Evaluating CNN Models for Deepfake Detection: A Case Study with ResNet50, MobileNetV2, and EfficientNetV2B0

CSE4006 – DEEP LEARNING

PROJECT REPORT

Class Number – **AP2024254000418**

SLOT – **D1+TD1**

Course Type – **EPJ**

Course Mode – **Project Based Component (Embedded)**

Department of Artificial Intelligence and Machine Learning

School of Computer Science and Engineering

By

22BCE8410	Nikhilesh Rachabattula
22BCE7272	Kanuri Sai Kiran
22BCE9104	P. Dwijesh Reddy

Submitted to:-

Dr. E.Sreenivasa Reddy
Professor-HAG, SCOPE, VIT-AP.

ABSTRACT

Deepfake technology has grown quickly in recent years, mainly because of improvements in artificial intelligence and generative adversarial networks (GANs). These tools can create videos that look very real but are actually fake, which can cause serious problems in areas like news, politics, and online safety. To fight this issue, we explore how deep learning can help detect deepfake videos. In this study, we compare three well-known deep learning models: ResNet50, MobileNetV2, and EfficientNetV2B0, to find out which one works best for deepfake detection.

Our method starts by taking several frames from each video. These frames are resized and cleaned to prepare them for training. We use data augmentation techniques to make the model more accurate and to help it learn better from a smaller dataset. All three models are first loaded with pretrained weights from ImageNet, and then fine-tuned to detect real versus fake videos. We use the same training setup for each model, including early stopping, learning rate control, and mixed precision training on Kaggle's GPU environment.

We test the models using different metrics such as frame-level accuracy, video-level accuracy, precision, recall, F1-score, confusion matrix, and the ROC-AUC score. Based on our experiments, EfficientNetV2B0 gives the best results, with the highest accuracy and ability to generalize across videos. ResNet50 also performs well and offers a good balance between speed and accuracy. MobileNetV2 is the fastest and uses fewer resources, making it suitable for mobile or real-time applications, but it gives slightly lower accuracy.

This comparison shows the strengths and weaknesses of each model in detecting deepfakes. It also helps others choose the right model depending on their needs—whether they want speed, accuracy, or a balance of both. Our work shows that deep learning can be a powerful tool for identifying deepfake videos and protecting people from fake media.

Keywords: Deepfake Detection, Convolutional Neural Networks (CNNs), MobileNetV2 , EfficientNetV2B0, Video Forensics, Media Integrity, Misinformation, Real-Time Detection.

CHAPTER 1

1. Introduction

Deepfake technology has advanced rapidly in recent years, driven primarily by advancements in artificial intelligence (AI) and generative adversarial networks (GANs). These technologies enable the creation of highly realistic videos that, in reality, are entirely fabricated. While this innovation holds potential in entertainment and content creation, it also poses significant risks, especially in areas like news, politics, and online safety, where deep-fake videos can be used maliciously to spread misinformation, manipulate public opinion, and create security threats.

As the prevalence of deepfakes continues to rise, detecting them has become an urgent challenge. Traditional methods of detection, such as manual inspection or simple anomaly detection, are increasingly insufficient due to the increasing sophistication of these videos. Therefore, there is a growing need for automated, AI-driven solutions that can identify deepfakes accurately and efficiently. Deep learning, with its ability to learn complex features from large datasets, offers a promising approach for solving this problem. In particular, convolutional neural networks (CNNs) have shown great potential in detecting visual anomalies in images and videos.

In this study, we explore the application of deep learning models to deepfake detection by comparing three well-established CNN architectures: ResNet50, MobileNetV2, and EfficientNetV2B0. These models have been selected due to their strong performance in various computer vision tasks and their differing trade-offs in terms of accuracy, computational efficiency, and resource usage. ResNet50 is known for its depth and residual connections, which help it learn from deep architectures, while MobileNetV2 is designed for efficiency, making it ideal for resource-constrained environments like mobile devices. EfficientNetV2B0, on the other hand, uses a combination of advanced techniques to achieve high performance while maintaining a relatively small model size.

