King Saud University
College of Computer and Information Sciences
Department of Information Technology
IT 461 course project - Semester 1, 1447H

جامعة الملك سعود
King Saud University
كلية علوم الحاسب والمعلومات
قسم تقنية المعلومات

Final Report
< DeepDetect >

Prepared by

Student name	Student ID
Nora Alasmari	444200062
Norah Alfaheed	444200779
Hissah Alotaibi	444200349
Rawan Albatati	443204566

Supervised by
Dr. Luluah Alhusain

Table of Contents

Table of Figures

Table of Tables

1 Introduction

The rapid advancement of artificial intelligence has made it increasingly difficult to distinguish real facial images from manipulated ones. Modern deepfake technology can alter a person's appearance, expressions, or identity with a level of realism that often escapes human detection. As these manipulated images spread across social platforms, they pose significant risks to digital trust, personal security, and the credibility of visual information. This growing challenge has created a critical need for automated systems capable of identifying subtle manipulation patterns that are not visible to the human eye.

To understand the nature of this problem and the technologies behind it, the following section provides essential background on deepfakes, and the mechanisms used to generate them.

Deepfakes: are digitally manipulated facial images created using deep learning models that learn and replicate fine details of human appearance. A deepfake is generated by training neural networks to imitate facial features such as texture, lighting, geometry, and expression patterns, allowing the system to reconstruct or modify a face in a highly realistic way.

Two foundational architectures enable deepfake generation:

- **Autoencoders**: Neural networks that compress facial images into a latent representation and reconstruct them. When two autoencoders are jointly trained, they can transfer one person's expressions onto another's identity, enabling face-swapping.
- **Generative Adversarial Networks (GANs):** A generator creates synthetic images while a discriminator tries to detect them. Through this adversarial training, the generator learns to produce highly realistic faces.

Deepfake manipulation generally appears in three forms:

Face Swapping: Replacing one person's identity with another.

Face Reenactment: Transferring expressions or mouth movements from a source person to a target face.

Identity Synthesis: Creating new faces that do not belong to real individuals.

While deepfakes can be used creatively, they introduce risks such as misinformation, identity fraud, reputational damage, and non-consensual manipulated content. The rapid

improvement of generative models and the accessibility of tools make detection increasingly challenging.

The goal of this project is to build a machine-learning system capable of determining whether a cropped facial image is **Real** or **Deepfake**. The classifier analyzes fine-grained facial cues including texture consistency, lighting coherence, and structural alignment to identify signs of manipulation that may not be detectable by human observers. The task is formulated as a binary classification problem in which the **input** is a single cropped facial image, and the **output** is a label predicting whether the face is authentic "Real" or manipulated "fake", Figure 1 provides a high-level overview of the system's workflow, showing how the data flows through the model from input to final prediction.

To assess performance, the system uses standard binary classification metrics:

- **Accuracy:** Overall correctness of predictions.
- **Precision:** How reliably deepfakes are identified without false positives.
- **Recall:** How effectively true deepfakes are detected.
- **F1-Score:** A balanced combination of precision and recall.
- **Loss:** Measures training stability and model convergence.

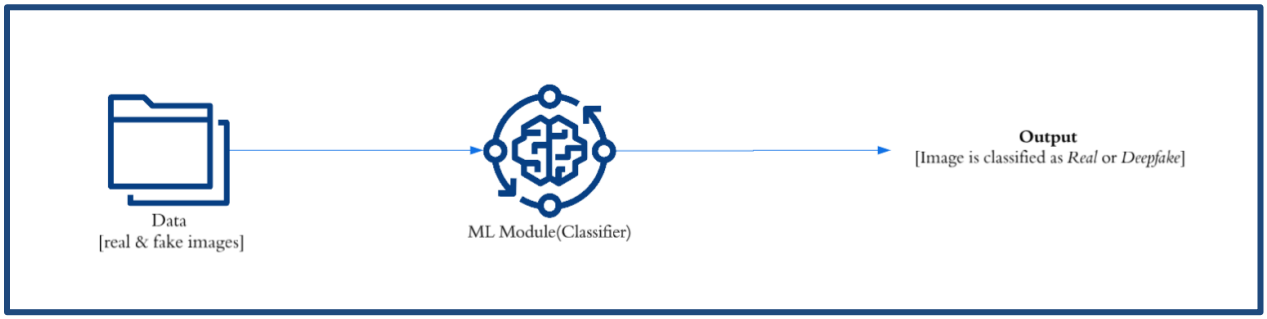

Figure 1: Deepfake image classification process

2 Related Works

1. Rössler et al. (2018) – FaceForensics++ [1]

Rössler et al. introduced the FaceForensics++ dataset and evaluated several CNN models for detecting manipulated facial videos. Their study highlighted how visual artifacts and compression patterns can be used to distinguish real from fake content, establishing one of the strongest baselines in deepfake detection research. This work is directly relevant to DeepDetect because it established the baseline architectures and datasets that our project also builds upon.

2. Yu et al. (2019) – Capsule Networks for Deepfake Detection [2]

Yu et al. proposed a Capsule Network approach to detect deepfakes by modeling spatial relationships between facial features. Their method focused on analyzing static images from the DeepFake-TIMIT dataset, showing that capsule-based feature grouping can improve sensitivity to subtle manipulations. The study is related to DeepDetect because it highlights alternative feature-aware architectures.

