國立清華大學資訊工程系 113 學年度下學期專題報告

專題名稱	Image Forgery Detection using EfficientNets and Multi-attentional Methods at different levels of JPEG Lossy Compression.				
參加競賽或計畫	☐ 參加對外競賽	☐ 參與其他計畫	☐無參加對外競賽或任何計畫		
學號	108006205				
姓名	葛洋晴				

摘要

With the proliferation of the Internet and social media, the dissemination of information has become effortless, leading to increased risks of deepfake images and videos. These manipulations pose significant threats to privacy, identity, and trust. To address these challenges, this project focuses on enhancing image forgery detection using advanced neural network architectures and compression techniques.

The study employs EfficientNet models, specifically leveraging EfficientNet-B4, due to its superior performance in accurately distinguishing between real and manipulated images across various compression levels. By incorporating multi-attentional mechanisms, the model's robustness is further enhanced, allowing it to focus on relevant features even under high compression scenarios.

A critical component of this research is the use of MozJPEG, an advanced JPEG encoder developed by Mozilla. MozJPEG incorporates progressive encoding, trellis quantization, and optimized quantization tables, significantly improving the quality-to-filesize ratio. Additionally, SmartScale and DCT scaling techniques are employed to further enhance compression efficiency while maintaining high visual quality. These methods ensure that the model operates effectively in real-world conditions where images undergo various degrees of lossy compression.

中華民國　　113 年　　5 月

I. Contents

The structure of this report is organized as follows:

1. RESEARCH MOTIVATION AND OBJECTIVES

The rise of the Internet and social media, combined with the widespread use of smartphones, has made it easier than ever for people to share and access information. Advances in technology, particularly in deep learning, have enabled the creation of highly realistic images and videos that are difficult to authenticate. Among these technological tools, deepfakes, which allow for sophisticated face-swapping, have gained prominence. Unfortunately, this technology can be misused to spread false information, leading to significant societal issues such as privacy violations, identity theft, and a general erosion of trust. The pervasive nature of social media exacerbates these problems by facilitating the rapid spread of misinformation.

While many methods have been proposed for detecting image forgeries, few address the challenges posed by compressed images or videos. The compression process often introduces noise, which can reduce the effectiveness of forgery detection techniques. This is particularly concerning given the vast amount of digital content shared on social networks, where compression is commonly used.

My study focuses on improving the robustness and accuracy of forgery detection models, particularly under varying levels of JPEG compression (quality levels of 77, 60, 15 and 10), which is commonly used in digital media. Initially, I employed the Meso4 (Afchar et al., 2018) model for forgery detection due to its simplicity and decent performance. However, as my research progressed, I observed that Meso4's accuracy was insufficient for more challenging detection tasks. Therefore, in this report, I will explore the use of EfficientNet (Tan & Le, 2021), a state-of-the-art convolutional neural network, known for its efficiency and accuracy, as our primary detection model. Additionally, I will incorporate multi-attentional mechanisms as the ones use in the work of, (Zhao et al., 2021), to enhance the model's performance by allowing it to focus on relevant features across different levels of compression.

Through this comprehensive study, I aim to contribute to the field of digital computer vision by providing a more robust and accurate method for detecting image forgeries, even under challenging conditions such as lossy compression.

2. CURRENT STATE OF RELATED RESEARCH AND COMPARISONS

2.1 *Meso4 Network*

The Meso4 network is a convolutional neural network (CNN) designed for binary classification tasks, particularly for detecting deepfake images. Its architecture includes convolutional layers with batch normalization and LeakyReLU activation functions. LeakyReLU mitigates the vanishing gradient problem by maintaining a small, non-zero gradient when the unit is inactive, ensuring consistent gradient flow during training and allowing the model to learn complex patterns effectively.

Despite its advantages, Meso4 has several limitations. The network struggles with generalization, especially under different compression levels and varying forgery types. It exhibits poor performance in distinguishing between real and fake samples, particularly when evaluated on datasets with diverse manipulations. Additionally, the Meso4 network's configuration, while capturing intricate details, often fails to adapt to unseen forgery types, highlighting a significant generalization gap. These limitations necessitate the exploration of more advanced and robust models for deepfake detection.