1.1 Introduction to the Problem

The ability to generate convincing deepfake videos has made it more difficult to distinguish between authentic and fake content. This problem is particularly critical in domains where the integrity of video content is vital, such as journalism, law enforcement, and social media. Detecting deepfakes requires sophisticated algorithms that can analyze visual and temporal inconsistencies in videos, which traditional techniques struggle to identify. In the face of this challenge, CNNs have emerged as a promising tool due to their capacity to learn spatial features and temporal dynamics from video frames.

Despite the success of CNNs in other computer vision tasks, there is still no clear consensus on which architecture is best suited for deepfake detection. The performance of CNN models can vary significantly depending on the task at hand, with some models excelling in accuracy but requiring substantial computational resources, while others are faster but less accurate. This raises the question: which CNN model offers the best trade-off between performance and efficiency for detecting deep-fake videos?

1.2 Motivation

The primary motivation behind this research is to evaluate and compare three well-known CNN models—

ResNet50, MobileNetV2, and EfficientNetV2B0—on their ability to detect deepfake videos. By systematically analyzing each model's strengths and weaknesses, we aim to provide valuable insights for researchers and practitioners working in the field of deep-fake detection. With an increasing need for reliable and efficient solutions, understanding the trade-offs between accuracy, speed, and resource usage is crucial for selecting the most appropriate model for specific use cases.

This research also contributes to the growing body of work aimed at mitigating the impact of deepfakes on society. By leveraging deep learning techniques, we hope to advance the state-of-the-art in deepfake detection and offer a foundation for future improvements and innovations in this area.

1.3 Problem Statement & Objectives

The problem addressed in this study is the lack of a comprehensive comparison between three popular CNN models—ResNet50, MobileNetV2, and EfficientNetV2B0—in the context of deepfake detection. While each model has demonstrated success in other computer vision tasks, their relative performance on deep-fake detection tasks remains underexplored. This study aims to identify which model is best suited for detecting deep-fake videos, based on a variety of evaluation metrics such as accuracy, precision, recall, F1-score, confusion matrix, and ROC-AUC score.

The primary objectives of this research are to perform a comparative analysis of three convolutional neural networks—ResNet50, MobileNetV2, and EfficientNetV2B0—for deepfake video detection, with a focus on the impact of different learning rates and image sizes on their performance. The models will be evaluated based on key metrics such as frame-level accuracy, video-level accuracy, precision, recall, F1-score, and ROC-AUC score. By testing learning rates of 1e-4 and 5e-5, the study will assess the models' ability to converge effectively and avoid overfitting or underfitting, as these learning rates are commonly used to fine-tune pretrained weights. Additionally, all models will be trained with an input image size of 128x128 to ensure consistency and evaluate how each model handles different resolutions while detecting deepfake anomalies in video sequences.

Beyond detection accuracy, the research will also explore the trade-offs between computational efficiency, training time, and model performance. With the use of two different learning rates, the study will analyze how each model's training time and accuracy are affected by these hyperparameters, which are crucial for optimizing the models for practical deployment. EfficientNetV2B0 is expected to perform well in terms of computational efficiency, while ResNet50 may offer a good balance between speed and accuracy. MobileNetV2, known for its lightweight architecture, is expected to demonstrate faster training times, making it ideal for real-time applications but with potentially slightly lower accuracy compared to the other models. This research will help determine the best model for specific operational requirements, such as high-accuracy detection or real-time processing with lower resource consumption.

1.4 Scope

This study focuses on comparing three CNN architectures—ResNet50, MobileNetV2, and EfficientNetV2B0—using a consistent methodology and training setup across all models. The evaluation will be conducted on deepfake video datasets, with performance measured in terms of accuracy, precision, recall, and other classification metrics. While the primary focus is on deepfake detection, the insights from this research can be generalized to other video-based anomaly detection tasks.

This study will not explore the use of other CNN variants or architectures outside the selected models.

Additionally, while the models will be fine-tuned using pretrained weights from ImageNet, the research will not delve into the impact of training from scratch or other transfer learning approaches.

CHAPTER 2

2.1 Literature Survey

Deepfake technology has emerged as a critical challenge in the digital age, leveraging advancements in artificial intelligence to create highly realistic but fabricated content, particularly in videos. These manipulated videos pose significant risks to digital media integrity, often being used for malicious purposes such as misinformation, identity theft, and fraud. Detecting deepfakes is vital to maintaining trust in digital media and mitigating the societal and security risks they pose. Researchers have explored a range of techniques for deepfake detection, with deep learning methods, particularly Convolutional Neural Networks (CNNs), leading the way due to their superior ability to analyze and classify visual data.