3. Dolhansky et al. (2020) – Deepfake Detection Challenge [3]

Dolhansky et al. released the DFDC dataset and benchmarked CNN-based architectures enhanced with texture cues and heavy augmentation. Their work emphasized real-world variability and robustness, making DFDC a practical benchmark for deepfake detection systems. This work relates to DeepDetect because it emphasizes generalization and robustness two goals that our models also aim to address through diverse image preprocessing and model comparisons.

4. Zhang et al. (2020) – Pixel-Level Artifact Detection [4]

Zhang et al. introduced a pixel-level CNN model trained on Celeb-DF to identify fine-grained artifacts left by deepfake generation techniques. Their study demonstrated that focusing on subtle low-level textures can significantly improve detection performance. This research is relevant to DeepDetect because our EfficientNetB0 and Xception models similarly rely on detailed pixel-level patterns to differentiate real and fake images.

Table 1: Summary Table of Related Works

Study	Target Task	Dataset Used	Method / Algorithm	Performance
Rössler et al., 2018	Detect manipulated facial videos	FaceForensics++ (Video)	CNN	~93%
Yu et al., 2019	Image-based deepfake detection	DeepFake-TIMIT (Images)	Capsule Network (CapsNet)	~94%
Dolhansky et al., 2020	Robust deepfake detection on diverse conditions	DFDC (Video)	CNN + Texture features	92–95%
Zhang et al., 2020	Pixel-level detection of AI-generated artifacts	Celeb-DF (Images)	CNN (Pixel-level artifacts)	~96%

3 Dataset

For this project, we utilize the Deepfake and Real Images dataset by Manjil Karki [5], which provides a large collection of labeled **face images** categorized as either Real or Fake. The dataset is specifically designed for training and evaluating deepfake detection models, making it well-suited for our binary classification task. Each image is labeled, allowing supervised learning algorithms to distinguish between authentic and manipulated faces.

We selected this dataset for several reasons. First, it contains a large number of examples, with over 70,000 fake images and several thousand real images, providing sufficient material for both training and testing phases. Second, the dataset includes diverse samples of manipulated faces generated by different deepfake techniques, which improves the robustness and generalization capability of the model. Finally, the dataset is publicly available on Kaggle, making it a reliable and easily accessible resource.

3.1 Summary Statistic

Table 2: Distribution of images across the Deepfake and Real Images dataset

Dataset (Deepfake and Real Images)	Class	Number of examples (Images)	Features
Train	Real	70,000	RGB pixels (~256×256)
	Fake	70,000	RGB pixels (~256×256)
Test	Real	5,413	RGB pixels (~256×256)
	Fake	5,492	RGB pixels (~256×256)

3.2 Representative Examples

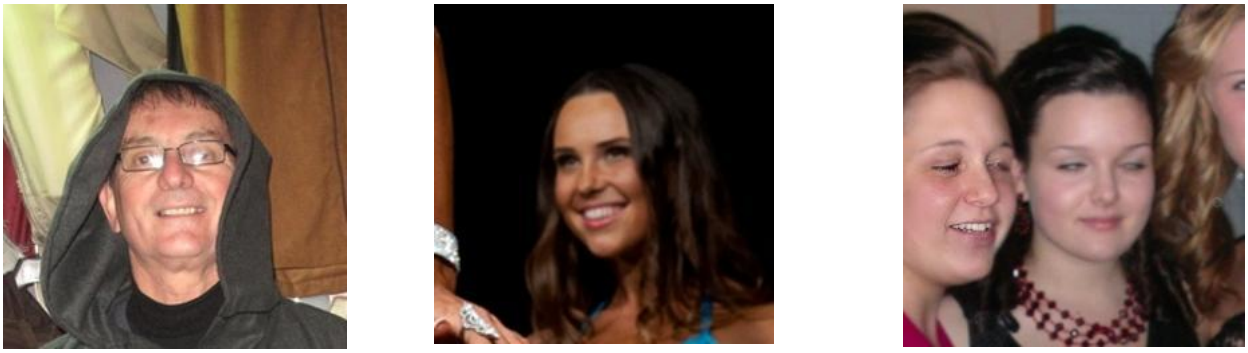

Figure 2: Examples of real images from the dataset

Figure 3: Examples of fake images from the dataset

Figures 2 show samples of Real face images from the dataset, while Figures 3 shows samples of Fake face images. These examples highlight the challenge of the task: although deepfake images may appear highly realistic, subtle artifacts in texture, lighting, or facial alignment can be detected by machine learning models.

3.3 Data Cleaning and Preparation

Since the dataset was originally released in a clean and well-structured format, no manual cleaning or relabeling was required. The only preparation step involved organizing the images into consistent train, validation, and test splits and ensuring that each class (*Real* and *Fake*) was placed in its corresponding directory. This provided a stable and noise-free foundation for the later preprocessing steps applied during model training.

4 Methods

This project uses transfer learning for binary image classification with three CNN architectures: ResNet50, Xception, and EfficientNetB0. The workflow includes model selection, preprocessing, feature engineering, dimensionality reduction, and training strategy.