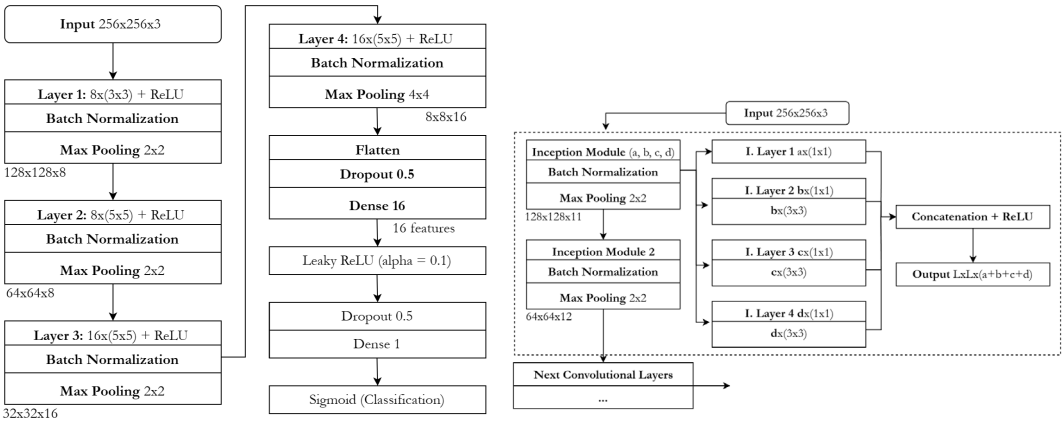

Figure 1 *Block Diagram of the Meso4 network Architecture and Inception Layer from Inception4.*

2.1.1 MesoInception4

The architecture of MesoInception4 builds on the foundational MesoNet design but incorporates additional inception modules to improve feature extraction and pattern recognition. The inception modules are designed to capture multi-scale features by applying multiple convolutional filters of different sizes in parallel. This allows the network to analyze the input data at various scales simultaneously, thereby enhancing its ability to detect subtle manipulations in deepfake images. The MesoInception4

network shows a marked improvement over the Meso4 network, as evidenced by its superior performance across various within-domain evaluations, as can be seen in the table.

Detector	Backbone	FaceForensics++						
		FF-c23	FF-c40	FF-DF	FF-F2F	FF-FS	FF-NT	Avg
Meso4	MesoNet	0.6077	0.5920	0.6771	0.6170	0.5946	0.5701	0.6097
MesoIncept	MesoNet	0.7583	0.7278	0.8542	0.8087	0.7421	0.6517	0.7571
EffNetB4	Efficient	0.9567	0.8150	0.9757	0.9758	0.9797	0.9308	0.9389
EffNetB4*	Efficient	*	*	0.9806	0.9870	0.9708	0.9531	0.9729

Detectors	Overall Gain						
EffNetB4 vs EffNetB4*	*	*	+0.4%	+1%	−0.8%	+2%	+3%
Meso4 vs EffNetB4*	37.29%	39.50%	29.86%	35.88%	37.62%	38.30%	36.32%
MesoIncept vs EffNetB4*	22.23%	25.92%	12.15%	16.71%	22.87%	30.14%	22.23%

Table 1 *AUC Metric in the FaceForesics++ dataset, and an Avg within the domain. The Meso4, MesoIncept and EffNetB4 percentage were extracted from the work made by (Yan et al., 2023). EffNetB4* are the results of my modifications to the EfficientNetB4 detector and backbone. In addition, a Direct comparison of how much increase each model had compared to my implementation.*

2.2 EfficientNet

To address these limitations, I transitioned from Meso4 to EfficientNet, which offers superior performance and accuracy. EfficientNet's advanced architecture, designed for scalability and efficiency, allows it to handle a broader range of compression levels and forgery types more effectively. The t-SNE visualization clearly demonstrates EfficientNet's advantage, as it successfully clusters real and fake samples into distinct groups. This effective separation indicates that EfficientNet can more reliably distinguish between authentic and manipulated images, making it a more robust choice for deepfake detection. The improved clustering and generalization capabilities of EfficientNet underscore its effectiveness in handling diverse and challenging datasets, thus providing a significant improvement over Meso4.

[Figure with Visualization]

EfficientNet is a family of convolutional models *(Tan & Le, 2020)* and they represents a significant advancement in the field of convolutional neural networks (CNNs) by systematically addressing the challenges of model scaling. Traditional methods of scaling CNNs often focus on a single dimension, such as depth, width, or resolution, which can lead to suboptimal performance. EfficientNet introduces a compound scaling method that uniformly scales all three dimensions—depth, width, and resolution—using a compound coefficient. This method ensures a balanced scaling that enhances both accuracy and efficiency. The compound coefficient, denoted as φ in the EfficientNet architecture, is a user-specified parameter that controls the overall scaling of the network. It dictates how much more computational resources are available

for scaling the network. The method ensures that the total FLOPS (floating point operations per second) increase approximately by a factor of 2^{ϕ}, maintaining a balance between the three dimensions of the network.

