EfficientNetB4 model as a baseline. Alongside with original network, they also created another version with an Attention block. They trained both models in different ways: end-to-end classical training and Siamese training. Finally, all these four variations were ensembled in different combinations. Ensemble of all four models had the best result and submitted to the DFDC Kaggle challenge [4] by authors which finished in $41^{st}$ place.

Wang et al. [5] propose a universal detector for CNN-generated fake images. They first generated fake images by using ProGAN [6] model and trained a binary classifier on this set. For evaluation purposes, they created a new dataset - *ForenSynths* which consists of images generated from 11 different models. One major difference of this study is, they were not just focused on faces but included various objects and scenes in their dataset. Unfortunately, internal structure of the classifier was not specified.

Neves, J.C. et al.[7] proposes that fake images generated with GAN's have unique GAN "fingerprints," and this makes detection easy. In the study, together with a survey of manipulated face detection works, they also introduced an auto-encoder-based strategy (GANprintR) to remove GAN "fingerprints" and a novel dataset named

iFakFaceDB with fingerprints deleted without affecting the quality of the resulting images in this work. They applied their technique to two real face image datasets and three synthetic face images: CASIA-WebFace, VGGFace2, TPDNE, 100K-Faces, and PGGAN. There were three methods: XceptionNet based binary detector, Steganalysis, and Local Artifacts applied before and after fingerprint was removed. They showed that models trained and tested on the same dataset learn the "fingerprints" of the generator network, and by removing it, the models can be spoofed.

Nguyen, T.T. et al. [8] did a survey on deepfake creation and detection methods. They studied detection methods for fakes under two headlines: Detection in images and detection in videos. They divided detection in videos section into 2 more sections : *Temporal Features across Video Frames* and *Visual Artifacts within Video Frame*.Works examined under the first section mainly focused on detection changes across frames such as eye blink times. These works mainly used Recurrent Neural Networks, CNN, LSTM, or a combination of them. Methods under Visual Artifacts within Video Frame section mainly focused on frame-based detection. These methods that try to detect images generally have artifacts and inconsistencies around the face, eyes, teeth, false shadowing, and missing reflections. One studied method in the survey paper focuses on detecting PRNU - Photo Response Non-Uniformity, which were considered the fingerprints of digital images by using statistical methods.

Tolosana et al. [9] also did an analysis of facial regions on deepfake images. They trained Xception and Capsule Network models with different 1st and 2nd generation datasets on the entire face, only eyes, nose, mouth, and rest of face (excluding eyes, nose, and mouth). Their models achieved 20-32% equal error rate on DFDC Preview Dataset [10]. Considering various conditions and extreme poses, finding face landmarks and extracting parts introduces its own challenges and errors into the system.

StyleGAN [11] is a Generative Adversarial Network for image generation proposed by NVIDIA inspired by style transfer literature. They focused on re-designing the generator, added noise directly into the network to separate and control high-level details such as pose and illumination from low-level details such as hair and freckles.

FSGAN [12] can generate successful fake faces despite occlusions in front of the face, skin color, and gender difference of source and target videos and even from a target image. To achieve this, they first generate landmarks of guiding face with small pose changes, then used four different, dedicated generator models: the first model learns creating face reenactment from the source image, second one segments face from target image, third generator model is used to overcome occlusions and fulfill missing

parts and last generator used to combine images seamlessly.

## 1.2 Objective of the thesis

Modern deep learning algorithms have a wide range of use. One of the most interesting and popular applications is face manipulation. Together with improvements in hardware technologies, bigger, more powerful, and sophisticated fake generation methods emerged. These new techniques are capable of creating high-quality fake images, including human faces. Alongside powerful deepfake methods, techniques to detect them also improve, and more diverse and challenging datasets are released. One of the most recent and most extensive deepfake datasets was released by Facebook AI called DeepFake Detection Challenge Dataset. In this work, by using this dataset, we compared several models learning ability under different augmentations. We first trained and tested the models on the DFDC train and test set. Next, we compared their generalization ability on an entirely different and unseen dataset - Celeb-DF-v2.

## 1.3 Hypothesis

Fake face generation has several good applications, such as creating videos of someone who is not alive, re-filming scenes without a real actor, or generating a wide variety of characters for the animations. However, unfortunately, deep fakes are more popular with their improper use: creating fake videos and images for harmful purposes. Popular examples are: making politicians/celebrities "say" things they have never said and swapped one's face on inappropriate videos. Later, these videos can be used to either blackmail target or damage his/her public reputation by spreading it. It is crucial to detect and prevent this kind of thread before they are spread. In this work, we compared models and their generalization ability to prepare a baseline for such a prevention system. An automatic fake video control mechanism can build on top of this work's output to check if uploaded video/image contains any fake faces before they are released.

# 2
# METHODOLOGY AND FAKE FACE DATASETS

## 2.1 Models

**Table 2.1** Number of trainable parameters

| **Model** | **Params count (Million)** |
|---|---|
| ResNet50 | 23.5 |
| MobileNet-v3 | 4.2 |
| Xception | 20.8 |
| Inception-v4 | 41.1 |
| Inception-ResNet-v2 | 54.3 |
| DenseNet161 | 26.5 |
| EfficientNet-B0 | 4.0 |
| EfficientNet-B4 | 17.6 |

In this section, we briefly explained the architecture of the compared models. The number of trainable parameters for each model was given at Table 2.1.

### 2.1.1 ResNet

ResNet [13] is one of the most popular neural network architectures. The previous models were series of convolution kernels, poolings and fully connected layers. These were good enough to train small models. However, as the number of layers increases, the models suffer vanishing gradient problems and not learning. The authors showed that adding simple skip connection, model depth can be increased several times. The main building block of the ResNet was given in Figure 2.1. As seen from the figure, one connection skipped, and the output of a layer was added to the output of the next layer. So that, information from the earlier layers was carried to the further layers. In this work, we used the ResNet50 model, which 48 Convolution layers, 1 MaxPool and 1 Average Pool layer - total of 50 layers.

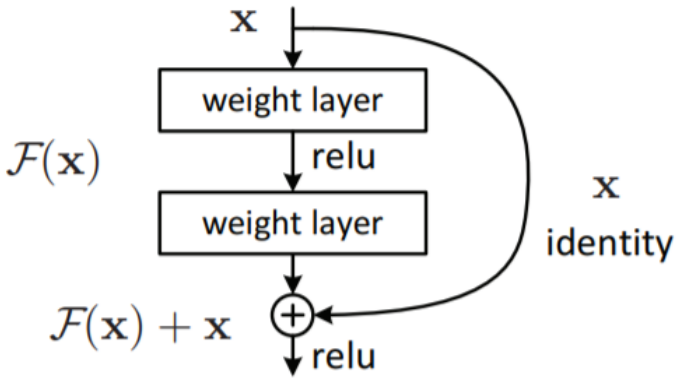


**Figure 2.1** Main building block of the ResNet model [13]

### 2.1.2 MobileNet-V3

MobileNet models were originally designed for mobile devices. The MobileNet-V3 [14] model is an improved version of the previous ones. The authors were used hardware-aware network architecture search and NetAdapt algorithm to find optimal architecture. Figure 2.2 shows the architecture of the MobileNet-V3-Large model, which was used in this work. The authors added new squeeze and excitation layers, did a block-wise and layer-wise search, introduced h-swish non-linearity, and optimized general architecture by removing some layers in order to improve and optimize the model.

| Input | Operator | exp size | $\#out$ | SE | NL | $s$ |
|---|---|---|---|---|---|---|
| $224^2 \times 3$ | conv2d | - | 16 | - | HS | 2 |
| $112^2 \times 16$ | bneck, 3x3 | 16 | 16 | - | RE | 1 |
| $112^2 \times 16$ | bneck, 3x3 | 64 | 24 | - | RE | 2 |
| $56^2 \times 24$ | bneck, 3x3 | 72 | 24 | - | RE | 1 |
| $56^2 \times 24$ | bneck, 5x5 | 72 | 40 | ✓ | RE | 2 |
| $28^2 \times 40$ | bneck, 5x5 | 120 | 40 | ✓ | RE | 1 |
| $28^2 \times 40$ | bneck, 5x5 | 120 | 40 | ✓ | RE | 1 |
| $28^2 \times 40$ | bneck, 3x3 | 240 | 80 | - | HS | 2 |
| $14^2 \times 80$ | bneck, 3x3 | 200 | 80 | - | HS | 1 |
| $14^2 \times 80$ | bneck, 3x3 | 184 | 80 | - | HS | 1 |
| $14^2 \times 80$ | bneck, 3x3 | 184 | 80 | - | HS | 1 |
| $14^2 \times 80$ | bneck, 3x3 | 480 | 112 | ✓ | HS | 1 |
| $14^2 \times 112$ | bneck, 3x3 | 672 | 112 | ✓ | HS | 1 |
| $14^2 \times 112$ | bneck, 5x5 | 672 | 160 | ✓ | HS | 2 |
| $7^2 \times 160$ | bneck, 5x5 | 960 | 160 | ✓ | HS | 1 |
| $7^2 \times 160$ | bneck, 5x5 | 960 | 160 | ✓ | HS | 1 |
| $7^2 \times 160$ | conv2d, 1x1 | - | 960 | - | HS | 1 |
| $7^2 \times 960$ | pool, 7x7 | - | - | - | - | 1 |
| $1^2 \times 960$ | conv2d 1x1, NBN | - | 1280 | - | HS | 1 |
| $1^2 \times 1280$ | conv2d 1x1, NBN | - | k | - | - | 1 |

**Figure 2.2** Architecture of the MobileNet-V3-Large model [14] (SE: Squeeze-And-Excite; NL type of non-linearity: HS - h-swish, RE - ReLU; NBN - no batch normalization)

Adding swish (Formula 2.1) non-linearity increases accuracy but comes with a

computation cost. The authors introduced hard-swish (h-swish) (Formula 2.2) non-linearity and used it only in the second half of the MobileNet model to overcome this problem.

$$swishx = x \cdot \sigma(x) \tag{2.1}$$

$$h-swish[x] = x\frac{ReLU6(x+3)}{6} \tag{2.2}$$

Platform aware NAS was applied to find optimal network blocks. An RNN controller predicts an architecture with a probability. Then, a network was trained to converge achieving given accuracy. Finally, architecture gradients scaled according to the accuracy. This kind of search is generally used to optimize parts of a model. Because MobileNet models were intended for mobile devices, both latency and accuracy were included in the reward function to find smaller models during architecture search. Output of the NAS was used as an input to the NetAdapt algorithm to optimize number of filters in each layer. As a metric, maximizing $\Delta accuracy/\Delta latency$ was chosen instead of just minimizing $\Delta accuracy$ as in the original paper.