4.1 Models

1. **ResNet50** – Uses residual connections to prevent vanishing gradients. Loaded with include_top=False to replace the classifier for the task.
2. **Xception** – Employs depthwise-separable convolutions for efficient texture analysis, suitable for subtle visual differences.
3. **EfficientNetB0** – Uses compound scaling for depth, width, and resolution, balances accuracy and efficiency.

4.2 Justification

Transfer learning is particularly suitable for deepfake detection because the task relies on identifying subtle textural inconsistencies and high-frequency artifacts that pre-trained CNNs have learned to recognize from large-scale datasets.

- **ResNet50:** Chosen for its deep residual connections, which enable the model to capture complex hierarchical features without vanishing gradients. This makes it effective for detecting fine spatial inconsistencies typical in manipulated facial regions.
- **Xception:** Selected because its depthwise-separable convolutions capture local texture variations more efficiently than standard convolutions. Deepfakes often introduce unnatural blending artifacts and boundary distortions, making Xception highly suitable for this task.
- **EfficientNetB0:** Used to balance accuracy and computational efficiency. Its compound scaling enables strong feature extraction while keeping the model lightweight, which is ideal for large datasets and faster experimentation.

4.3 Common Setup

- Pre-trained on ImageNet.
- Global Average Pooling for dimensionality reduction.
- Dropout (0.3) to reduce overfitting.
- Dense layer with sigmoid for binary classification.

4.4 Preprocessing & Feature Engineering

- Resize images (224×224 or 256×256) and normalize with model-specific preprocessing.
- Data augmentation: horizontal flip, rotation (5–10%), zoom (10–20%).
- Global Average Pooling compresses feature maps to fixed-length vectors.

4.5 Training Strategy

- **Stage 1**: Freeze backbone, train classifier with Adam (lr=1e-4), binary cross-entropy, metrics: accuracy, precision, recall, F1.
- **Stage 2**: Fine-tune with reduced learning rate (1e-5 or 3e-5).
- Callbacks: ModelCheckpoint, EarlyStopping, ReduceLROnPlateau.
- Epochs: ResNet50=20, Xception=15, EfficientNetB0=15.

5 Experiment

This section presents the full training and evaluation pipeline used to develop four deep learning models for deepfake detection: **ResNet50 (Attempt 1), ResNet50 (Attempt 2), EfficientNetB0, and Xception**. The goal was to analyze how data preprocessing, augmentation strength, architectural design, and hyperparameter settings affected the stability and performance of each model.

5.1 Data Preparation, Feature Processing, and Dataset Splitting

The dataset contains more than 140,000 labeled images of *Real* and *Fake* faces. Before training, the dataset was cleaned, organized, and split into three subsets:

Dataset Splitting

We applied the following division strategy:

- **Training set: 70%**
- **Validation set: 20%**
- **Testing set: 10%**

This ensures:

- The model learns from the majority of images
- Validation monitors generalization during training
- Testing provides unbiased final evaluation

Feature Engineering & Normalization

No handcrafted features were used. Instead, models learned directly from image pixels after applying:

- **Image resizing**
    - ResNet50 → 256×256
    - EfficientNetB0 & Xception → 224×224
- **Normalization**: pixel intensity scaling to match the statistical distribution expected by ImageNet-pretrained weights

- **Shuffling**, **prefetching**, and **streamed batch loading** to stabilize and speed up the training process

Each training loop used fixed numbers of steps per epoch to maintain consistent epoch lengths despite the dataset size.

Table 3 : Training setup for each model

Model	Input Size	Batch Size	Steps per epoch
ResNet50 v1	**256×256**	**8**	**2000**
ResNet50 v2	**256×256**	**4**	**4000**
EfficientNetB0	**224×224**	**4**	**4000**
Xception	**224×224**	**2**	**4000**

5.2 Data Augmentation & Regularization Techniques

To improve the model's ability to generalize to unseen images, real-time augmentation was applied during training. Augmentation strength differed between experiments:

Augmentation Techniques

- Horizontal flipping
- Light rotations
- Zoom transformations
- (ResNet50 v1 only) random contrast alteration

ResNet50 v1 used **aggressive augmentation** to improve robustness but sometimes introduced noise; ResNet50 v2 and the remaining models used **milder augmentation**, preserving subtle deepfake artifacts.

Regularization

To prevent overfitting:

- Dropout (0.3)
- Early stopping based on validation loss/F1

- Automatic learning rate reduction
- Class weighting (ResNet50 v1 only)

These strategies ensured more stable fine-tuning.

Table 4 : Model-specific augmentation and regularization

Model	Rotation	Zoom	Contrast	Dropout	Class Weights
ResNet50 v1	0.10	0.20	Yes	0.3	Yes
ResNet50 v2	0.05	0.10	No	0.3	No
EfficientNetB0	Mild	Mild	No	0.3	No
Xception	Mild	Mild	No	0.3	No

5.3 Model Architecture and Structure

All models shared the same high-level structure:

1. Input layer
2. Data augmentation block
3. Normalization layer
4. Pretrained CNN backbone (top removed)
5. Global average pooling
6. Dropout for regularization
7. Sigmoid output neuron for binary classification

Two-Stage Training

Stage 1 : Frozen Backbone

Only the newly added classifier layers were trained.