The EfficientNetV2 *(Tan & Le, 2021)* architecture introduces several key improvements over its predecessor, EfficientNet. One of the primary distinctions is the extensive use of both MBConv and the newly added Fused-MBConv blocks in the early layers of the network. MBConv block aims to further enhance performance by integrating convolutional and depthwise separable convolution operations into a single layer. This fusion helps reduce memory access overhead, which can be a bottleneck in neural network training and inference. EfficientNetV2 also removes the last stride-1 stage present in the original EfficientNet, which significantly reduces the parameter size and memory access overhead.

EfficientNet-B4 by PyTorch (AMP) (*Efficientnet-B4 Pretrained Weights (PyTorch, AMP, ImageNet) | NVIDIA NGC*, n.d.) is part of the EfficientNet model family, known for its high accuracy and efficiency. It leverages mixed precision training with NVIDIA Tensor Cores, significantly speeding up training. The model includes Squeeze-and-Excitation (SE) layers for better performance and is trained on the ImageNet dataset using the NHWC data layout for optimal processing. The model utilizes mixed precision training with Tensor Cores on NVIDIA Volta and Ampere GPUs, which claim to enhance training speed by over 2 times.

3. DESIGN PRINCIPLES

3.1 *JPEG compression principles Using MozJPEG*

In my project, I utilized MozJPEG (*Mozilla/Mozjpeg*, 2014/2024), an advanced JPEG encoder developed by Mozilla, to achieve superior image compression. Progressive Encoding with Jpegrescan Optimization reorders JPEG scan data to reduce file sizes without any loss of quality. Progressive JPEGs are encoded in multiple passes, allowing the image to become incrementally clearer as it loads. MozJPEG uses quantization tables that are specifically tuned for high-resolution displays. This enhances the visual quality of the compressed images, making them look better on modern high-DPI screens.

One of the significant advantages of MozJPEG is that it remains fully compatible with the existing JPEG standard and all web browsers. This compatibility means that images compressed with MozJPEG can be viewed without issues across different platforms and devices.

SmartScale, introduced in *libjpeg v8* (*Libjpeg-Turbo | About / A Study on the Usefulness of DCT Scaling and SmartScale*, n.d.), and DCT scaling are advanced features for JPEG compression. By using techniques such as DCT Scaling which allows

images to be scaled by a factor of $\frac{8}{N}$, where N = 1 to 16, during compression. It operates over multiple blocks but outputs only one, effectively reducing the image size for transmission or previewing.

As mentioned before the tool introduce SmartScale, which allows adjusting DCT block sizes beyond the standard 8x8, providing a wider range of useful scaling factors (from 0.06 to 0.94). This offers additional control over quality versus compression ratio, reducing blocking artifacts by adjusting block sizes to better represent sharp features.