### 2.1.3 DenseNet

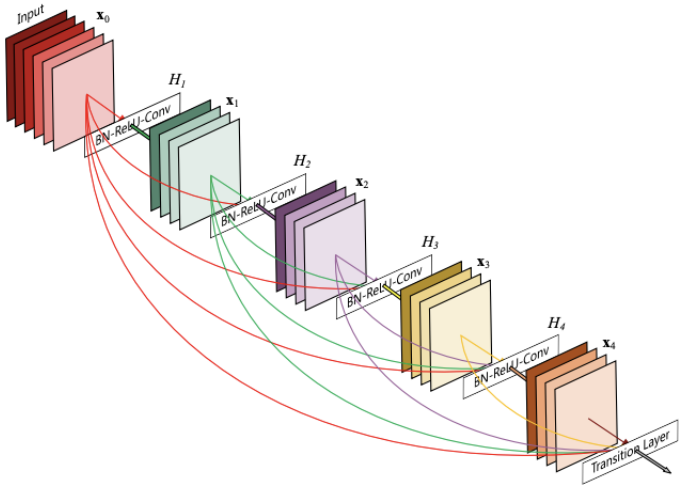


**Figure 2.3** Sample Densenet block [15]

The main idea of the Dense Convolutional Network or DenseNet[15] is connecting the output of each layer to the other layers in front of it, thus having "dense" connections. Figure 2.3 shows sample 5-layer dense block. These kinds of connections increase feature reuse and strengthen their propagation, reduces the vanishing-gradient

problem. At first glance, this architecture looks similar to ResNet, but there are few different key points: ResNet has one skip connection while DenseNet connects every subsequent layer; ResNet combines features by summing them; meanwhile, DenseNet concatenates features. In this work, we decide to use the DenseNet-161 variation, which has the lowest ImageNet top-1 error rate compared to other variations. This model consists of 4 Dense Blocks consisting of 6, 12, 36, and 24 BN-ReLU-Conv(1x1) and BN-ReLU-Conv(3x3) sequences. The authors define this sequence as Batch Normalization, ReLU, 1x1, and 3x3 convolutions.

### 2.1.4 Xception

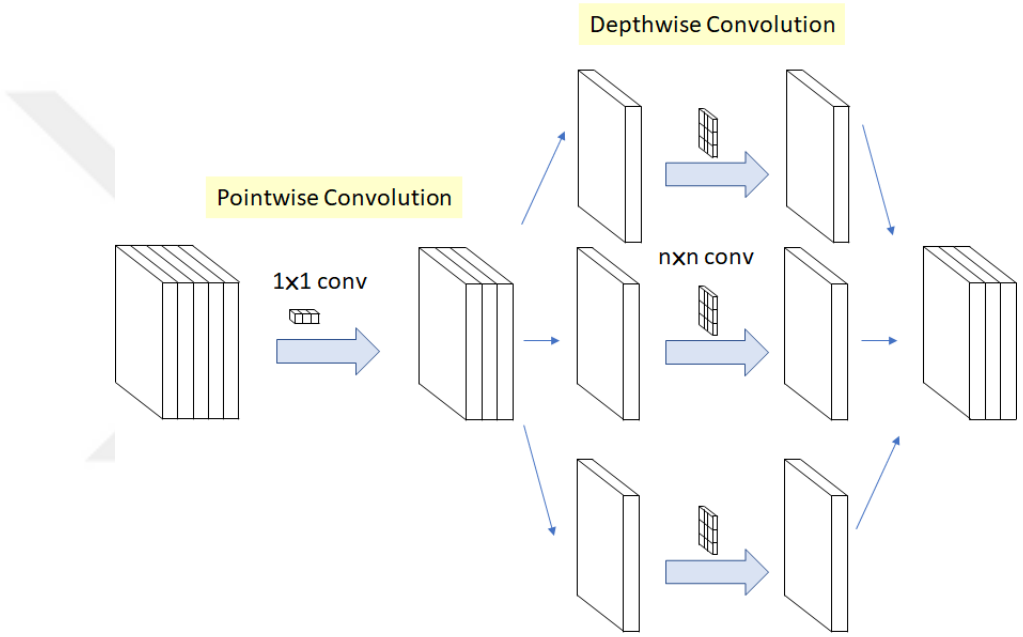


**Figure 2.4** Modified depthwise separable convolutions [15]

Xception [16] is an interpretation of Inception models introduced by François Chollet. It is an "extreme" version of the Inception-v3 [17] model with improved results. It uses modified depthwise separable convolutions, which are depicted in Figure 2.4. The modifications to the original method are:

1. The order of the operations - Pointwise Convolution was applied before Depthwise Convolution
2. There is no intermediate non-linearity. Original depthwise separable convolution has ReLU between operations

The purpose of the depthwise separable convolutions is to speed up the convolution process. The overall architecture of the Xception model was given in Figure 2.5.

The author defines Xception architecture as "a linear stack of depthwise separable convolution layers with residual connections."

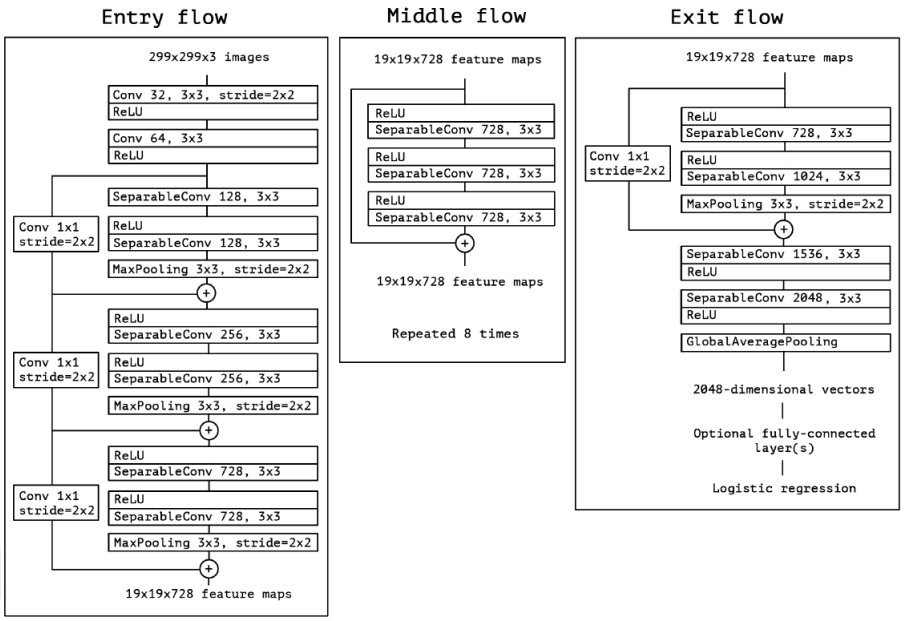


**Figure 2.5** Overall architecture of Xception model [16]

### 2.1.5 Inception models

Inception-v4 [18] is an updated version of the Inception-v3 [17] model. In this work, we compared two types of it: First pure Inception-v4 (denoted as inception_v4) which is simplified version of the previous and uses more inception modules and combination of Inception V3 and Residual networks [13] version (denoted as inception_resnet_v2). According to the original paper, both models are computationally equal. Architecture of the Inception-v4 model was given in Figure 2.6 and Inception-ResNet-v2 in Figure 2.7.

### 2.1.6 EfficientNet models

EfficinetNet[19] introduced by Google Research Team in late 2020. They proposed that adjusting network width, depth, and resolution simultaneously with certain formulas can increase performance. Also, proportional scaling significantly decreases the number of trainable parameters in a model while preserving performance. Their biggest model - EfficientNet-B7, achieved Imagenet Top-1 accuracy while having 8.4x fewer parameters and running 6.1x faster than the best model at that time. Figure 2.8 represents comparison of the EfficientNet models' parameters and accuracy with other top models.

As a backbone, the authors were used *mobile inverted bottleneck - MBConv* [20]. Figure