Stage 2 :Fine-Tuning

A portion of the backbone layers was unfrozen:

- ResNet50 v1 → last **100 layers**
- ResNet50 v2 → last **80 layers**
- EfficientNetB0 → fine-tuned the last convolutional block (~25 layers)
- Xception → upper block (~80 layers)

This allowed each model to refine high-level facial features that represent deepfake inconsistencies.

Table 5 : Fine-tuning configuration per model

Model	Fine-Tuned Layers	Training Stages
ResNet50 v1	**100**	**2**
ResNet50 v2	**80**	**2**
EfficientNetB0	**~25**	**2**
Xception	**~80**	**2**

5.4 Hyperparameter Tuning Process

Hyperparameters were tuned manually through iterative experimentation due to the large dataset and GPU constraints.

Hyperparameters Tuned

- **Learning rate:** explored 1e-3 → 1e-5
- **Batch size:** 2, 4, 8
- **Input resolution:** 224×224 vs 256×256
- **Number of fine-tuned layers:** varied per model
- **Dropout rate:** between 0.3–0.5
- **Augmentation strength**: strong vs mild
- **Steps per epoch:** 2000 vs 4000

Selection Criteria

The final values were chosen based on:

- Highest validation F1
- Smooth convergence
- Minimal overfitting
- Stable fine-tuning performance

Table 6 : Selected hyperparameters per model

Model	LR Stage 1	LR FT Stage	Epochs
ResNet50 v1	1e-4	1e-5	20 + FT 8
ResNet50 v2	1e-4	3e-5	15+ FT 4
EfficientNetB0	1e-4	3e-5	15+ FT 4
Xception	1e-4	3e-5	15+ FT 4

5.5 Evaluation Method and Metrics

After training, each model was evaluated on the **test set**, which remained completely unseen during training.

Metrics Used

- **Accuracy**: overall correctness
- **Precision**: reliability of positive predictions
- **Recall**: ability to recover all positive samples
- **F1-score** (main metric)
- **Confusion matrix** for visualizing errors

5.6 Libraries and Computational Resources (Final Short Version)

Software Libraries

The experiments were implemented using a set of standard deep-learning and data-processing libraries:

- **TensorFlow / Keras** – for model building, pretrained backbones, training loops, and callbacks.
- **NumPy** – numerical operations and tensor manipulation.
- **scikit-learn** – evaluation metrics and confusion matrices.
- **Matplotlib** – visualizing training curves and results.

These libraries provided all essential functions for data preparation, transfer learning, fine-tuning, and evaluating model performance.

Computational Resources

Training was executed across multiple hardware configurations depending on availability and experiment size:

- **CPU** – used for initial testing and light preprocessing.
- **NVIDIA T4 GPU** – used for medium-scale training and validation cycles.
- **NVIDIA A100 GPU** – used for full fine-tuning and high-load experiments due to its higher memory and compute power.

The combination of these resources enabled efficient training despite the large dataset and deep CNN architectures.

6 Results and Discussion

This section presents the performance evaluation of the three implemented deep learning models Xception, EfficientNetB0, and ResNet50 for real versus fake image classification. The analysis includes quantitative results, generalization capabilities, learning behavior, and visualizations to support the findings.

6.1 Overall Performance Results

The final performance of the models is summarized in Table 7, which presents accuracy, F1-scores for both classes, and key performance observations.

Table 7 :Overall performance comparison of the three models

Model	Accuracy	F1-Score (Fake)	F1-Score (Real)	Observations
Xception	**0.8774**	**0.8837**	**0.8703**	**Strong convergence, stable curves, excellent balance**
EfficientNetB0	**0.8915**	**0.8911**	**0.8919**	**Best balanced performance, highest accuracy among all models.**
ResNet50 (v1 and v2)	**~0.83**	**~0.95**	**0.93**	**Very strong detection of Fake, slightly weaker overall accuracy compared to Xception & EfficientNet.**

EfficientNetB0 achieved the strongest overall performance, while Xception provided very stable behavior and solid generalization. ResNet50 showed high sensitivity to fake images due to its deep hierarchical feature extraction, and both of its variants (v1 and v2) produced almost identical results, suggesting that architectural differences between them had negligible influence on final performance.

6.2 Generalization to Unseen Data

All models demonstrated good generalization to unseen test samples, with EfficientNetB0 showing the most consistent predictions across both classes. Xception also generalized well, as reflected in its balanced F1-scores and smooth validation curves.

ResNet50, although initially underperforming, improved significantly after fine-tuning. Its high recall for Fake images (~0.95) indicates strong detection capability, but its slightly lower accuracy compared to the other two models suggests that further preprocessing (e.g., face cropping, artifact enhancement) could improve generalization even more.