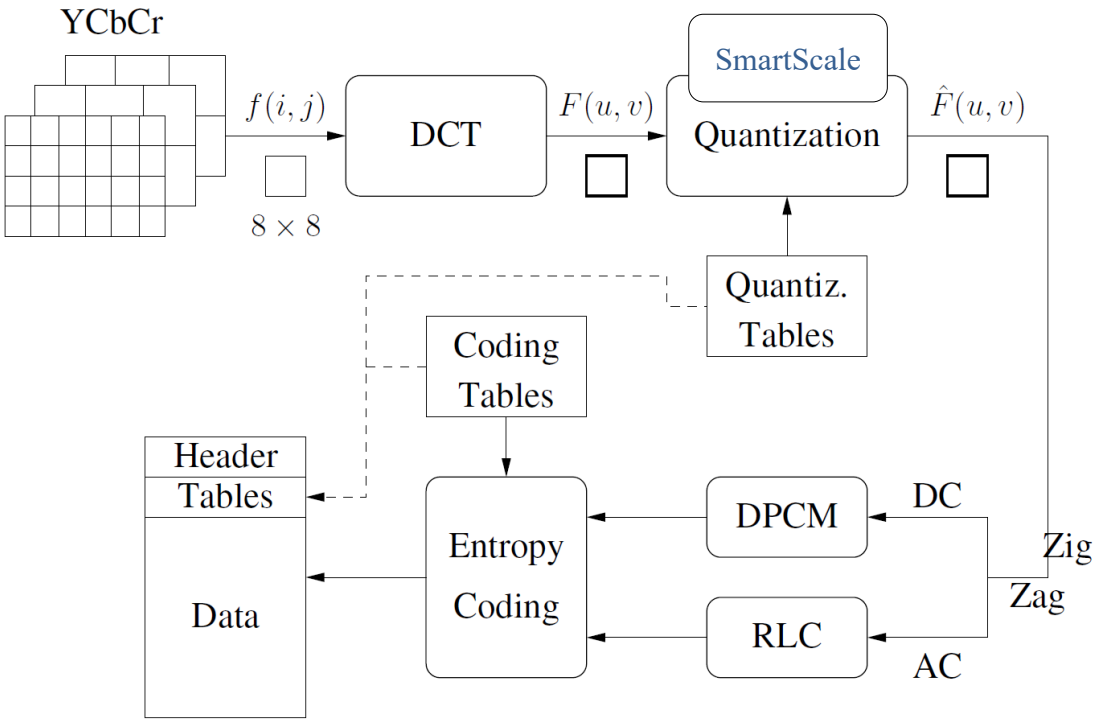

Figure 2 *Quantization with SmartScale: During the quantization step, SmartScale allows for adjusting the DCT block sizes beyond the standard 8x8, providing a wider range of useful scaling factor, (Li et al., 2021).*

3.2 EfficientNets principles and comprehensive understanding

The process of creating EfficientNet-B0 involves leveraging a multi-objective neural architecture search (NAS) that optimizes both accuracy and computational efficiency (measured in FLOPS). Inspired by prior work (Tan & Le, 2020), the optimization goal is defined by the formula:

$$\mathrm{ACC}(m) \times \left(\frac{\mathrm{FLOPS}(m)}{T}\right)^{w}$$

Equation 1 *Optimization Goal Formula*

Where:

- **ACC(m):** Accuracy of model **m.**
- **FLOPS(m):** Floating Point Operations per Second required by model m

- **T:** Target FLOPS.
- **w:** hyperparameter (define at 0.07) that controls the trade-off between accuracy and FLOPS.

The design principles behind EfficientNet revolve around the concept of compound scaling, which balances scaling of depth, width, and resolution of convolutional neural networks (CNNs) to optimize both accuracy and efficiency. The compound scaling method used for the network uses a compound coefficient, ϕ, to uniformly scale the network's width (w), depth (d), and resolution (r) in a principled way:

$$\textbf{\textit{Depth}} \quad (\boldsymbol{d}): \quad \alpha^{\phi}$$
$$\textbf{\textit{Width}} \quad (\boldsymbol{w}): \quad \beta^{\phi}$$
$$\textbf{\textit{Resolution}} \quad (\boldsymbol{r}): \quad \gamma^{\phi}$$

Equation 2 *Scaling factors.*

These scaling factors are subject to the constraint $\alpha \cdot \beta^2 \cdot \gamma \approx 2$, ensuring that the total computational cost (measured in FLOPS) increases approximately by 2^{ϕ}. The constants α, β, and γ are determined through a small grid search.

The steps to Apply Compound Scaling exposed in the work of (Tan & Le, 2020) are summarized as follows:

1) To Determine Scaling Factors, fix $\phi = 1$ and perform a grid search to find optimal α, β, and γ. Using the baseline network, EfficientNet-B0 as example, the best values found were α =1.2, β =1.1 and γ =1.15.

2) To do the Scale Up the Baseline Network: With the constants α, β, and γ fixed, the network is scaled using different values of ϕ to obtain a family of models (EfficientNet-B1 to B7) that progressively increase in size and computational requirements.

EfficientNet	Width	Depth	Resolution	Dropout	FLOPS
B0	1.0	1.0	224	0.2	**0.39B**
B1	1.0	1.1	240	0.2	**0.70B**
B2	1.1	1.2	260	0.3	**1.0B**
B3	1.2	1.4	300	0.3	**1.8B**
B4	1.4	1.8	380	0.4	**4.2B**
B5	1.6	2.2	456	0.4	**9.9B**
B6	1.8	2.6	528	0.5	**19B**
B7	2.0	3.1	600	0.5	**37B**
B8	2.2	3.6	672	0.5	**89.5B**
L2	4.3	5.3	800	0.5	**-**

Table 2 *Pre-defined parameters for each EfficientNet model. Each model (B0 to B8, and L2) has specific values for these parameters, which determine the model's complexity and computational requirements.*

3.3 EfficientNetV2 Architecture and Scaling

EfficientNetV2 builds upon the original EfficientNet by incorporating additional operations and optimizations to enhance training efficiency and parameter utilization.