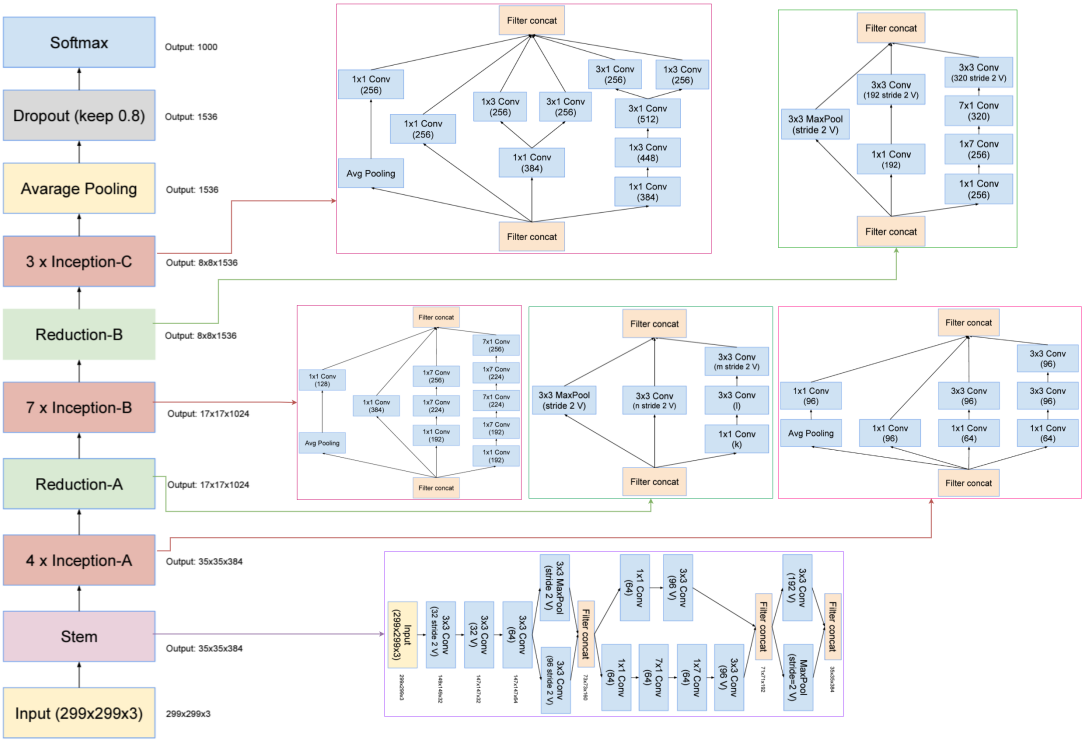


**Figure 2.6** Architecture of Inception-v4 [18]

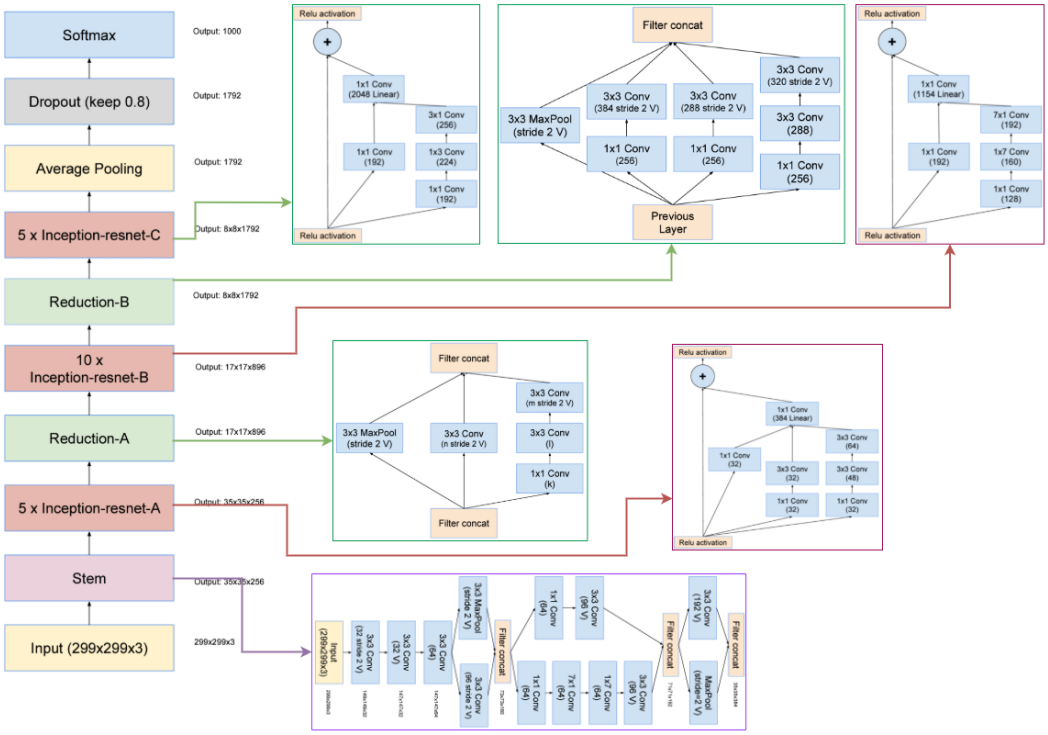


**Figure 2.7** Architecture of Inception-Resnet-v2 [18]

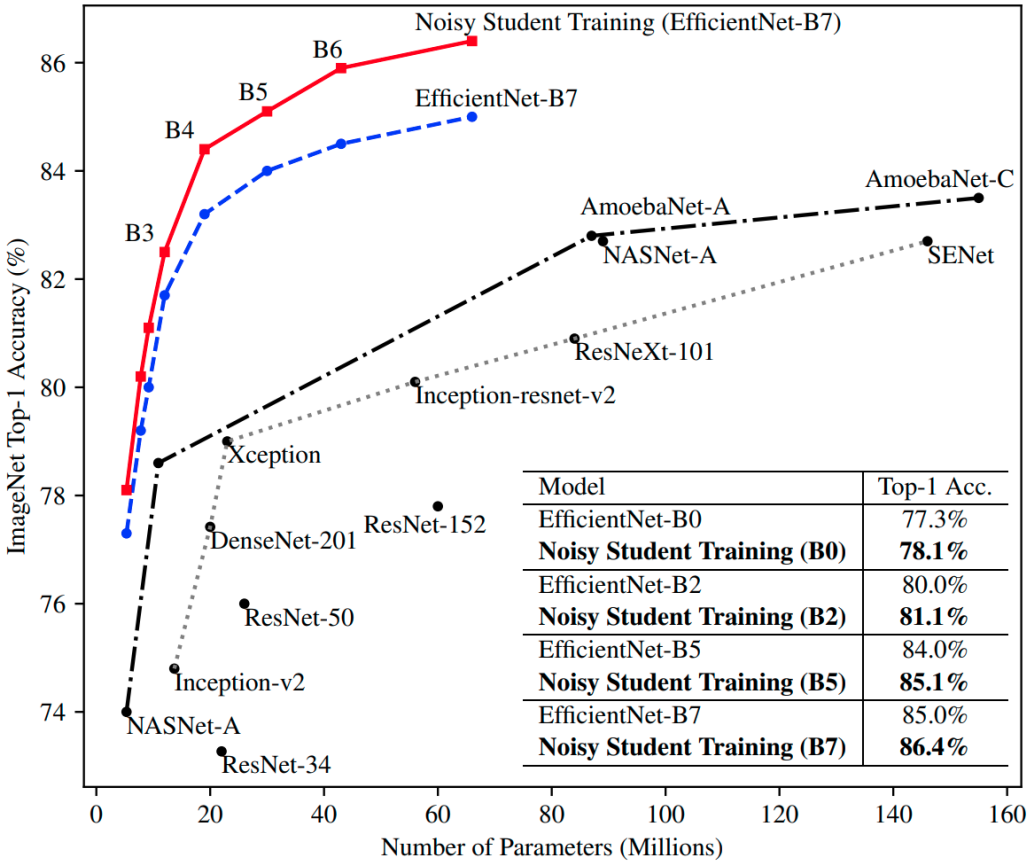


**Figure 2.8** EfficientNets parameter and ImageNet Top-1 Accuracry comparison with other models

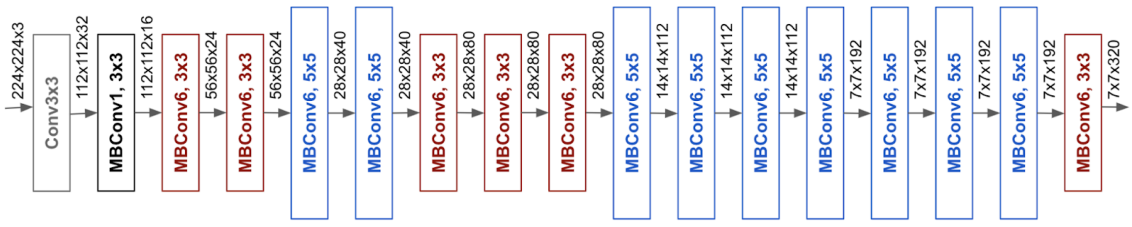


**Figure 2.9** EfficientNet-B0 architecture

2.9 was borrowed from the Google AI Blog post[21] and represents architecture of EfficientNet-B0. From here, the authors applied their **compound scaling method** to get bigger models by resizing the depth, width, and input resolution of the network.

In this work, we used pre-trained weights of the Imagenet dataset with the Noisy Student [22] approach. Noisy Student is a semi-supervised learning approach proposed by Google's Brain Team. They trained the teacher-student model using labeled and unlabeled data: First, the labeled data was used to train a teacher model. Pseudo labels for the unlabelled data were generated by using the teacher model. These labels can be soft (continues) or hard (one-hot encoding). Then, a student model was trained by using both labeled and pseudo labeled data. They also trained another student model by using the previous student model as a teacher model. The authors call their method as *knowledge expansion* rather than knowledge distillation because they $i$) added noise and $ii$) the student model is not smaller than the teacher. According to the original paper, Noisy Student Training reaches 88.4% top-1 accuracy, which was calculated on popular ImageNet dataset. Figure 2.8 also shows the effect of the Noisy Student training on EfficientNets.

## 2.2 Dataset

### 2.2.1 DFDC

We used The DeepFake Detection Challenge (DFDC) Dataset [23] by Facebook AI. The released part of this dataset was collected by filming 960 paid actors under various conditions, environments, and angles. The age, gender, and ethnicity of actors vary. Train dataset consists of more than 120,000 videos; validation set has 4,000, and test set has 5,000 videos. While the train set is highly unbalanced, test and validation sets are balanced. The majority of fake faces were generated by using Deepfake Autoencoder (DFAE). They used one shared encoded and two decoders for each identity. Unfortunately, all details of the DFAE architecture are not available. Other methods are the frame-based morphable-mask (MM/NN) model, which uses facial landmarks and matches source face to the target face; fine-tuned version of neural talking heads of people (NTH); FSGAN and modified version of StyleGAN. After fake faces were generated, some of them went through a sharpening filter to increase the quality of the result. Audio manipulation was also applied to some of the videos, but we only focused on image manipulations in this work. Various distractors and augmenters were applied to the test and validation sets to make it even more challenging. Applied augmenters were: Gaussian blurring, brightness change, adding contrast, frame rate change, gray-scale, horizontal flip, audio removal, adding noise, changing encoding quality, altering the resolution, and rotating. Distractors were

random shapes, text, or dots on each frame, randomly moving or appearing images or texts. Examples of these augmenters were given in Figure 2.10. All augmenters and distractors were applied randomly applied to 79% of all test datasets.