6.3 Model Visualizations

- Xception

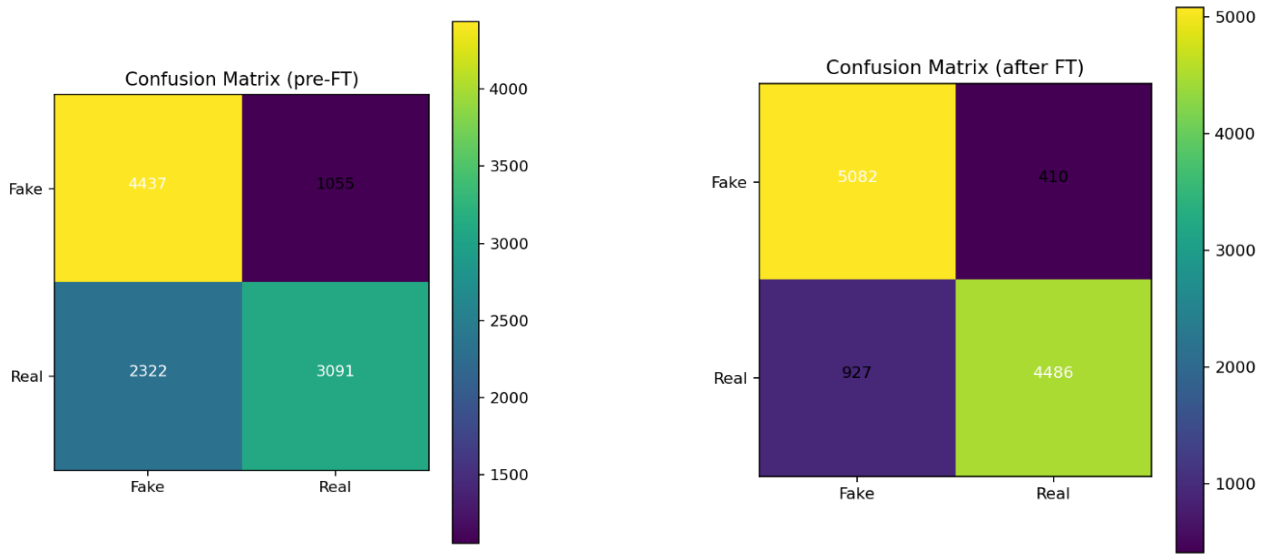

Figure 4 :Confusion matrix of Xception before and after FT

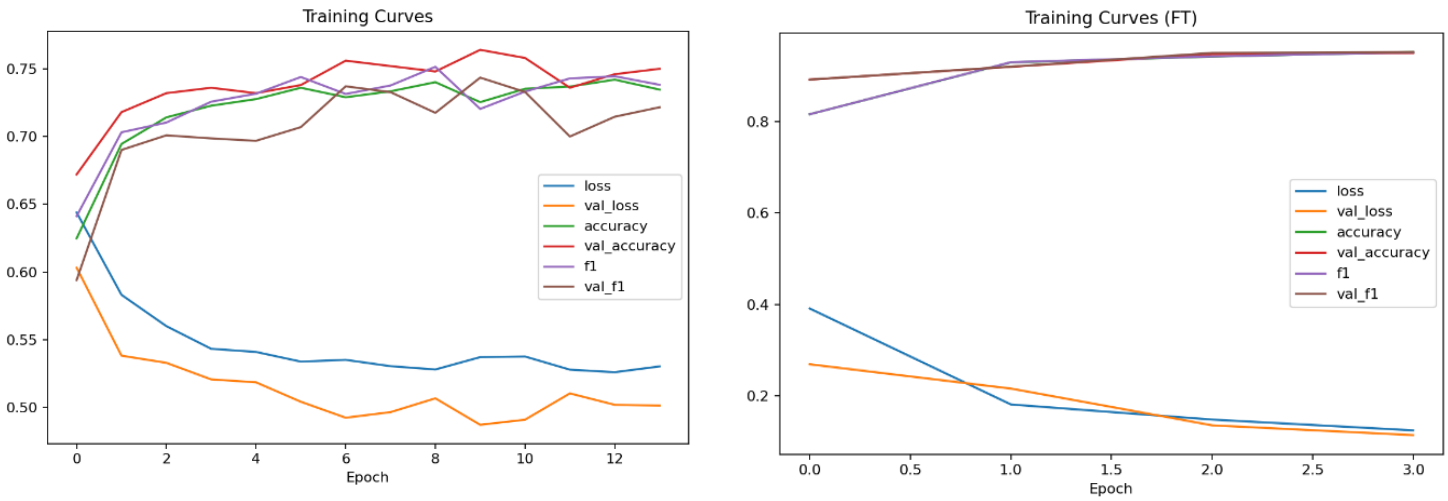

Figure 5: Training Curves of Xception before and after FT

- EfficientNetB0

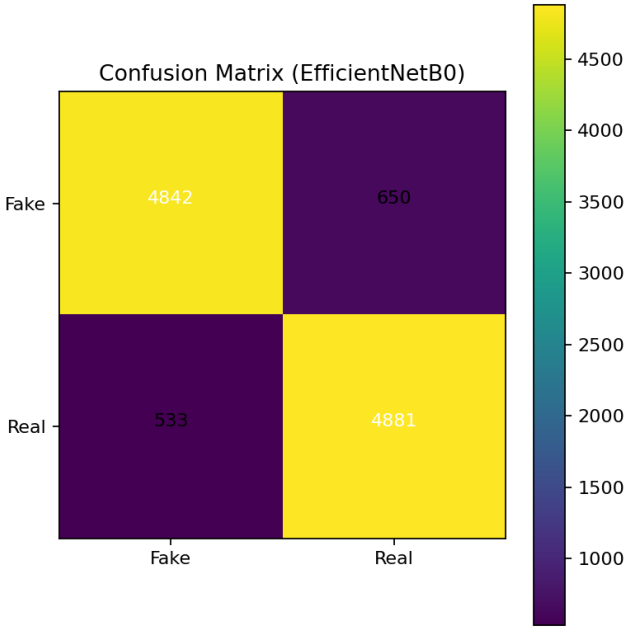

Figure 6: Confusion matrix of the EfficientNetB0 model after FT

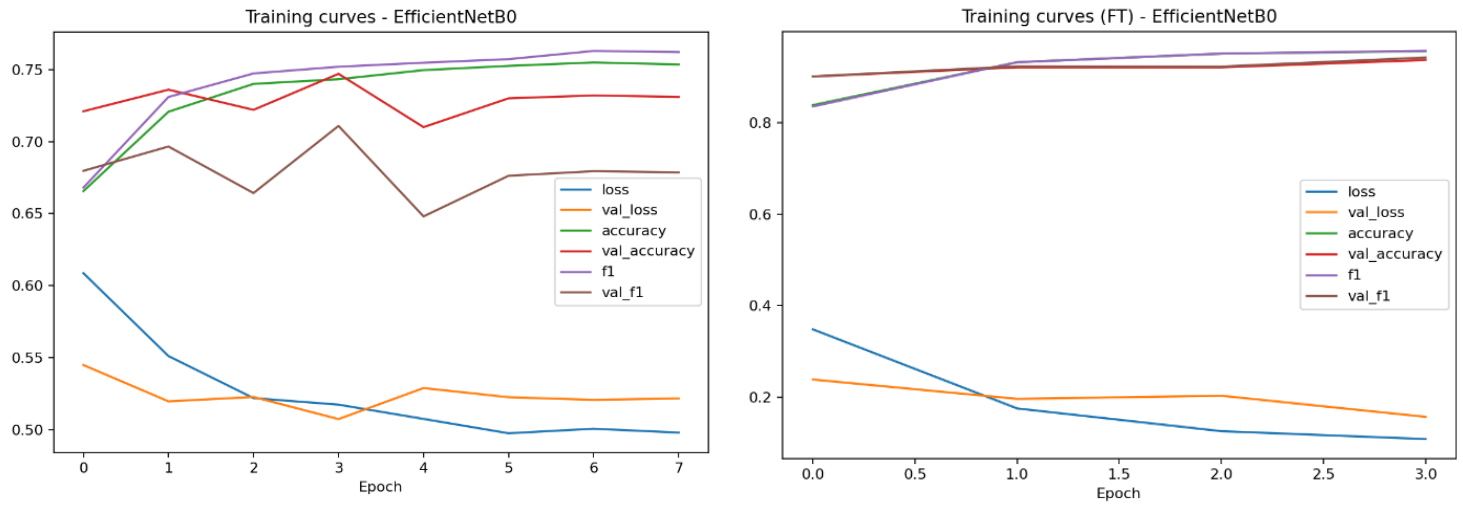

Figure 7: Training Curves of the EfficientNetB0 before after FT

- **ResNet50**

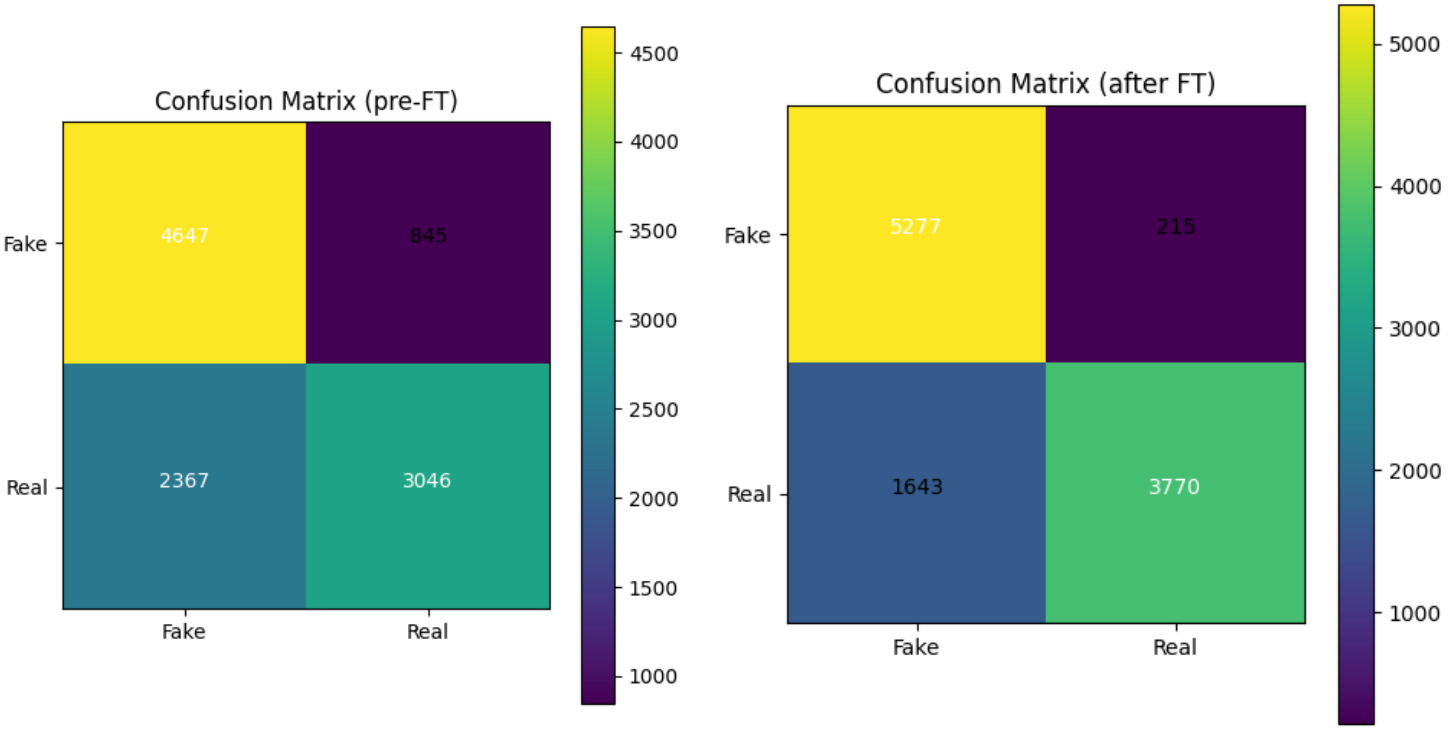

Figure 8 :Confusion matrix of ResNet50 before and after FT

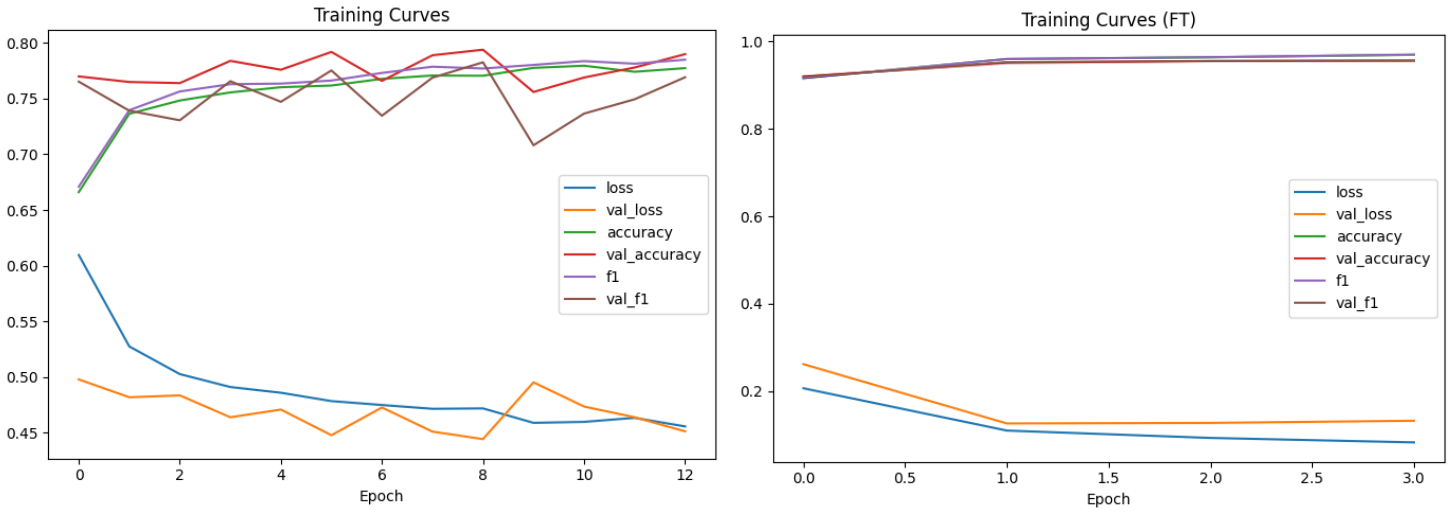

Figure 9: Training Curves of the ResNet50 before after FT

6.4 Interpretation and Insights