The Fused-MBConv Blocks integrate convolutional and depthwise separable convolution operations into a single layer to reduce memory access overhead. The Smaller Expansion Ratios in EfficientNetV2 prefers smaller expansion ratios in MBConv blocks to decrease memory usage. By using Smaller Kernel Sizes of 3x3 kernel sizes, the network compensates by adding more layers to maintain a wide receptive field. It also removes the Last Stride-1 Stage: This reduces parameter size and memory overhead.

EfficientNetV2 uses a similar compound scaling method as EfficientNet but with some more added optimizations such as Restriction of Maximum Image Size and make the inference image size limited to 480 pixels to avoid excessive memory consumption. Gradual Layer Addition: More layers are added to later stages of the network (e.g., stages 5 and 6) to increase capacity without significantly increasing runtime overhead. This type of balances and optimize scaling strategy ensures that EfficientNetV2 models maintain high accuracy while being efficient in terms of computational resources and memory usage.

4. RESEARCH METHODS AND STEPS

This section outlines the detailed methodology and steps undertaken in the research to develop and evaluate an effective image forgery detection model using EfficientNet and advanced JPEG compression techniques.

4.1 Research Methods and Objectives

The primary objective of this research is to enhance the accuracy and robustness of deepfake detection models, particularly under varying levels of JPEG lossy compression. The study aims to:

1) Utilize advanced JPEG compression techniques, specifically MozJPEG, to simulate real-world image compression scenarios.
2) Moved from a MesoNet's models to a more efficient network.
3) Develop a robust baseline model using EfficientNet architecture.
4) Incorporate multi-attentional mechanisms to improve feature extraction.
5) Evaluate the performance of the proposed model against traditional deepfake detection models.

The Dataset Preparation consisted in collect and preprocess Faceforensics++ datasets, including both original and manipulated images. Then, apply various levels of JPEG compression to the datasets using MozJPEG to simulate real-world scenarios.

Use of MozJPEG was facilitated through a Python script by using the subprocess

module and passing the following command as an argument to subprocess called in the script. Here is an example of the command used:

```
cjpeg -quality (n, from 1 - 100) -outfile output_image input_image
```

Training and Evaluation was done by training the EfficientNet-B4 model on compressed datasets. Use metrics such as AUC, ACC, and F1-score to evaluate the model's performance. Then, compare the performance of the proposed model with traditional models like Meso4 and MesoInception4.

Experimental Setup Detail The hardware and software environment used for training and evaluation were mostly set up using paid versions of Google Cloud's services. Data storage and management were handled through Google Drive, while the training and evaluation of models were conducted using Google Colab Pro, an interactive notebook environment that supports high-end GPU and TPU accelerators. The specific hyperparameters, learning rate schedules, and optimization techniques applied during training are detailed in the preceding sections.

4.2 Current State of Related Research and Comparison

In the preceding sections, we reviewed the limitations of existing models such as Meso4 and MesoInception4. EfficientNet, particularly the EfficientNet-B4 model, was identified as a more effective alternative due to its superior performance in handling complex image manipulations and various compression levels.

5. SYSTEM IMPLEMENTATION AND EXPERIMENTS

5.1 JPEG Lossy Compression utilizing MozJPEG

In this section, I demonstrate the effects of JPEG lossy compression using MozJPEG on a sample image. MozJPEG is an advanced JPEG encoder that improves compression efficiency while maintaining high visual quality. We applied various levels of JPEG compression to the image and evaluated the results using two common image quality metrics: Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM).

5.1.1 PSNR (Peak Signal-to-Noise Ratio)

PSNR is a metric used to measure the quality of a reconstructed image compared to its original version. In this metric, higher PSNR values indicate better image quality and less distortion. It is expressed in decibels (dB) and calculated as follows:

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{\mathrm{MAX}_I^2}{\mathrm{MSE}}\right)$$

Equation 3 *Peak Signal-to-Noise Ratio,*

Where:

- MAX_I^2 is the maximum possible pixel value of the image (255 for an 8-bit image).
- MSE is the Mean Squared Error between the original and compressed image.