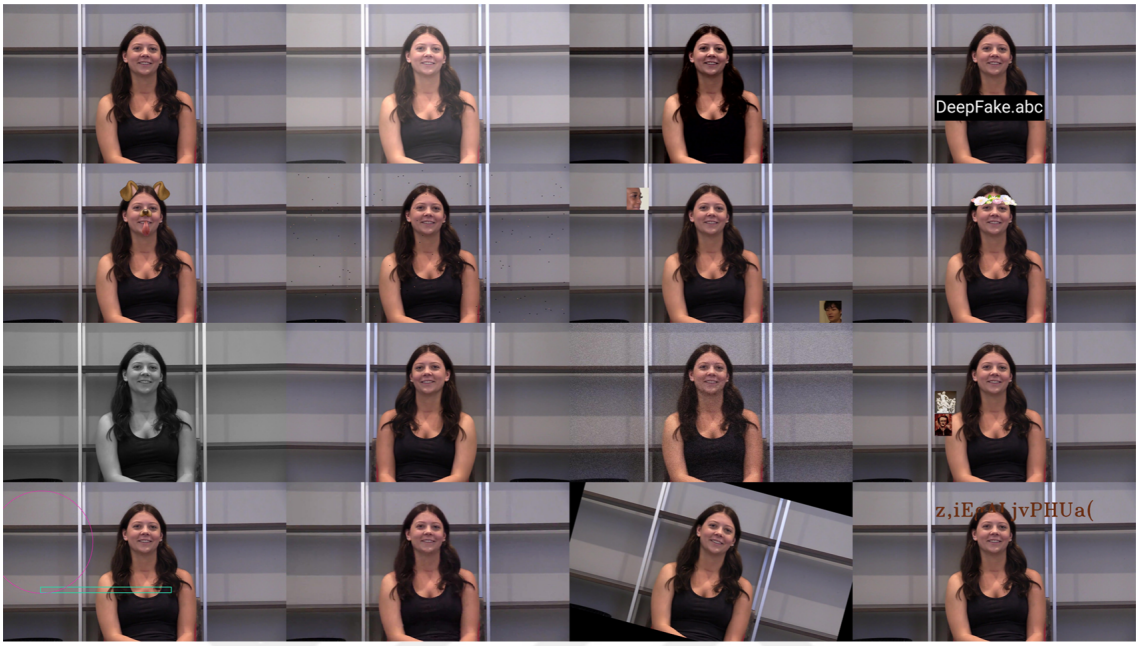


**Figure 2.10** Examples of the augmenters and distractors applied to the DFDC test set. Left top is the original image.[23]

However, this dataset is not perfect and has some problems:

1. Videos with multiple actors - it is not clear if both actors got face manipulation or not. From our observation, only one of them got manipulated. Only video-based labels are present. We assumed that if there is at least one fake face in the video, then the video is fake.
2. The face detection method for the dataset is unknown. There are some cases where manipulation is not applied. This can either be because the face is not detected, or the fake generation method was failed.
3. Test and train sets vary a lot. Distractors and augmenters applied to the test are not present in the train set.
4. Extreme head poses - some faces were turned more than 90 degrees.
5. Too noisy and dark videos.

The last two points cause problems with face detection. It is also hard to understand the person's identity with the naked eye, even if it is a real video. We believe that these cases are against the nature of DeepFakes; thus, we ignored them.

### 2.2.2 Celeb-DF-V2

Celeb-DF-V2 [24] contains 590 real videos of the celebrities obtained from YouTube. The real videos consist of 59 celebrities' interviews. From these videos, 5639 fake videos were generated. This work uses the extended version of the Celeb-DF-v2 dataset, which contains additional 300 real celebrity videos from YouTube. Figure 2.11 shows some real and fake examples from the Celeb-DF-v2 dataset.

**Figure 2.11** Examples from the Celeb-DF-v2 dataset [24]. Left (green) column is real face and other columns (red) are different DeepFake of the real person with various source subjects.

The authors were used an encoder-decoder model with several layers (the exact architecture of the deepfake model was not explained in the paper) to synthesize faces with the size of 256x256. During the face generation time, color augmentation was added to fix the color mismatch between source and target faces. We used all of this dataset as a test set to measure the generalization ability of the models trained by using the DFDC dataset.

## 2.3 Augmentations

All models were trained with 4 different augmentations: no augmentation, default augmentation, default + CoarseDropout, default + Gridmask [25] augmentation. Augmentations were used to increase diversity in the dataset instead of increase the size of the dataset. In order to satisfy the input size of the model, images were resized by preserving the aspect ratio. First, the longest side of an image resized to the desired

**Table 2.2** Order and occurrence probability of the applied augmentations.

| Order | Augmentation | Probability |
|---|---|---|
| 1 | ImageCompression | 0.5 |
| 2 | GaussNoise | 0.1 |
| 3 | GaussianBlur or Sharpening | 0.5 |
| 4 | HorizontalFlip | 0.5 |
| 5 | BrightnessContrast | 0.5 |
| 6 | To gray-scale | 0.2 |
| 7 | ShiftScaleRotate | 0.5 |

size, and then the shortest side resized to such a size that has the same aspect ratio as the original image. The remaining part of the image was padded by using zero values. All images were also normalized by using the mean and standard deviation of the Imagenet dataset. As a default augmentation, we chained and applied several augmentations with different occurrence probability by using Albumentations [26] library. Order and the details of the augmentations were given at table 2.2.

CoarseDropout was inspired from cutout [27] augmentation. It cuts out random parts of an image with fixed size and number. In this work, we chose to use two random patches with the size of 64x64 pixels. These patches may overlap. GridMask [25] augmentation cuts out a fixed number of parts with a certain pattern. We used a 5x5 pattern, size of patches was calculated according to the input size. The purpose of both augmentations was to hide some parts of an image and force models to learn other parts of the image. These augmentations are assumed as harder augmentations and applied separately at the end of the default chain with the probability of 0.5. Examples of all augmentations were given in Figure 2.12.

**Figure 2.12** Augmentations: From top left: Source image, compression, Gauss noise, blur, sharpening, horizontal flip, brightness, hue and saturation value adjustment, gray, shift+scale+rotate, Coarse dropout (size=64 pixels), Grid dropout. All images resized to 224 by preserving aspect ratio and padded with 0 value before an augmentation was applied.

## 2.4 Data preparation

Original datasets consist of high-resolution, high-frame videos. Processing all frames at high resolution is not feasible. Also, a high frame rate means many similar images. Instead, we issued sampling and detected faces. Train and test sets were followed different pipelines.

### 2.4.1 Train data extraction pipeline

The DFDC train dataset consists of 120,000+ videos. Almost 80% of all videos are fake videos. Since all fake videos were created from real videos, in order to speed up processing, we first detected faces at real videos and, by using the coordinates, extracted faces from fake videos. For face detection we used Pytorch implementation of FaceNet [28] model. From each video, up to 10 frames were sampled and passed through face detection. Faces with the size of 70x70 pixels and higher were kept. Faces were extracted with 30 pixels margin around them. A video was discarded if no face was detected. This generally happened when image quality is too bad or the face is barely visible. Next, we manually cleaned the face-only dataset. Here, our target was non-face, false positives. By using the bounding box coordinates of the real faces, we extracted faces from fake videos. Finally, fake images were down-sampled by 5 to keep the train set balanced. DFDC dataset only contains video labels. We propagated video labels to the detected faces from the video. As a result, 202,951 images with real labels and 212,454 images with fake labels left in the train set.

### 2.4.2 Test data extraction pipeline

For test and validation sets, a different flow was followed. First, we detected faces in each frame of a video frame and tracked them using IoU of face bounding boxes. Faces with more than 0.75 IoU in 2 consecutive frames were assumed to be the same person. A person who tracked over 50 frames in the video was included in the sampling set. Next, approximately 50 faces were sampled from each video. Sampling was done proportional to the number of frames tracked per face. This means if person-A tracked for 100 frames and face of person-B tracked for 200 frames in the same video, 17 faces of person-A and 32 faces of person-B were sampled. The same test procedure was applied to the Celeb-DF-v2 dataset. We decided to use this pipeline for the test dataset to avoid distractors and focus only on the main subject of the video. Resizing with preserving the aspect ratio, padding with zero value, and normalization was applied to the test sets.

## 2.5 Training

Due to the size of the dataset, models tend to overfit at early epochs; thus, we trained all models at most ten epochs. Figure 2.13 and 2.14 shows how quickly models converge. The last layer of the backbone models was replaced with a single Dropout layer and a Linear layer with two classes (fake and real). We used pre-trained Imagenet weights and fine-tuned all layers of the model with cross-entropy loss. All models were trained by using Adam optimizer, and the fixed learning rate equals 0.0001.

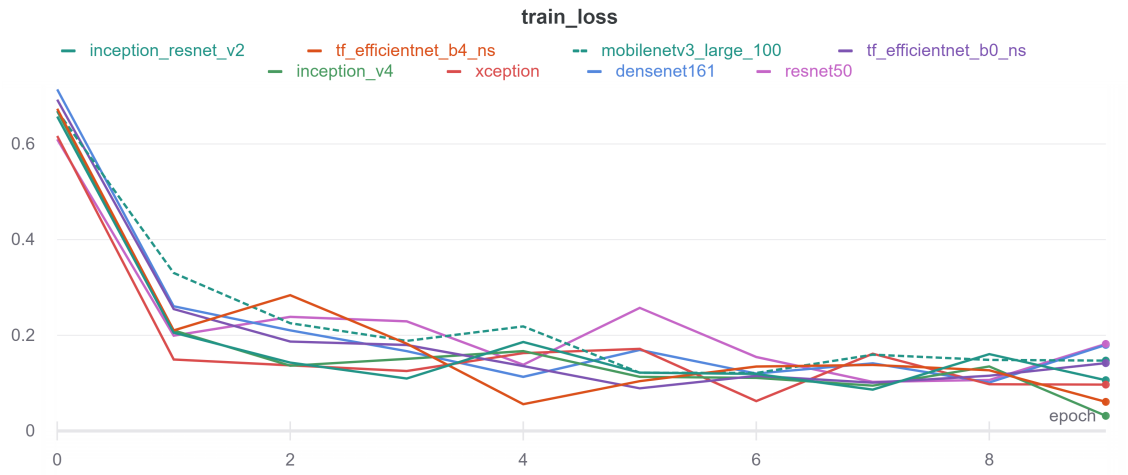


**Figure 2.13** Train loss over epochs of the models trained with fixed Learning rate and default augmentation setup.

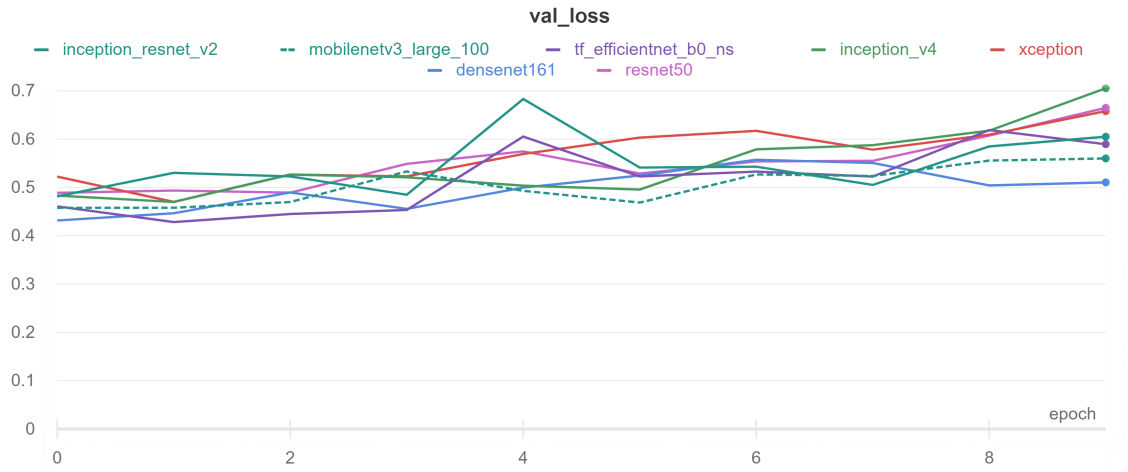


**Figure 2.14** Validation loss over epochs of the models trained with fixed Learning rate and default augmentation setup. (Note, because the EfficientNet-B4 model overfitted so quickly, validation loss became NaN; thus, it was excluded from this graphic

**Table 2.3** Training EfficientNet-B4 model with half-precision and single precision

| | Batch size | Training duration | Log loss |
|---|---|---|---|
| **FP16** | 64 | 9 hours 48 mins | 0.2191 |
| **FP32** | 32 | 5 hours 55 mins | 0.2128 |

### 2.5.1 Half-precision

Nvidia researches showed that training models with half-precision instead of single-precision could achieve similar results while using fewer resources and with shorter training time [29]. The main difference between the two is the number of bits to represent a number. While single-precision uses 8 bits to store the exponent and 23 bits for the precision, half-precision uses 5 bits for the exponent and 10 bits for the fraction. Due to this difference, half-precision has a small representable range and is less precise but enough to train most deep learning models. Considering millions of trainable parameters, this difference leads to several times faster and memory-efficient models. In this work, we also compared FP-16 and FP-32 performance on the Efficient-B4 model.