The visualizations and quantitative results provide deeper insights into how each model behaved during training and fine-tuning.

- **Xception**

  Xception showed smooth convergence and balanced performance across both classes.
  The confusion matrices before and after fine-tuning reveal a clear improvement:

  - Pre-FT: Higher misclassifications in the Fake → Real direction
  - Post-FT: Significant reduction in errors, especially for the Real class

  The training curves confirm this behavior: validation accuracy and F1-score steadily increased while both training and validation loss consistently decreased. This indicates that Xception generalized well without overfitting, benefiting strongly from fine-tuning its upper layers.

- **EfficientNetB0**

  EfficientNetB0 consistently delivered the best overall accuracy and the most stable generalization.
  The confusion matrix after FT shows:

  - Almost symmetric performance on both Fake and Real classes
  - Fewer extreme misclassifications compared to other models

Training curves reveal:

- Fast convergence
- Stable validation metrics
- No oscillation or divergence

This reflects EfficientNet's compound scaling design, which allows efficient feature extraction even with relatively small input sizes. Its balanced F1-scores demonstrate that it treats both classes fairly and avoids class-specific bias.

- **ResNet50 (v1 and v2)**

  Both versions of ResNet50 were evaluated using different settings (batch sizes, steps per epoch, and number of unfrozen layers). Despite these differences, both versions produced nearly identical results.