Vashistha et al. [1] compared traditional machine learning (ML) methods with deep learning (DL) approaches for deepfake detection. They found that CNNs outperformed ML algorithms like SVM, KNN, and Logistic Regression in terms of accuracy and F1 scores. However, the study highlighted challenges such as poor generalization to new, unseen data and the high computational costs associated with real-time detection. The authors suggested exploring hybrid models to enhance efficiency and accuracy in detecting both video and audio deepfakes. Similarly, Karandikar et al. [2] utilized transfer learning on the VGG-16 model, achieving commendable results for detecting facial manipulations. However, their model struggled with low-quality images and videos, emphasizing the need for improved datasets and temporal analysis to enhance detection performance.

Afchar et al. [3] introduced MesoNet, a lightweight CNN-based model specifically designed for detecting facial video forgeries, such as those generated by Deepfake and Face2Face techniques. The model achieved 98% and 95% accuracy for these two techniques, respectively. The study highlighted the importance of analyzing facial regions, particularly areas prone to manipulation, such as the eyes and mouth, to detect subtle forgery artifacts. Patel et al. [4] proposed a CNN-based architecture for deepfake image detection, achieving 97.2% accuracy across multiple datasets. Their model demonstrated robust generalizability and performance on unseen test sets, with suggestions for future research focusing on video deepfake detection by processing individual frames and applying face detection and cropping techniques.

Hybrid approaches combining spatial and temporal analysis have also shown promise in deepfake detection. Gharde et al. [5] explored Temporal Convolutional Networks (TCNs) for detecting deepfakes across images, audio, and videos. Their model achieved high training accuracies in all modalities, with the study emphasizing the potential of hybrid models to capture both spatial features and temporal dependencies. They suggested integrating transfer learning and ensemble methods to handle the rapidly evolving deepfake generation techniques. Kaushal et al. [6] reviewed various hybrid approaches, including DeepFakeStack and Blur Inconsistency Detection, and advocated for combining these techniques to enhance detection accuracy. These hybrid models are valuable as they can leverage multiple sources of information (spatial and temporal features) to improve detection accuracy, especially in more complex or subtle deepfake manipulations.

Bonettini et al. [7] introduced an ensemble of CNN models based on EfficientNetB4, incorporating attention layers and siamese training to improve model learning and classification accuracy. The study underscored the importance of incorporating temporal information for detecting inconsistencies in video deepfakes. This approach showed that combining multiple models could lead to more robust and accurate results.

The selection of CNN models for deepfake detection, particularly VGG-16, ResNet50, and EfficientNetB4, was based on their strong performance across a variety of computer vision tasks. VGG-16, with its deep architecture and hierarchical feature extraction capabilities, has been widely used for image and video recognition tasks. Its ability to perform well with transfer learning makes it a popular choice for detecting facial manipulations in deep fakes. ResNet50, known for its residual connections that help mitigate the vanishing gradient problem, enables deeper architectures and more accurate feature learning, which is particularly useful for detecting subtle differences in manipulated images or videos. EfficientNetB4, on the other hand, provides a balanced trade-off between model size, speed, and accuracy, making it suitable for real-time deepfake detection tasks. The combination of these three models allows for leveraging their individual strengths—deep feature extraction, accurate learning in deeper architectures, and efficiency in handling large-scale datasets—ensuring high performance in detecting deepfake videos.

The effectiveness of deepfake detection models is largely dependent on the quality and diversity of the datasets used for training and evaluation. Commonly used datasets for deepfake detection include FaceForensics++, Celeb-DF, and the DeepFake Detection Challenge (DFDC) dataset. These datasets consist of both real and manipulated videos, providing a diverse range of scenarios and deepfake techniques for training models.

Rossler et al. [8] introduced FaceForensics++, a comprehensive dataset containing over 1000 manipulated videos created using various deepfake techniques. This data set serves as a benchmark for evaluating detection models and is particularly useful for training models to identify facial manipulations. The study also advocated transfer learning to adapt detection models to emerging deepfake techniques. Adnan et al. [9] utilized the DFDC dataset to develop a CNN-based detection method, achieving 98% accuracy on a test set of real and fake videos. This dataset, which includes diverse real-world scenarios, further highlights the importance of using robust and varied datasets to improve detection performance. The inclusion of diverse datasets allows models to generalize better to unseen data, increasing their robustness in real-world applications.