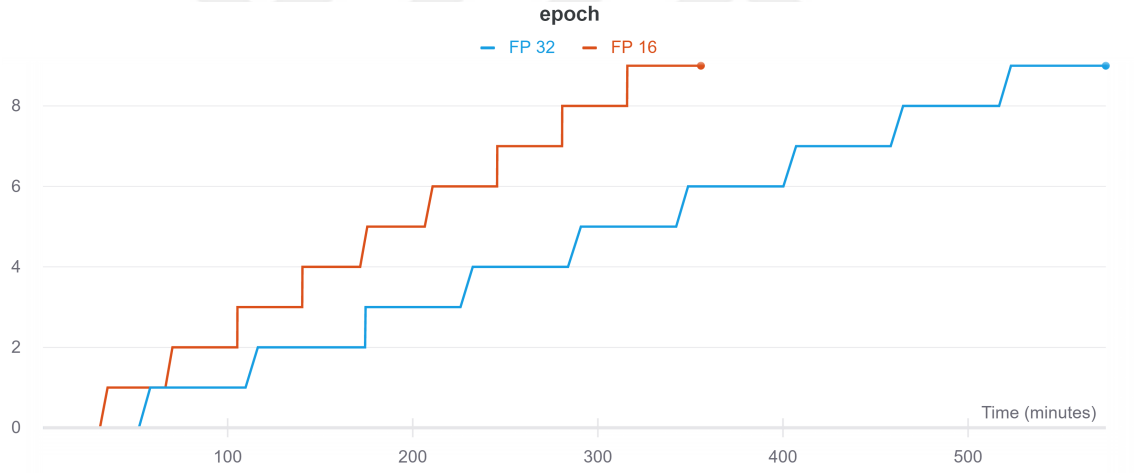


**Figure 2.15** Training time per epoch of EfficientNet-B4 with FP16 and FP32

The training time of single-precision is significantly high, while an improvement on log loss is less than 0.01. Results and difference of the setups can be found at Table 2.3. Time spent for each epoch during training was given in Figure 2.15. Because of these reasons, we decided to train all models with half-precision.

# 3
# RESULTS AND DISCUSSION

In this work, we trained and compared several performance of models explained at Section 2.1 on popular deepfake dataset - DeepFake Detection Challenge Dataset by Facebook AI. Because the dataset only has video based results, we chose video based log loss as a main metric. Log loss formula was given at 3.1. The purpose of using log loss is to "punish" model more if a video is classified false with higher confidence.

$$LogLoss = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\widehat{y}_i) + (1-y_i) log(1-\widehat{y}_i)] \tag{3.1}$$

Sampled frames from each video were passed through a model, and confidence for real and fake classes was predicted. Confidence of the fake class averaged, and a single confidence value was created for the input video. This value was used as $\widehat{y}$ to the formula. For the log loss calculation, 1 was used for the fake and 0 for the real class. If averaged confidence is higher than a certain threshold, the video was considered fake and vice versa.

The threshold was calculated for each model and configuration separately instead of using a fixed one. In order to do this, we first calculated false positive rates and false negative rates at different thresholds. The intersection of this rates was called Equal Error Rate, where both rates are low. The threshold at this point was picked and used. Example graphic was given in Figure 3.1. Figure 3.2 shows the chosen threshold values for each model.

Later, we compared models' generalization ability by testing them on the Celeb-DF-v2 dataset. Because this dataset is very unbalanced, we reported recall for each class separately.

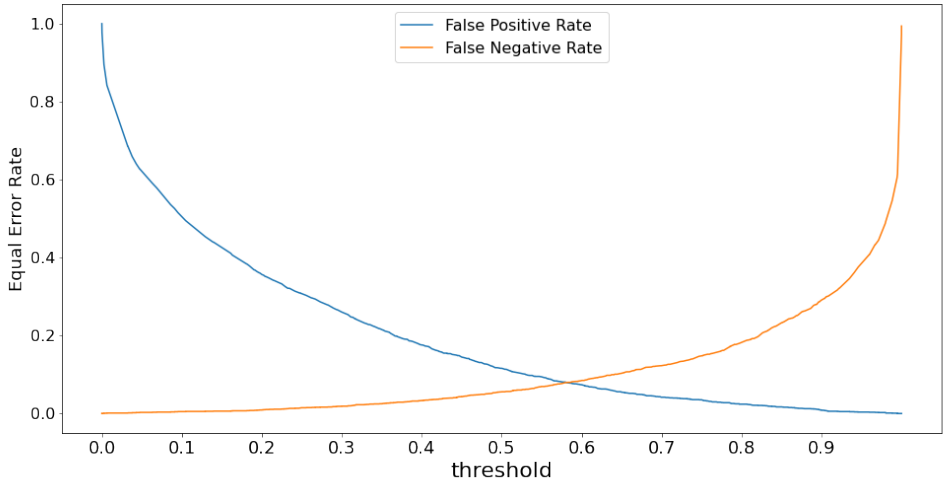


**Figure 3.1** Equal Error Rate for the EfficientNet-B4 model trained with Default augmentation.

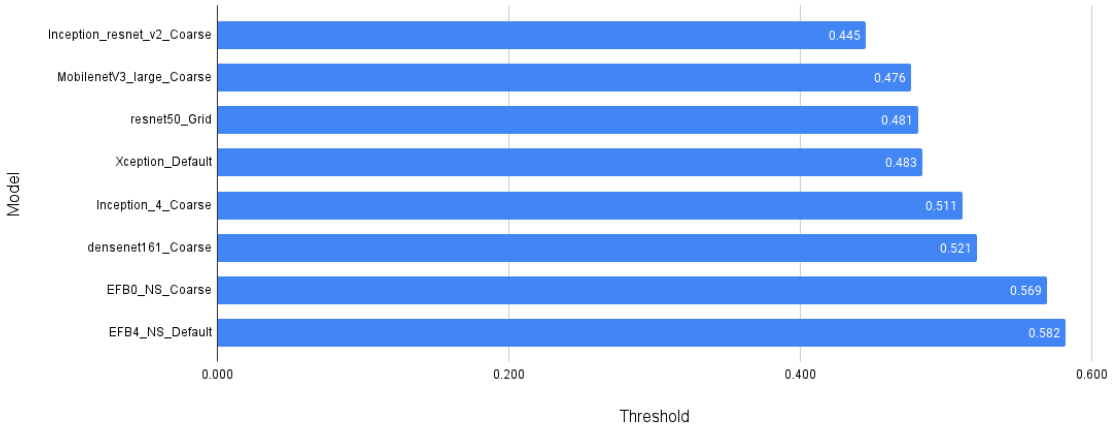


**Figure 3.2** Calculated threshold values for the best model/augmentation combinations.

## 3.1 DFDC results

Because deepfake models were already modified faces, we started experiments by training our baseline models with no extra augmentation except resizing with aspect ratio, zero-padding and normalizing. Video-based log loss results on DFDC test dataset were given at Figure 3.3. As seen from the figure, the results are not good. Despite the no augmentation, the EfficientNet-B4 model had a considerably lower loss.

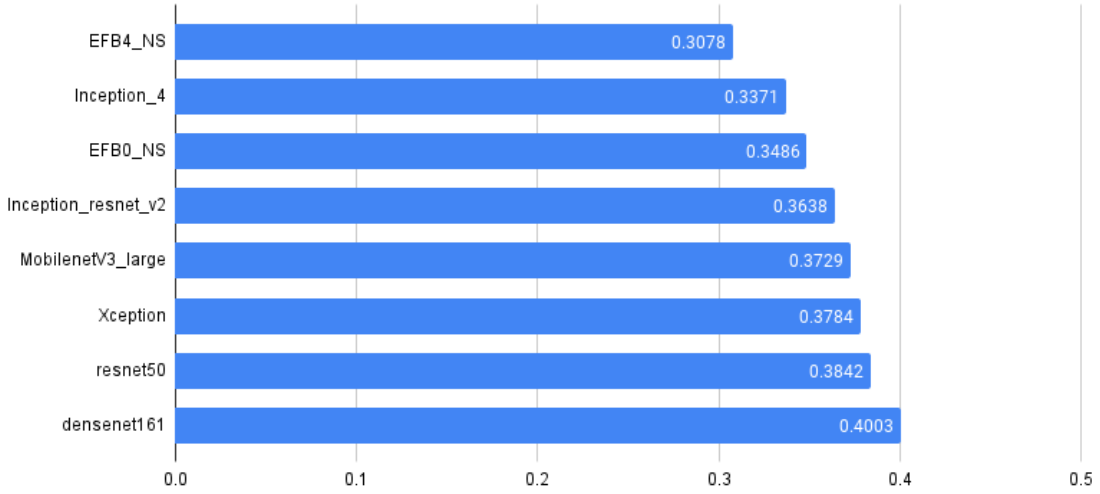


**Figure 3.3** Video based log loss of the trained models without any augmentation (lower is better).

Next, after evaluating Kaggle solutions and studying the test set, we decided to add several augmentations during training time. The details of the default augmentation were discussed at Section 2.3. The main purpose was to create diversity in the dataset and overcome distractors of the test set. Thus, instead of increasing the size of the dataset by adding new augmented images, we augmented only existing images. As seen from Figure 3.4, augmentations helped to improve results drastically. EfficientNet-B4 is the best model here.