5.1.2 SSIM (Structural Similarity Index)

SSIM is a perceptual metric that assesses image quality based on the degradation of structural information. It considers luminance, contrast, and structure. The SSIM index is calculated as follows:

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

Equation 4 *Structural Similarity Index.*

Where:

- μ_x and μ_y are the mean values of the original and compressed images.
- σ_x^2 and σ_y^2 are the variances of the original and compressed images.
- σ_{xy} is the covariance between the original and compressed images.
- C_1 and C_2 are constants to stabilize the division with weak denominators.

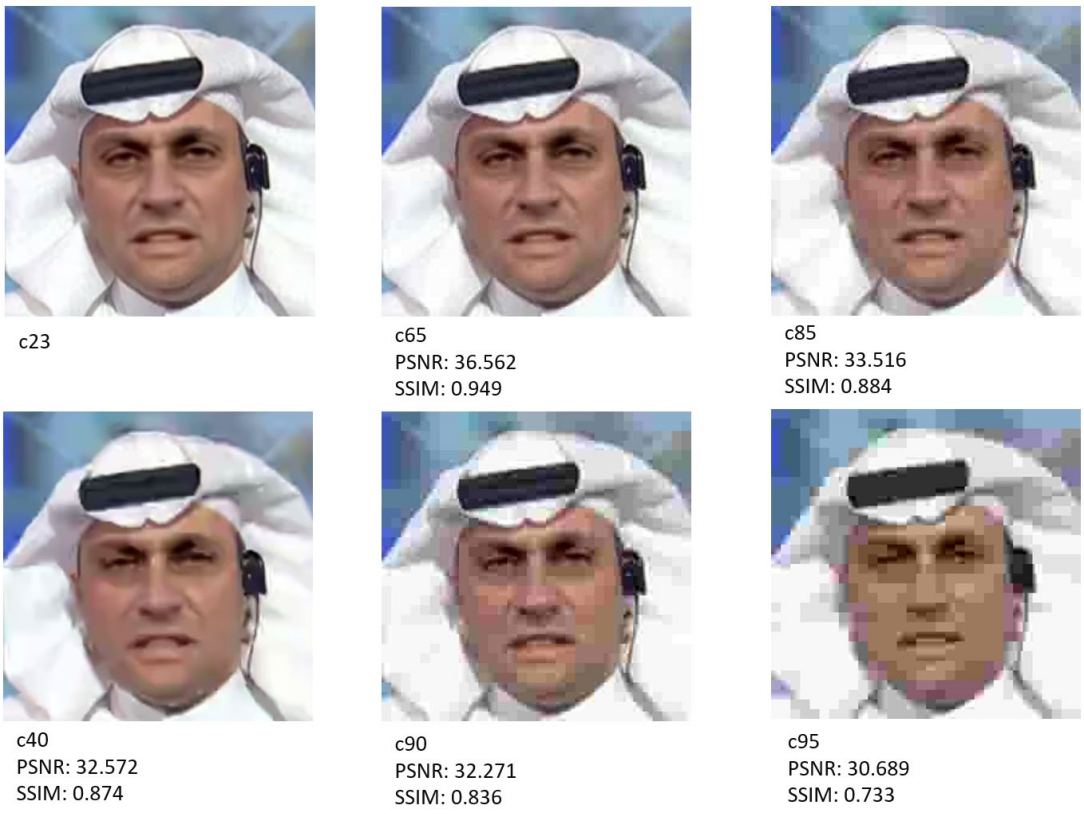

Figure 3 *Higher SSIM values indicate better image quality and higher similarity to the original image.*

To calculate the SSIM values, I utilized the compare_ssim function from the scikit-image library in Python. This function provides a straightforward way to compute the SSIM index between two images, (*Structural Similarity Index — Skimage 0.23.2*

Documentation, n.d.). The scikit-image library's implementation of SSIM ensures reliable and consistent quality evaluation, aiding in the analysis of compression effects on image quality.

5.1.3 Compression Results

The image in **[Figure x]** shows the effects of different levels of JPEG lossy compression using MozJPEG. The PSNR and SSIM values for each compression level are provided to quantify the image quality. As observed, higher compression levels (lower quality settings) result in lower PSNR and SSIM values, indicating increased image degradation. However, even at higher compression levels, MozJPEG manages to maintain a reasonable balance between file size reduction and image quality, making it a suitable choice for efficient image storage and transmission.

5.2 EfficientNet-b4 Detection, Real or Deepfake?

The detection of real versus deepfake images using EfficientNet-B4 involves a comprehensive and systematic process. Following the procedures from (Yan et al., 2023) Benchmark, the process begins with data preprocessing, moves through various detection modules, and culminates in evaluation and analysis.