Kaur et al. [10] discussed the challenges and opportunities in deepfake video detection, highlighting that while deep learning models show promise, they must overcome issues related to dataset bias, real-time detection, and handling new manipulation techniques. The authors emphasized the need for constant model updates and the development of more robust, generalized systems to detect emerging deepfake strategies.

In conclusion, the challenges posed by deepfake technology require advanced solutions to detect manipulated content accurately and efficiently. Deep learning, particularly CNNs, has proven to be an effective tool for deep-fake detection due to its ability to automatically learn complex spatial features from images and videos. While hybrid approaches that combine spatial and temporal analysis have shown potential, the diversity and quality of datasets remain crucial to achieving robust performance. Models such as MesoNet, VGG-16, and EfficientNetB4 have shown great promise in deepfake detection, with CNNs continuing to lead the way in terms of accuracy and generalizability. However, challenges such as poor generalization to new data and high computational costs remain. Future research should focus on improving dataset quality, exploring hybrid models, and enhancing the efficiency of deep-fake detection systems to address these challenges and further improve detection performance.

2.2 Limitations

The papers reviewed in the literature survey contribute valuable insights into deepfake detection, but they also come with certain limitations that hinder their broader application.

Vashistha et al. [1] compared traditional machine learning methods with deep learning approaches for deepfake detection and found that CNNs outperformed traditional models like SVM and KNN. However, one significant limitation of their study is the poor generalization of deepfake detection models to new, unseen data. This remains a challenge due to the rapid advancements in deepfake generation techniques. Moreover, the high computational cost of training deep learning models for real-time detection is another drawback, limiting the feasibility of deploying these models in real-time applications.

Karandikar et al. [2] focused on transfer learning using VGG-16 for deepfake detection but encountered challenges with low-quality images and videos. This model's effectiveness heavily depends on the quality of input data, limiting its performance in real-world scenarios where deepfakes often involve low-quality images and videos. Additionally, their approach lacked temporal analysis, which is essential for detecting manipulations that span across multiple frames in videos.

Afchar et al. [3] introduced MesoNet, a lightweight CNN-based model for detecting facial video forgeries. While it demonstrated high accuracy in detecting facial manipulations, MesoNet's focus on facial regions limited its applicability. The model is specifically designed to identify manipulations in facial features, leaving it vulnerable to deepfakes that affect other areas of the video, such as backgrounds or lighting. This narrow focus reduces its overall generalizability to other types of deepfake manipulations.

Patel et al. [7] proposed a CNN-based architecture for real-time deepfake detection, achieving strong performance across various datasets. However, their method was limited to image-level detection and did not address video-level deepfake detection. Video deepfakes often require temporal analysis to identify subtle manipulations that evolve over multiple frames, and Patel et al.'s model did not consider this crucial aspect.

Gharde et al. [6] explored Temporal Convolutional Networks (TCNs) for detecting deep-fakes across images, audio, and videos. While their model showed promise, it has the limitation of high computational cost. TCNs require substantial data and processing power, making them impractical for real-time applications, especially on devices with limited computational resources. This limitation restricts the scalability of their approach.

Kaushal et al. [8] reviewed hybrid approaches that combine spatial and temporal analysis for deep-fake detection. Although these methods demonstrate potential, they are computationally expensive and complex to implement. The requirement for extensive resources and time to train such models creates a barrier to their use, particularly in real-time settings where detection speed is crucial.

Bonettini et al. [11] introduced an ensemble of CNN models with attention layers and siamese training to improve deep-fake detection accuracy. While the model enhanced robustness, its complexity posed a challenge. The need for multiple models increases the computational burden, making it less practical for real-time use. Furthermore, while the model handled temporal inconsistencies well, it still required large-scale datasets and high computational resources for both training and inference.

In conclusion, while the studies reviewed make significant contributions to deepfake detection, they are hindered by limitations such as poor generalization, reliance on high-quality data, a narrow focus on facial manipulations, and high computational costs. These challenges emphasize the need for more efficient and scalable solutions to improve deepfake detection models for real-world applications.