Next, we tried hard augmentations together with the default chain. We added a 5x5 grid mask with zero values on the input images. The purpose was forcing models to learn less significant but still important parts of the data. Log-loss results for each model trained with this setup was given at Figure 3.5. While results were much better than no augmentation results, improvement was not much from only the default augmentation setup. Adding GridMask improved the results of the ResNet50 model. Just adding it decreased video-based log loss 25%.

Another hard augmentation we tried was Coarse dropout: 2 patches, each size of 64x64 randomly cropped out from an input image with an occurrence probability of 50%. As seen from the results 3.6 most of the models had improvements here.

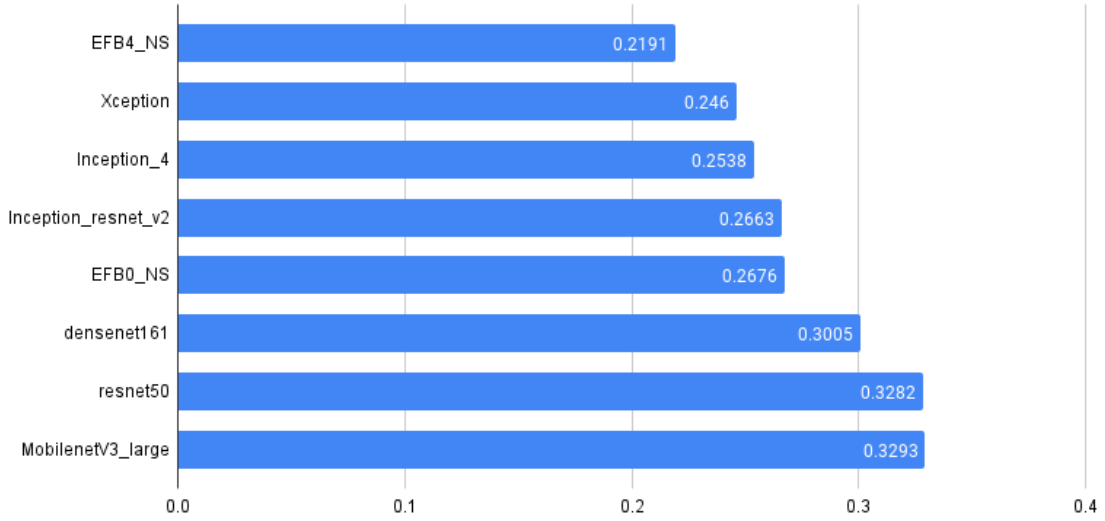


**Figure 3.4** Video based log loss of the trained models with default augmentation chain (lower is better).

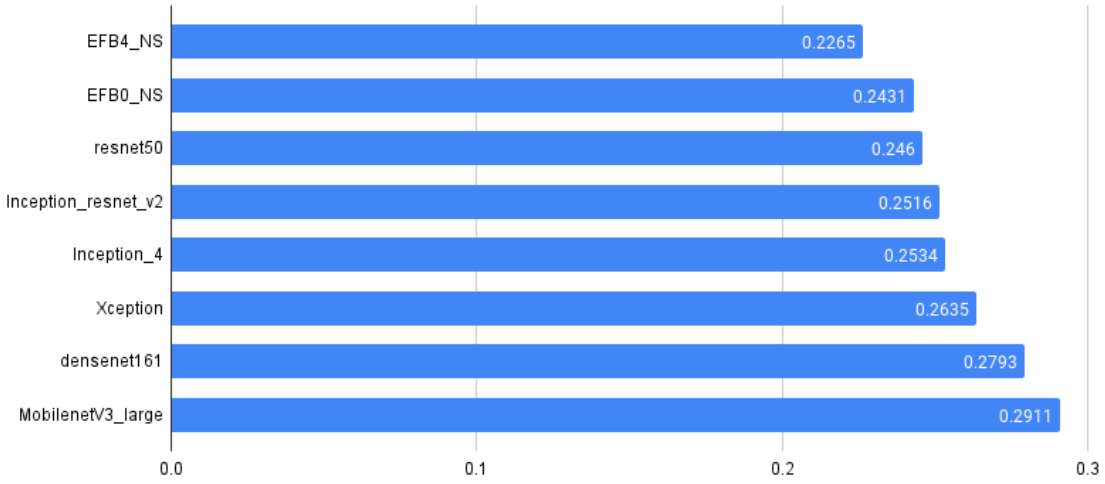


**Figure 3.5** Video based log loss of the trained models with default augmentation chain and grid dropout (lower is better).

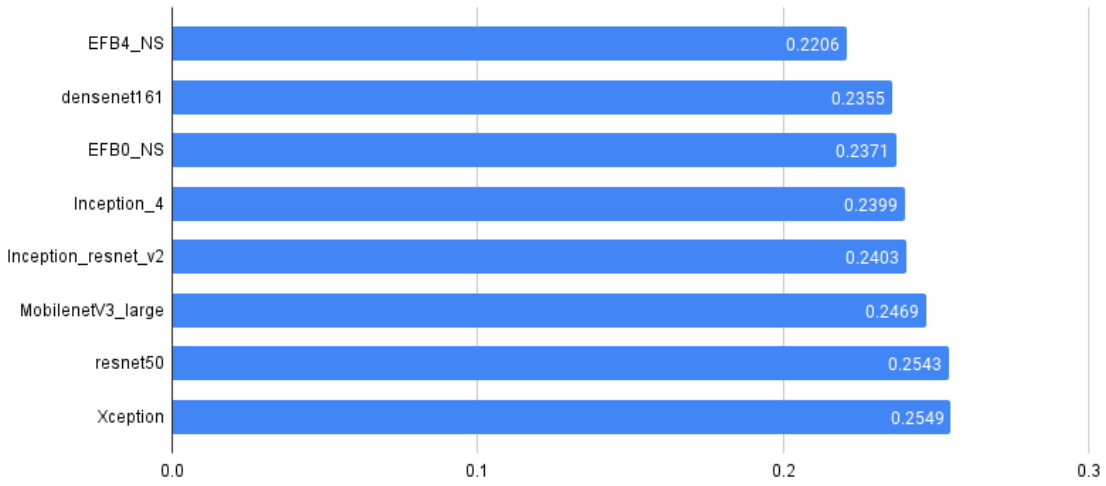


**Figure 3.6** Video based log loss of the trained models with default augmentation chain and coarse dropout (lower is better).

**Table 3.1** Log loss results of the DFDC dataset (lower is better). *Italic* values are the best result for the augmentation and **bold** values are the best result of the model

| Models/Augmentations | No Augm. | Default | Coarse Dropout | Grid Mask |
|---|---|---|---|---|
| **EFB4_NS** | *0.3078* | ***0.2191*** | *0.2206* | *0.2265* |
| **Xception** | 0.3784 | **0.2460** | 0.2549 | 0.2635 |
| **Inception_4** | 0.3371 | 0.2538 | **0.2399** | 0.2534 |
| **Inception_resnet_v2** | 0.3638 | 0.2663 | **0.2403** | 0.2516 |
| **EFB0_NS** | 0.3486 | 0.2676 | **0.2371** | 0.2431 |
| **densenet161** | 0.4003 | 0.3005 | **0.2355** | 0.2793 |
| **resnet50** | 0.3842 | 0.3282 | 0.2543 | **0.2460** |
| **MobilenetV3_large** | 0.3729 | 0.3293 | **0.2469** | 0.2911 |

Overall, the EfficientNet-B4 model pretrained with Noisy Student and used default augmentation chain had the best result. Comparison of the video-based log-loss values for each model and augmentation setup were given at Table 3.1 and RoC-AuC curve of the best model and augmentation combinations were given in Figure 3.7.

### 3.1.1 Distractor and augmenter analysis

The test set of the DFDC dataset has distractors and augmenters, which were not applied to the train set. Augmenters were considered as color and geometric transformations, frame rate changes, etc. Distractors were objects such as dots, random shapes, figures, texts, random face images overlayed on top of the video frame. Two popular social media filters: dog filter and flower crown filter, were also used as an augmenter. We analyzed the effect of the distractors and the augmenters separately for fake videos and real videos. Only the best configuration of each model

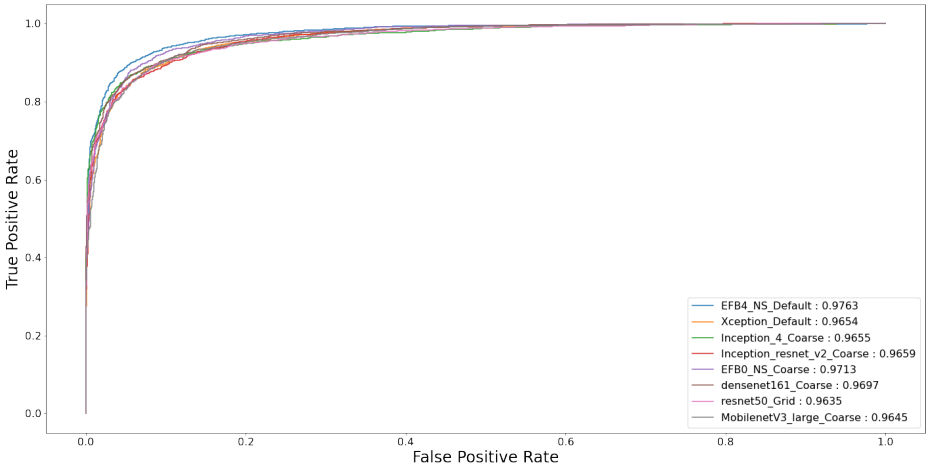


**Figure 3.7** RoC-AuC curves for the best configuration of each model.

(highlighted as **bold** in Table 3.1) were analyzed. Table 3.2 shows percentage of the misclassified fake videos per filter and model. As seen from the values, fake videos with dog filters were misclassified as real videos. Interestingly, the dog filter does not have a bad effect on real videos. From Table 3.3, we can see that other random objects and shapes have a higher impact than the dog filter. This may indicate that models tend to classify images with this filter as real. There is one extra distractor that is only applied to the real videos called df_tricks. Unfortunately, the original paper does not give any information about this distractor. The amount and strength of each augmenter vary.