5.2.1 Data Preprocessing:

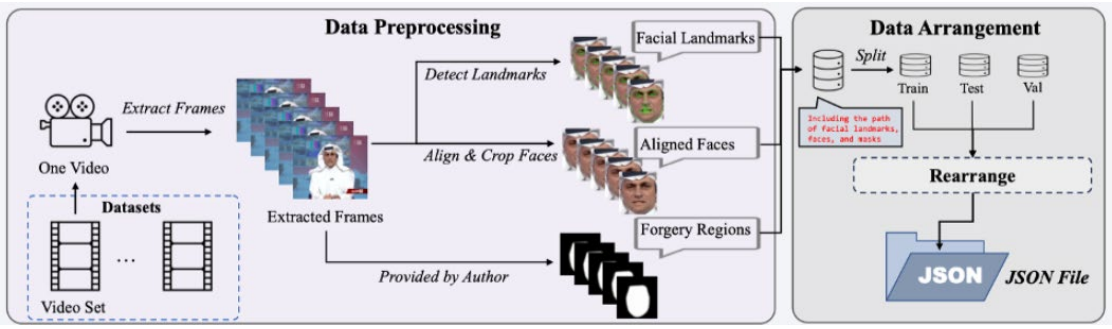

Figure 4 *Picture took from (Yan et al., 2023) benchmark github. In the preprocess for FaceForensics++ dataset. Frames are extracted from video datasets. Each video is broken down into individual frames. Facial landmarks are detected in each frame, and the faces are aligned and cropped to standardize the input data. Then the paths to the facial landmarks, faces, and masks are organized and stored in a JSON file, facilitating easy access and management during training and evaluation.*

In my work I adapted the rearrange.py file and changed it into an IPYNB file to facilitate the execution of the data rearrangement process and save the JSON files directly in my Drive storage unit. After running the rearrange script you will be left with the paths to the facial landmarks, faces, and masks such as `FaceForensics++.json`, `FF-DF.json`, `FF-F2F.json`, `FF-FS.json` and `FF-NT.json`.

5.2.2 Data Augmentation results

To enhance the robustness of the model, more frames are extracted per second than typically necessary. This increased frame rate acts as a form of data augmentation,

providing a larger and more varied dataset for training. The additional frames help the model generalize better by exposing it to a wider range of facial expressions, lighting conditions, and angles.

Before Data Augmentation	
Dataset	**AUC Score**
FF-F2F	0.883
FF-DF	0.930
FF-FS	0.916
FF-NT	0.718
After Data Augmentation	
FF-F2F	0.910
FF-DF	0.938
FF-FS	0.940
FF-NT	0.757
Improvement (Percentage)	
FF-F2F	+2.99%
FF-DF	+0.91%
FF-FS	+2.64%
FF-NT	+5.43%

Table 3 *These values represent the percentage improvements in AUC scores for each dataset after applying data augmentation.*

5.2.3 Training Module

The aligned and preprocessed images are fed into the EfficientNet-B4 model. As a naïve detector, EfficientNet-B4 is trained to classify images as either fake or real based on the provided training samples.

The training environment was set up in Google Colab, leveraging its interactive notebook capabilities. The environment setup involved converting Python scripts (PY) for rearranging data, training models, and calculating metrics into Jupyter notebooks (IPYNB) for seamless execution and monitoring. Most of the training was conducted using Google Colab's **L4 GPU**, providing the necessary computational power for efficient training of the deep learning models.

To mitigate the risk of progress loss due to Colab's runtime limitations, I modified the *trainer/trainer.py* script to save a checkpoint every *500 iterations*. This modification allows for the continuation of training from the last checkpoint in case of an interruption. The training script is also designed to load the last saved model and resume training from the last epoch and iteration.

5.2.3.1 Checkpoint Saving Mechanism

To save the model's state, the following parameters are captured and stored:

- *Epoch:* The current epoch number.
- *Iteration:* The current iteration number within the epoch.

- *Model State Dictionary:* The state of the model parameters, captured using self.model.state_dict().
- *Optimizer State Dictionary:* The state of the optimizer, captured using self.optimizer.state_dict().