CHAPTER 3

3.1 Hardware and Software Requirements

3.1.1 Introduction

For this project, we are using a hybrid deepfake detection system that integrates Convolutional Neural Networks (CNNs) and Temporal Convolutional Networks (TCNs). The Deep Fake Detection (DFD) dataset available on Kaggle will serve as the primary data source for model training and evaluation. Given the computational needs of this dataset, we will rely on cloud-based platforms such as Kaggle and Google Colab, which provide necessary GPU support and extensive resources for model development.

3.1.2 Cloud-Based Hardware Considerations

Kaggle Notebooks:

Kaggle provides free access to NVIDIA Tesla T4 GPUs, which are essential for handling the resource-intensive task of training CNNs and TCNs for deepfake detection. The platform also offers up to 29 GB of RAM, making it suitable for processing medium-sized datasets like the DFD dataset. With a maximum session limit of 9 hours for GPU usage, it is important to save model checkpoints frequently to avoid losing training progress. Additionally, Kaggle's 4-core CPUs are suitable for the preprocessing and data preparation tasks required for handling the DFD dataset.

3.1.3 Software Stack Recommendations

The software stack for this project should include both deep learning frameworks and additional libraries for handling video processing and model evaluation. The preferred deep learning framework for this project is PyTorch, due to its flexibility and dynamic computation graph, which is beneficial when implementing a hybrid model that combines CNNs and TCNs. TensorFlow can also be used if team members are more familiar with it. Additional essential libraries include OpenCV for video processing, Dlib for face detection and alignment, and FFmpeg for video format conversions.

We will also require NumPy and Pandas for data manipulation and management, Matplotlib and Seaborn for visualization, and Scikit-learn for model evaluation and performance metrics, such as accuracy, precision, recall, and F1-score.

3.2 Dataset Requirements

For the hybrid deepfake detection system, the dataset used is the Deep Fake Detection (DFD) dataset from Kaggle, which contains both real and fake videos. The dataset is initially stored in two separate folders: one for real videos and another for fake videos.

Dataset Balancing and Splitting Strategy:
Since the dataset consists of an unequal number of real and fake videos, the following strategies were applied to balance and split the data effectively:

Balancing Fake and Real Videos: To ensure balanced training, the number of fake videos was adjusted to match the number of real videos. This step is crucial to avoid any bias in the model, as having more fake videos than real videos could lead to an imbalanced model.

70-30 Train-Test Split: A 70-30 split was applied to both real and fake video datasets separately, ensuring that 70% of the data is used for training and 30% for testing. Specifically:
For real videos: The real_train set consists of 254 videos, and the real_test set contains 110 videos.
For fake videos: Similarly, the fake_train set contains 254 videos, and the fake_test set includes 110 videos.
This train-test split is designed to provide a sufficiently large training set while maintaining an independent testing set for model evaluation.

Final Dataset Configuration:
Training Set: A total of 254 real and 254 fake videos (508 videos in total) are used for training.
Test Set: A total of 110 real and 110 fake videos (220 videos in total) are used for testing.

The dataset is preprocessed by extracting frames at consistent intervals, applying face detection and alignment, and resizing the images to a standard resolution. Additionally, data augmentation techniques such as rotation, flipping, and brightness adjustment are applied to increase the diversity of the dataset and improve the model's generalization.

By balancing the dataset and ensuring the proper train-test split, the model is trained on a diverse set of data, which helps in effectively detecting deep-fake videos.

CHAPTER 4

4. Methodology – 1

The methodology adopted for deepfake detection in this study is structured into two primary phases: (1) the development and training of a convolutional neural network model based on ResNet50, and (2) the evaluation of the model on a curated test dataset comprising real and fake videos. This section outlines the architectural design, data preparation process, training configurations, and evaluation metrics employed in the study.

4.1 Model Architecture and Training Strategy

The core of the deepfake detection system is a customized ResNet50-based Convolutional Neural Network (CNN). A pre-trained ResNet50 model, excluding its top layers (include_top=False), is used as a frozen feature extractor. On top of this, task-specific layers are added: a Global Average Pooling layer, a Dropout layer to reduce overfitting (commonly set to 0.3), and a Dense output layer with a single neuron using sigmoid activation for binary classification. The model is compiled with the Adam optimizer and binary cross-entropy loss, utilizing early stopping and learning rate reduction callbacks for effective training convergence.