When we look at the augmenters - Table 3.4 - we can see that noise has the highest impact on all models, especially for fake videos. All models had a hard time distinguishing fake videos with noise from real videos. We applied Gaussian Noisy augmentation with very low probability. Increasing the occurrence probability of the noisy augmentation may solve this problem. For real videos, there is not any significant augmenter that affected all models. Some of the videos in the DFDC test set has neither augmenters nor distractor. Percentage of the misclassified videos with no artifact was given at the last row of the Table 3.4 and Table 3.5.

**Table 3.2** Percentage of misclassified fake videos per distractor for each model's best setup. **Bold** values are the worst results for each model.

| | EFB4 | EFB0 | Xception | InceptionV4 | IncpResV2 | Resnet50 | Densenet161 | MobileNetV3 |
|---|---|---|---|---|---|---|---|---|
| **dog_filter** | **22.1** | **22.1** | **51.2** | **41.9** | **50.0** | **37.2** | **40.7** | **25.6** |
| **dots** | 14.9 | 10.7 | 10.7 | 13.2 | 14.0 | 13.2 | 14.0 | 12.4 |
| **faces** | 8.0 | 9.6 | 6.4 | 12.8 | 8.8 | 9.6 | 10.4 | 12.8 |
| **flower_filter** | 10.0 | 8.3 | 10.0 | 10.8 | 10.0 | 8.3 | 10.0 | 8.3 |
| **random images** | 13.5 | 13.5 | 13.5 | 12.6 | 10.8 | 13.5 | 13.5 | 14.4 |
| **shapes** | 6.4 | 8.3 | 2.8 | 7.3 | 6.4 | 6.4 | 8.3 | 10.1 |
| **text** | 8.9 | 11.1 | 8.9 | 6.7 | 13.3 | 10.0 | 12.2 | 14.4 |



**Table 3.3** Percentage of misclassified real videos per distractor for each model's best setup. **Bold** values are the worst results for each model.

| | EFB4 | EFB0 | Xception | InceptionV4 | IncpResV2 | Resnet50 | Densenet161 | MobileNetV3 |
|---|---|---|---|---|---|---|---|---|
| **df_tricks** | 7.9 | 10.1 | 9.0 | 6.7 | 11.2 | 9.0 | 7.9 | 6.7 |
| **dog_filter** | 6.3 | 10.1 | 7.6 | 8.9 | 15.2 | 10.1 | 7.6 | 10.1 |
| **dots** | 8.9 | 12.9 | 13.9 | 13.9 | **15.8** | 12.9 | 12.9 | 13.9 |
| **faces** | **9.5** | 8.3 | 10.7 | 10.7 | 10.7 | 9.5 | 8.3 | 11.9 |
| **flower_filter** | 2.4 | 4.7 | 8.2 | 3.5 | 5.9 | 7.1 | 8.2 | 4.7 |
| **random images** | 8.6 | 8.6 | **16.1** | 11.8 | 14.0 | 10.8 | **14.0** | 11.8 |
| **shapes** | 7.8 | **13.0** | 13.0 | **14.3** | 14.3 | **13.0** | 13.0 | 14.3 |
| **text** | 8.9 | 11.1 | 8.9 | 6.7 | 13.3 | 10.0 | 12.2 | **14.4** |

**Table 3.4** Percentage of the misclassified fake videos for each model's best configuration per augmenter. **Bold** values are the worst result for each model.

| | **EFB4** | **EFB0** | **Xception** | **InceptionV4** | **IncpResV2** | **Resnet50** | **Densenet161** | **MobileNetV3** |
|---|---|---|---|---|---|---|---|---|
| **blur** | 4.1 | 4.1 | 2.4 | 3.6 | 0.0 | 1.2 | 3.0 | 1.8 |
| **brightness** | 7.2 | 10.8 | 7.2 | 7.8 | 5.4 | 8.4 | 8.4 | 9.6 |
| **contrast** | 13.2 | 16.9 | 14.7 | 16.9 | 21.3 | 17.6 | 18.4 | 16.9 |
| **framerate change** | 3.6 | 6.0 | 7.8 | 3.6 | 6.6 | 4.8 | 9.0 | 4.2 |
| **greyscale** | 10.6 | 7.0 | 12.0 | 5.6 | 8.5 | 12.0 | 16.2 | 10.6 |
| **horizontal flip** | 7.7 | 6.5 | 7.7 | 7.1 | 5.2 | 7.1 | 7.7 | 6.5 |
| **no_audio** | 5.4 | 5.4 | 9.4 | 6.7 | 8.1 | 4.7 | 6.0 | 5.4 |
| **noise** | **30.3** | **35.9** | **22.8** | **53.8** | **37.9** | **49.7** | **22.8** | **50.3** |
| **quality level change** | 12.5 | 14.5 | 9.9 | 11.8 | 10.5 | 9.2 | 15.8 | 7.2 |
| **resolution change** | 7.0 | 3.2 | 3.2 | 4.4 | 2.5 | 2.5 | 3.2 | 1.3 |
| **rotate** | 4.0 | 5.3 | 6.0 | 3.3 | 4.0 | 6.6 | 4.0 | 3.3 |
| **no distractor/augmenter** | 6.2 | 7.4 | 8.1 | 4.8 | 5.6 | 5.6 | 6.2 | 6.6 |

**Table 3.5** Percentage of the misclassified real videos for each model's best configuration per augmenter. Bold values are the worst result for each model.

| | **EFB4** | **EFB0** | **Xception** | **InceptionV4** | **IncpResV2** | **Resnet50** | **Densenet161** | **MobileNetV3** |
|---|---|---|---|---|---|---|---|---|
| **blur** | 7.5 | 6.2 | 8.9 | 10.3 | 11.0 | 8.9 | 10.3 | 13.0 |
| **brightness** | 7.0 | 8.3 | 14.0 | 10.8 | 13.4 | 9.6 | 8.9 | 8.3 |
| **contrast** | 5.6 | 3.5 | 4.9 | 7.7 | 14.0 | 5.6 | 9.1 | 8.4 |
| **framerate change** | **9.6** | **12.2** | 11.5 | 9.6 | 14.1 | 11.5 | 10.3 | **14.1** |
| **greyscale** | 7.1 | 7.1 | 9.6 | 8.3 | 10.3 | 12.8 | **12.2** | 7.1 |
| **horizontal flip** | 9.3 | 8.6 | 9.9 | 10.6 | 14.6 | 9.3 | 7.9 | 9.3 |
| **no_audio** | 5.9 | 4.6 | 7.9 | 7.2 | 11.2 | 7.9 | 3.3 | 8.6 |
| **noise** | 5.6 | 8.3 | 13.2 | 11.1 | **15.3** | 10.4 | 11.8 | 11.1 |
| **quality level change** | 5.2 | 6.4 | 11.0 | 8.7 | 9.3 | 8.1 | 8.7 | 9.3 |
| **resolution change** | 8.6 | 9.6 | **14.4** | **13.4** | 12.8 | **13.4** | 10.7 | 11.2 |
| **rotate** | 6.4 | 7.6 | 8.8 | 7.0 | 11.1 | 7.6 | 7.0 | 10.5 |
| **no distractor/augmenter** | 6.4 | 9.2 | 11.3 | 12.2 | 12.0 | 11.3 | 9.8 | 10.3 |

**Table 3.6** Sensitivity and Specificity results of Celeb-DF-v2 dataset for each model and augmentation (higher is better).

| | **No Augmentation** | | **Default setup** | | **Default + Coarse** | | **Default + GridDropout** | |
|---|---|---|---|---|---|---|---|---|
| | *Sensitivity* | *Specificity* | *Sensitivity* | *Specificity* | *Sensitivity* | *Specificity* | *Sensitivity* | *Specificity* |
| **densenet161** | 0.793 | 0.795 | 0.831 | 0.832 | 0.854 | 0.856 | 0.836 | 0.841 |
| **EfficientNet-B0** | 0.781 | 0.783 | 0.865 | 0.87 | 0.885 | 0.886 | 0.877 | 0.88 |
| **EfficientNet-B4** | 0.875 | 0.876 | 0.92 | 0.921 | 0.924 | 0.925 | 0.921 | 0.922 |
| **Inception_4** | 0.785 | 0.786 | 0.873 | 0.875 | 0.874 | 0.874 | 0.867 | 0.873 |
| **Inception_resnet_v2** | 0.783 | 0.791 | 0.867 | 0.872 | 0.872 | 0.873 | 0.845 | 0.848 |
| **MobilenetV3_large** | 0.77 | 0.773 | 0.842 | 0.846 | 0.874 | 0.876 | 0.866 | 0.868 |
| **resnet50** | 0.753 | 0.756 | 0.812 | 0.814 | 0.852 | 0.858 | 0.853 | 0.855 |
| **Xception** | 0.777 | 0.782 | 0.865 | 0.868 | 0.864 | 0.866 | 0.858 | 0.862 |

**Figure 3.8** Fake face examples from DFDC test set classified as real

**Figure 3.9** Real face examples from DFDC test set classified as fake

Examples of the fake faces classified as real were given in Figure 3.8. The test sets were not manually cleaned to match real-life scenarios as much as possible. Because of this, it is possible to see false face detection examples. Also, some faces were covered by hand, which probably prevented deepfake from being applied properly. Out of all problems, extreme head poses are the most significant and hard ones. Figure 3.9 shows misclassified real face examples.

**Table 3.7** F1-score of the models tested with Celeb-DF-v2. **Bold** values are the best results for the given model and italic values are the best results for an augmentation.

| Model/Augmentation | No Augmentation | Default | Coarse | Grid |
|---|---|---|---|---|
| densenet161 | 0.869 | 0.895 | **0.91** | 0.898 |
| EFB0_NS | 0.86 | 0.917 | **0.93** | 0.925 |
| EFB4_NS | *0.924* | *0.952* | ***0.954*** | *0.953* |
| Inception_4 | 0.863 | 0.922 | **0.923** | 0.919 |
| Inception_resnet_v2 | 0.862 | 0.919 | **0.922** | 0.904 |
| MobilenetV3_large | 0.853 | 0.902 | **0.923** | 0.918 |
| resnet50 | 0.841 | 0.882 | **0.909** | 0.909 |
| Xception | 0.858 | **0.918** | 0.917 | 0.913 |

## 3.2 Celeb-DF-v2 results

We also tested the generalization ability of the models by testing them with the Celeb-DF-v2 dataset. Since this dataset was only used for the testing purpose instead of the default train/test split, we used the whole dataset as a test set. Because of this, the dataset is very unbalanced. Thus, separately looking at results for each class will be more healthy. At Table 3.6 we compared 2 values for each model and configuration: *Sensitivity* - represents the amount of correctly classified fake videos in other words, coverage of actual positives formula 3.2. *Specificity* - is the amount of correctly classified real videos dataset or coverage of actual negatives formula and 3.3. When we analyze results, we can see a similar pattern with DFDC test results: No augmentation has the worst results; adding augmentation and enforcing it with harder augmentations increased results. One interesting detail is that Coarse Dropout has the best results overall for most of the models. This indicates that adding this augmentation helps models to generalize better. This is mostly because it forces models to learn less significant parts, too, and does not follow any pattern.

$$Sensitivity = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{3.2}$$

$$Specificity = \frac{True\ Negative}{True\ Negative + False\ Positive} \tag{3.3}$$

F1-score for each model-augmentation combination was reported at Table 3.7 and F1-score comparision of the each model's best augmentation configuration was given in Figure 3.10. Fake class was assumed as the true class since our main target is detecting fake videos. Examples of misclassified fake and real faces were given in Figure 3.11 and 3.12.