  Key observations:

  - ResNet50 achieved the highest Fake-class F1-score (~0.95) of all models
  - The model is extremely sensitive to manipulated content, even before tuning
  - After fine-tuning, the confusion matrix shows a major drop in Fake → Real misclassification (from ~845 errors to 215)

  However:

  - Its Real-class accuracy remained lower than Xception and EfficientNet
  - Training curves display less stability and more oscillation during early epochs
  - This suggests ResNet strongly focuses on Fake cues but struggles with perfect Real vs Fake balance

  This behavior is consistent with its deeper residual blocks, which detect fine-grained artifacts but may slightly underfit natural images.

6.5 Discussion

The comparative analysis of Xception, EfficientNetB0, and both versions of ResNet50 highlights the impact of transfer learning and fine-tuning on deepfake detection performance.

All models improved after fine-tuning, as reflected in their confusion matrices and training curves. Among the three architectures, **EfficientNetB0 delivered the strongest overall performance**, combining the highest accuracy with balanced F1-scores and extremely stable learning dynamics.

Xception also demonstrated excellent generalization. Its pre- and post-fine-tuning confusion matrices clearly show how fine-tuning helped the model correct previous biases toward misclassifying real images. Its smooth training curves support the conclusion that Xception maintains **a robust and stable training process**.

ResNet50, evaluated in two versions, exhibited the largest performance gain after fine-tuning. Both v1 and v2 produced nearly identical outcomes, suggesting that ResNet50's behavior is largely governed by its architecture rather than specific hyperparameter variations. It achieved the strongest Fake-class recall, making it highly effective when prioritizing manipulated-image detection, although its overall accuracy remained slightly lower than that of the other models.

In summary:

- **EfficientNetB0** offers the **best balance, stability, and overall accuracy**.
- **Xception** provides **smooth convergence and reliable generalization**.
- **ResNet50 (both versions)** is **most effective when Fake-image sensitivity is prioritized**, despite lower balance across classes.

These insights demonstrate how architectural differences influence deepfake-detection behavior, and how fine-tuning significantly enhances all models' ability to identify subtle manipulation patterns.

7 Conclusion

This project explored the challenging task of detecting deepfake facial images by leveraging modern deep learning techniques. Using transfer learning, three pretrained CNN architectures—ResNet50, Xception, and EfficientNetB0—were trained and evaluated on a large dataset of real and manipulated facial images. The models were able to learn subtle visual cues such as inconsistencies in texture, lighting, and facial alignment, which are often difficult for humans to recognize. Among the three architectures, EfficientNetB0 demonstrated the strongest and most stable performance, achieving the highest accuracy, precision, recall, and F1-score. Xception also performed well with balanced metrics, while ResNet50 showed comparatively lower accuracy and struggled more with generalization.

Throughout the project, several challenges were encountered. Training on a dataset exceeding 140,000 images required careful resource management, especially when working with limited GPU capacity. Fine-tuning the models also proved sensitive to hyperparameters such as learning rate, augmentation strength, and the number of trainable layers. Additionally, some models experienced overfitting, particularly when augmentation was too strong or the network architecture was more complex. Despite these limitations, the overall results highlight the effectiveness of deep learning in identifying deepfake content when supported by appropriate preprocessing and a structured training pipeline.

There remains significant potential for improvement. Future work can explore more advanced architectures such as Vision Transformers, which may capture global image relationships more effectively and improve detection of subtle manipulation patterns. Another direction involves incorporating frequency-domain features, as many deepfake artifacts become more visible when analyzed beyond the spatial domain. Expanding the system to support video-based deepfake detection would also provide stronger reliability by leveraging temporal consistency. Finally, increasing dataset diversity and evaluating the models against newly emerging manipulation techniques will help enhance the system's robustness as deepfake methods continue to evolve.

8 Contributions

Table 8: Roles and contributions within the project

Student name	Student contributions
Nora Alasmari	Methods, conclusion, ResNet50 model training
Norah Alfaheed	Related Works, Experiment, EfficientNetB0 model training, organizing the document.
Hissah Alotaibi	Dataset, Results and Discussion
Rawan Albatati	introduction, Experiment, Xception model training, ResNet50 model training, Results and Discussion, organizing the document.

9 References

[1] D. K. M. H. Rössler, "FaceForensics++: Learning to Detect Manipulated Facial Images," 2018. [Online]. Available: https://arxiv.org/abs/1803.09179.

[2] X. Yu, "Deepfake Detection with Capsule Networks," July 2019. [Online]. Available: https://arxiv.org/abs/1907.06539. [Accessed September 2025].

[3] W. B. Dolhansky, "The Deepfake Detection Challenge," IEEE, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9299473.

[4] Z. e. al, " Pixel-Level Artifacts from AI-Generated Images, Improving Deepfake Detection Accuracy," 2020. [Online]. Available: https://www.researchgate.net/publication/388330926_Unmasking_AI-Created_Visual_Content_A_Review_of_Generated_Images_and_Deepfake_Detection_Technologies.

[5] M. Karki, "Deepfake and Real Images [Dataset]," 2025. [Online]. Available: https://www.kaggle.com/datasets/manjilkarki/deepfake-and-real-images.