Experimental Setup and Configurations

The dataset is split into training and testing sets with 218 real and 460 fake videos for training, and 104 real and 260 fake videos for testing. From each video, 10 uniformly spaced frames are extracted. All frames are resized to three different input resolutions—128×128×3, 64×64×3, and 32×32×3—based on the model configuration being tested. Training is performed over 10 epochs. Multiple experiments were conducted by varying learning rates and image sizes while keeping the dropout fixed at 0.3. The configurations include:

- Learning rate = 0.001 with input size 128×128×3
- Learning rate = 1e-3, 5e-4, and 1e-5 with input size 32×32×3
- Learning rate = 1e-3, 5e-4, and 1e-5 with input size 64×64×3

Test Data Processing and Frame-Level Prediction

Each frame is normalized (pixel values scaled to [0,1]) and labeled as 0 for real and 1 for fake. The model predicts the probability of each frame being fake; frames with a probability ≥ 0.5 are labeled as fake, and those with a probability < 0.5 as real. This per-frame prediction approach enables a fine-grained analysis of the model's ability to detect deepfakes.

Evaluation Metrics

Model performance is evaluated using a variety of metrics including frame-level accuracy, confusion matrix, classification report (precision, recall, F1-score), and ROC-AUC. These metrics collectively provide a comprehensive view of the model's performance across different resolutions and training configurations.

Frame size: 128 x 128

Learning Rate	Accuracy
0.001	66.34

Frame size: 32 x 32

Learning rate	Accuracy
1e-3	71.43
5e-4	71.43
1e-5	63.02

Frame size: 64 x 64

Learning rate	Accuracy
1e-3	71.0
5e-4	71.0
1e-5	71.0

4.2 Method - 2

In this study, a deep learning-based pipeline was developed for deepfake video detection using a ResNet50 backbone. The dataset consisted of 182 real and 184 fake videos for training, and 72 real and 100 fake videos for testing. From each video, 5 representative frames were extracted, resized to 128×128, normalized, and labeled for binary classification. A ResNet50 model pre-trained on ImageNet was used with only the last 10 layers unfrozen for partial fine-tuning. On top of the base model, Global Average Pooling, Batch Normalization, Dropout, and Dense layers were added, concluding with a sigmoid-activated output layer. Multiple experiments were conducted using different configurations: (1) learning rate = 0.01 with dropout = 0.3, (2) learning rate = 0.01 with dropout = 0.2, and (3) learning rate = 0.0001 with dropout = 0.3. The models were trained using the Adam optimizer, binary crossentropy loss, and an 80-20 train-validation split. Optimization techniques like EarlyStopping and ReduceLROnPlateau were applied, and mixed precision with XLA was enabled to improve training performance. Evaluation on the test set was performed using accuracy, confusion matrix, classification report, and ROC-AUC score.

Learning Rate	**Dropout**	**Accuracy**
0.0001	0.3	60.37
0.0001	0.2	51.11

Learning Rate	**Dropout**	**Accuracy**
0.01	0.3	41.09

4.3 Method - 3

Our pipeline begins by extracting 10 equally spaced frames from each video, which are then resized and used for training. The model training is carried out with a batch size of 32, and each configuration is trained for 20 epochs. To explore how input resolution and learning rates affect performance, we conducted experiments using three different input image sizes:

128×128×3, 64×64×3, and 32×32×3, combined with different learning rates.

The first model, implemented in the build_model_v1() function, uses ResNet50 as the base with pretrained ImageNet weights, where all layers are set as trainable to allow fine-tuning. After extracting features using the ResNet50 backbone, the model applies global average pooling followed by batch normalization. To prevent overfitting, dropout layers with rates of 0.3 and 0.4 are added before and after a dense layer of 256 units with ReLU activation and L2 regularization (factor of 0.01). The final output layer uses a single sigmoid neuron to predict the probability of a frame being real or fake. This model is compiled with the Adam optimizer using a learning rate of 1e-4 and binary cross-entropy as the loss function.