6. IMPORTANT CONTRIBUTIONS OF THE PROJECT

This project has made several significant contributions to the field of image forgery detection, particularly in the context of deepfake detection under varying levels of JPEG lossy compression. The advancements achieved through this research can be summarized as follows:

1) *Enhanced Deepfake Detection Accuracy*
   By leveraging the EfficientNet-B4 model, this project has substantially improved the accuracy of detecting deepfake images. The EfficientNet architecture, known for its scalability and efficiency, provides a robust framework for distinguishing between real and manipulated images.
2) *Robustness Under Compression*
   A critical challenge in deepfake detection is maintaining accuracy under different compression levels, which are common in digital media. This project utilized MozJPEG, an advanced JPEG encoder, to simulate real-world compression scenarios.
3) *Comprehensive Data Augmentation*
   The project employed an extensive data augmentation strategy by extracting more frames from videos, increasing the robustness of the model. This approach provided a larger and more varied dataset, helping the model generalize better.
4) *Efficient Training and Checkpointing*
   The training process was optimized for efficiency by utilizing Google Colab's L4 GPU, enabling faster convergence and handling of complex models. Additionally, the implementation of a checkpointing mechanism. Beside I optimize the pipeline by

7. EFFICIENCY EVALUATION AND RESULTS

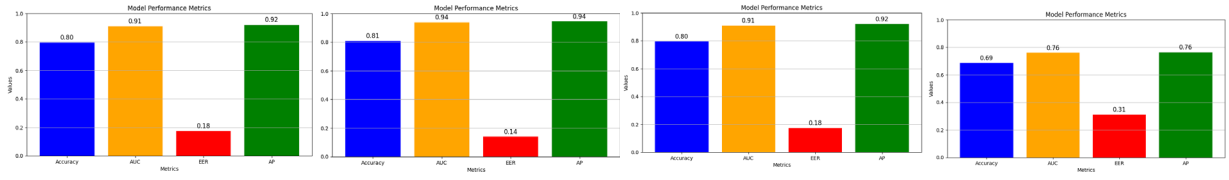

Figure 5 *visualize the single metric values in a bar chart.*

8. CONCLUSIONS

The contributions of this project extend beyond the immediate improvements in deepfake detection accuracy and robustness. By addressing key challenges such as image compression, the project provides a comprehensive framework for future research and practical applications in the field of digital forensics. The methodologies and tools developed herein offer valuable insights and solutions that can be adopted and further refined by the broader research community.

9. REFERENCES

[1] Afchar, D., Nozick, V., Yamagishi, J., & Echizen, I. (2018). MesoNet: A Compact Facial Video Forgery Detection Network. *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, 1–7. https://doi.org/10.1109/WIFS.2018.8630761

[2] *Efficientnet-b4 pretrained weights (PyTorch, AMP, ImageNet) | NVIDIA NGC*. (n.d.). NVIDIA NGC Catalog. Retrieved May 26, 2024, from https://catalog.ngc.nvidia.com/orgs/nvidia/models/efficientnet_b4_pyt_amp

[3] Li, Z.-N., Drew, M. S., & Liu, J. (2021). *Fundamentals of Multimedia*. Springer International Publishing. https://doi.org/10.1007/978-3-030-62124-7

[4] *Libjpeg-turbo | About / A Study on the Usefulness of DCT Scaling and SmartScale*. (n.d.). Retrieved May 27, 2024, from https://libjpeg-turbo.org/About/SmartScale

[5] *Mozilla/mozjpeg*. (2024). [C]. Mozilla. https://github.com/mozilla/mozjpeg (Original work published 2014)

[6] *Structural similarity index—Skimage 0.23.2 documentation*. (n.d.). Retrieved May 27, 2024, from https://scikit-image.org/docs/stable/auto_examples/transform/plot_ssim.html

[7] Tan, M., & Le, Q. V. (2020). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks* (arXiv:1905.11946). arXiv. https://doi.org/10.48550/arXiv.1905.11946

[8] Tan, M., & Le, Q. V. (2021). *EfficientNetV2: Smaller Models and Faster Training* (arXiv:2104.00298). arXiv. http://arxiv.org/abs/2104.00298

[9] Yan, Z., Zhang, Y., Yuan, X., Lyu, S., & Wu, B. (2023). *DeepfakeBench: A Comprehensive Benchmark of Deepfake Detection* (arXiv:2307.01426). arXiv. https://doi.org/10.48550/arXiv.2307.01426

[10] Zhao, H., Zhou, W., Chen, D., Wei, T., Zhang, W., & Yu, N. (2021). *Multi-attentional Deepfake Detection* (arXiv:2103.02406). arXiv. http://arxiv.org/abs/2103.02406