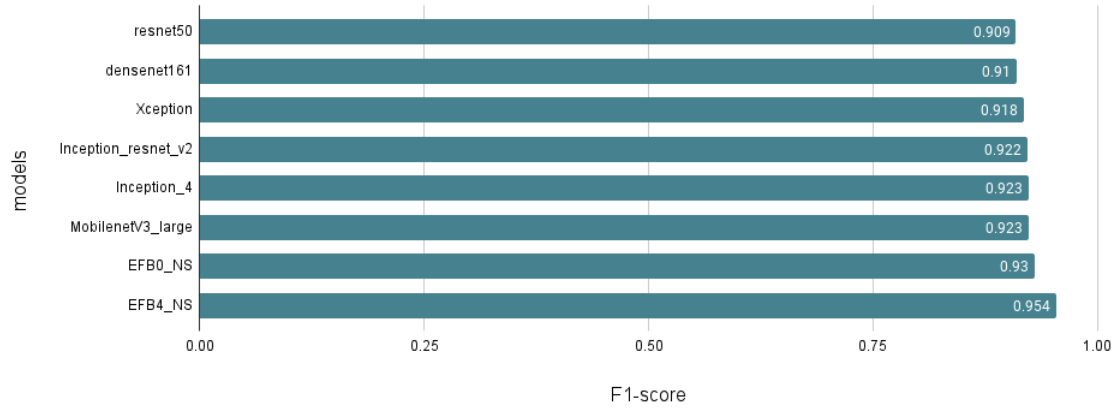


**Figure 3.10** F1-score for the best configuration for each model. Except for the Xception model, all models have the best score with default + Coarse augmentation. Xception model has the best result with only default augmentation.

**Figure 3.11** Examples of misclassified fake faces from Celeb-DF-v2 dataset

**Figure 3.12** Examples of misclassified real faces from Celeb-DF-v2 dataset

## 3.3 Discussion

As AI technologies improve, misuse of them increases as well. One of these threats to society is DeepFakes. As more challenging and sophisticated fake face generators emerge every day, detecting them becomes harder and harder. In this work, we trained eight different models - Xception, Inception-v4, Inception-ResNet-v2, ResNet50, DenseNet161, MobilenetV3, EfficientNet-B0 and EfficientNet-B4 - with various augmentation setups on the recent DFDC dataset and compared their generalization ability by testing on the Celeb-DF-v2 dataset. Due to hardware limitations, we could not train bigger EfficientNet models and emerging Vision Transformers as well as other bigger models. We concluded that, despite the smaller size, the Efficient-B4 model has the best learning ability among all trained models. The use of the augmentations, not to increase the size of the dataset but to increase diversity in the dataset, can still be helpful. Finally, adding dropout augmentations can help models focus less significant but important parts of the face and generalize better.

This project can be split into three main phases: Data preparation, model training, and combining results.

- **Data preparation** is the key of all modern Deep Learning projects. In our case, processing raw videos are resource-intensive and unnecessary since most frames

look similar (except video-based and 3D convolutional models where multiple consecutive frames have to pass through a model). Also, looking at the whole frame is not very useful since the modification is only applied to the face(s). Thus, a *region of interest* should be defined. We defined this as face and 30-pixel margin around the detected face and sampled faces every 30 frames for the training set. An extensive analysis can be done in future works to find out optimal sampling rate, face detection model, and other input parameters such as input and margin size.

- **Model training** is the biggest part of the whole flow. Deciding learning rate scheduler, optimizer, augmentations applied to the input, and model backbone are crucial parts of training a successful model. For example, the test set of the DFDC dataset contains augmenters and distractors that were not applied to the train set. Using various filters to overcome this feature can be one improvement. In addition, neural architecture search techniques and hyper-parameter tuning methods can be applied to find an optimal model.

- **Combining results** is the final part of the deep fake detection flow. Because sampling is required at the first step, results should be brought together for a single output. In this work, we averaged the confidence of the fake class for each frame and used Equal Error Rate to find an optimal threshold for each model. Different fixed threshold values could also be tested. Also, several frames with higher confidence can be used to create more sophisticated averaging techniques. Ensembling several deep learning models gains popularity as computers get more powerful and handle several of them. $a$) Several different model architectures, $b$) Same model with different input sizes and slight changes, $c$) Models from the same family, for example, EfficientNets, DenseNets, ResNets, etc. can be ensembled. The easiest way of ensembling is training and inferencing each model separately and joining only their output. Training several models simultaneously may also be possible but requires more powerful hardware and complex model design.

Finally, all models and setup configurations can be tested with more test sets to further check generalization ability. It would be interesting to see the performance of the models trained with different datasets.

# REFERENCES


[1] R. Tolosana, R. Vera-Rodriguez, J. Fierrez, A. Morales, J. Ortega-Garcia, "Deepfakes and beyond: A survey of face manipulation and fake detection," *arXiv preprint arXiv:2001.00179*, 2020.

[2] N. Bonettini, E. D. Cannas, S. Mandelli, L. Bondi, P. Bestagini, S. Tubaro, "Video face manipulation detection through ensemble of cnns," *arXiv preprint arXiv:2004.07676*, 2020.

[3] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, M. Nießner, "Faceforensics++: Learning to detect manipulated facial images," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1–11.

[4] *Deepfake detection challenge*. [Online]. Available: `https://www.kaggle.com/c/deepfake-detection-challenge/overview` (visited on 12/15/2020).

[5] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, A. A. Efros, "Cnn-generated images are surprisingly easy to spot... for now," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 7, 2020.

[6] T. Karras, T. Aila, S. Laine, J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[7] J. C. Neves, R. Tolosana, R. Vera-Rodriguez, V. Lopes, H. Proença, J. Fierrez, "Ganprintr: Improved fakes and evaluation of the state of the art in face manipulation detection," *arXiv preprint arXiv:1911.05351*, 2019.

[8] T. T. Nguyen, C. M. Nguyen, D. T. Nguyen, D. T. Nguyen, S. Nahavandi, "Deep learning for deepfakes creation and detection," *arXiv preprint arXiv:1909.11573*, vol. 1, 2019.

[9] R. Tolosana, S. Romero-Tapiador, J. Fierrez, R. Vera-Rodriguez, "Deepfakes evolution: Analysis of facial regions and fake detection performance," *arXiv preprint arXiv:2004.07532*, 2020.

[10] B. Dolhansky, R. Howes, B. Pflaum, N. Baram, C. C. Ferrer, "The deepfake detection challenge (dfdc) preview dataset," *arXiv preprint arXiv:1910.08854*, 2019.

[11] T. Karras, S. Laine, T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401–4410.

[12] Y. Nirkin, Y. Keller, T. Hassner, "Fsgan: Subject agnostic face swapping and reenactment," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7184–7193.

[13] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[14] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1314–1324.

[15] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[16] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[18] C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[19] M. Tan, Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, PMLR, 2019, pp. 6105–6114.

[20] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[21] M. Tan, Q. V. Le, *Efficientnet: Improving accuracy and efficiency through automl and model scaling*, 2021. [Online]. Available: `https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html`.

[22] Q. Xie, M.-T. Luong, E. Hovy, Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 687–10 698.

[23] B. Dolhansky, J. Bitton, B. Pflaum, J. Lu, R. Howes, M. Wang, C. C. Ferrer, "The deepfake detection challenge dataset," *arXiv preprint arXiv:2006.07397*, 2020.

[24] Y. Li, X. Yang, P. Sun, H. Qi, S. Lyu, "Celeb-df: A large-scale challenging dataset for deepfake forensics," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3207–3216.

[25] P. Chen, S. Liu, H. Zhao, J. Jia, "Gridmask data augmentation," *arXiv preprint arXiv:2001.04086*, 2020.

[26] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, A. A. Kalinin, "Albumentations: Fast and flexible image augmentations," *Information*, vol. 11, no. 2, 2020, ISSN: 2078-2489. DOI: 10.3390/info11020125. [Online]. Available: `https://www.mdpi.com/2078-2489/11/2/125`.

[27] T. DeVries, G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[28] F. Schroff, D. Kalenichenko, J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[29] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, *et al.*, "Mixed precision training," *arXiv preprint arXiv:1710.03740*, 2017.

[30] R. Wightman, *Pytorch image models*, `https://github.com/rwightman/pytorch-image-models`, 2019. DOI: `10.5281/zenodo.4414861`.

[31] L. Biewald, *Experiment tracking with weights and biases*, Software available from wandb.com, 2020. [Online]. Available: `https://www.wandb.com/`.

# A Environment setup

## A.1 Hardware

- RAM: 64 GB
- CPU: Intel i7-8700K @ 3.7GHz x 12
- GPU: Nvidia Titan V with 12 GB VRAM

## A.2 Software

- Operating System: Ubuntu 16.04 LTS
- Programming language: Python 3.7
- CUDA Version: 9.0.176
- CuDNN version: 7.3.1
- Libraries and Frameworks with versions
  - pytorch-lightning == 1.2.6
  - facenet-pytorch == 2.5.1
  - albumentations == 0.5.2 [26]
  - timm == 0.4.9 [30]
  - wandb == 0.10.23 [31]
  - opencv-python == 4.5.1
  - numpy == 1.20.1
  - pandas == 1.2.3
  - scikit-learn == 0.24.1
  - matplotlib == 3.3.4

# PUBLICATIONS FROM THE THESIS

## Conference Papers

1. İ. Hüseynli, S. Varlı, "Analyzing Deep Learning Models' Generalization Ability under Different Augmentations on Deepfake Datasets" in *Proceedings of the International Conference on Artificial Intelligence and Soft Computing*, pp. 1-5, Jun 2021