The second model, defined in build_model_v2(), also leverages the ResNet50 architecture but adopts a simpler and lighter structure. It reduces the number of dense layer units to 128 and excludes any L2 regularization, relying instead on slightly reduced dropout rates of 0.2 and 0.3 to mitigate overfitting. This model is ideal for scenarios where faster convergence is desired or where the dataset might not be complex enough to require aggressive regularization. It uses a lower learning rate of 5e-5, aiming for more stable updates during training.

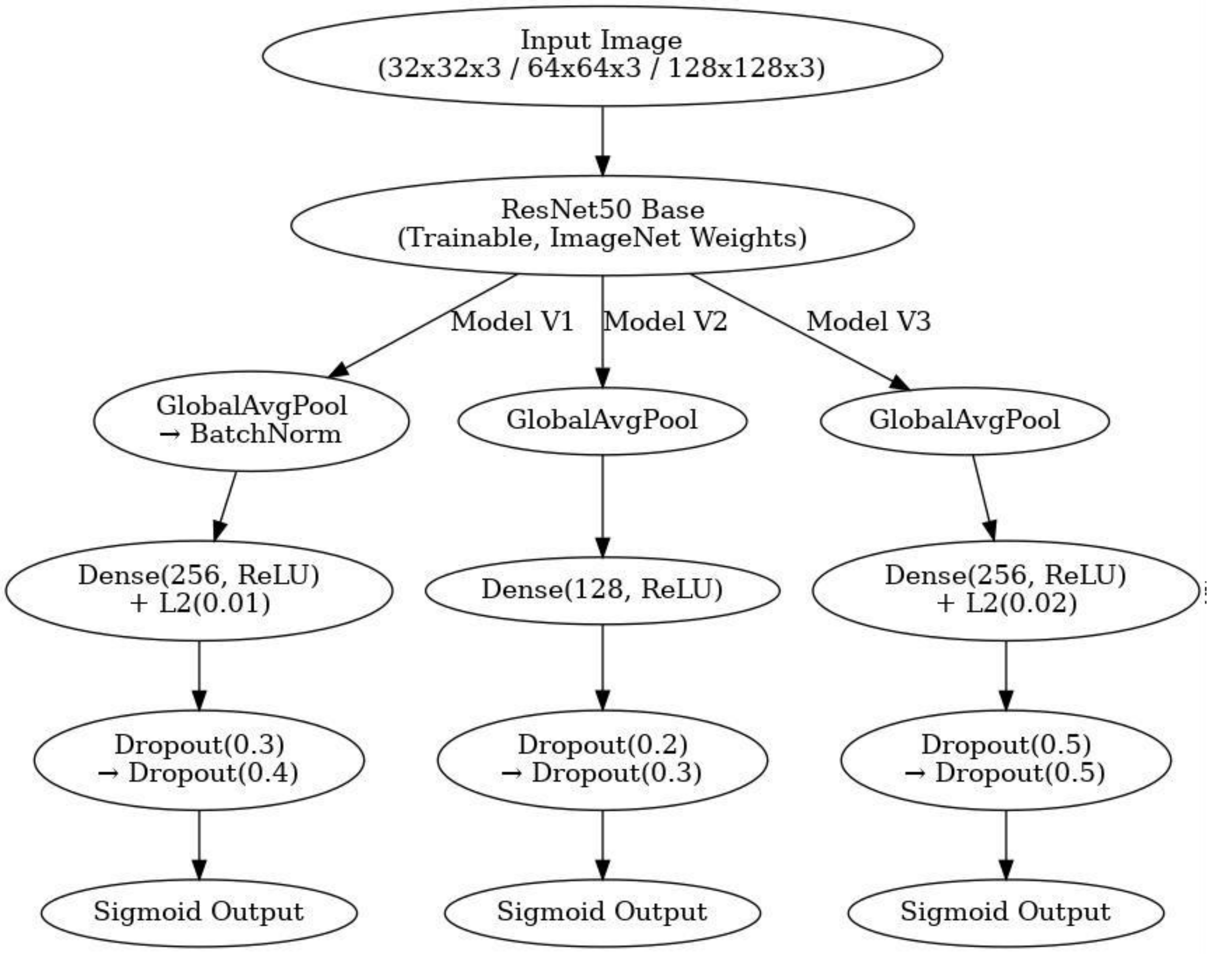
The third model, represented by build_model_v3(), takes a more aggressive approach to regularization. Like the previous versions, it uses a fully trainable ResNet50 base but includes higher dropout rates of 0.5 before and after the dense layer to strongly prevent overfitting. Additionally, the dense layer includes 256 units with L2 regularization of 0.02, further penalizing overly complex weight values. This model is also optimized with the Adam optimizer and a learning rate of 1e-4, suitable for training on datasets with significant variability or potential noise.

To support better generalization across all models, a data augmentation pipeline is applied during preprocessing. This includes random horizontal flipping, rotation, zoom, and brightness adjustments to make the model robust to visual variations.

Evaluation Strategy: To thoroughly evaluate our models, we adopted both frame-level and video-level metrics.

Frame-Level Evaluation: Each frame was independently classified as real or fake. We computed standard metrics: accuracy, precision, recall, and F1-score, with the classification report providing a balanced view of performance, especially under class imbalance. The confusion matrix was used to visualize prediction outcomes, and the ROC-AUC score assessed the model's ability to distinguish between real and fake frames across thresholds.

Video-Level Evaluation: Since deepfake detection ultimately concerns entire videos, we aggregated predictions from all frames using majority voting or average probability to determine the final label. We computed the same set of metrics—accuracy, precision, recall, F1-score, and ROC-AUC—on this video-level prediction. A dedicated confusion matrix was also plotted. This evaluation ensured that our models made consistent and reliable predictions across all frames, better reflecting real-world usage.



CHAPTER 5

5. Methodology – 2

This chapter details the complete workflow developed for the detection of deep-fake videos. It encompasses the sourcing and preparation of data, the design and configuration of multiple convolutional neural network architectures, the training and validation strategies, and the suite of evaluation metrics and analyses used to compare performance. Our goal is to produce a rigorous, reproducible pipeline that yields both frame-level and video-level predictions, and to document every decision and parameter choice along the way.

Data Acquisition and Organization:
The first stage of the project involved assembling a balanced collection of genuine and manipulated video clips. We selected high-quality samples from widely used public repositories, ensuring a diverse mix of subjects, resolutions, compression levels, and background scenes. Genuine ("real") videos depict unaltered human faces speaking or expressing natural motions, while manipulated ("fake") videos include deepfake forgeries produced by state-of-the-art generative algorithms.

To maintain clarity and reproducibility, the dataset was organized into two top-level folders "real" and "fake"—each containing only MP4 files. Filenames were retained from the original sources to preserve traceability, and all videos were catalogued in a master manifest that recorded file paths, durations, frame rates, and resolution metadata. This manifest served as the single point of reference for downstream splitting and processing.

Frame Sampling and Sequence Construction:
Deepfake detection exploits both spatial artifacts (texture irregularities, lighting inconsistencies) and temporal anomalies (unnatural motion patterns). To capture these dimensions, we convert each video into a fixed-length sequence of image frames. Our protocol uniformly samples ten frames per clip by dividing the total frame count into ten equal segments and selecting one representative frame from each segment. Any video containing fewer than ten frames was omitted from analysis to maintain consistency in sequence length.

Each selected frame was initially resized to a modest resolution to reduce I/O overhead during batching. Although future work may incorporate face detection and alignment to crop tightly around facial landmarks, our current framework processes full-frame images. However, face-centric preprocessing remains an optional extension: one can detect and align face bounding boxes to canonical orientations, thereby reducing pose variance and focusing the model on regions most susceptible to deepfake manipulations.

Dataset Splitting and Class Balance:
Maintaining a balanced representation of real and fake samples is critical for preventing classification bias. After loading and sampling frames from all videos, we truncated the larger class so that the counts of fake and real clips were equal. We then performed a stratified split at the video level, not merely on individual frames, to ensure that no video's frames appear in both training and test sets. The primary split allocated 70 percent of videos to training and 30 percent to testing. Optionally, one may further subdivide the training set into training and validation subsets (e.g., 85 percent/15 percent), yielding an overall 59.5 percent/10.5 percent/30 percent distribution.

We fixed random-seed values for all sampling operations, guaranteeing that dataset partitions could be precisely reproduced in subsequent experiments. A detailed roster of training, validation, and test video filenames was saved alongside random-state metadata.

Data Normalization and Augmentation:
To promote stable and efficient neural network training, all pixel intensities were scaled to the [0, 1] range by dividing raw values by 255.0. Data augmentation was applied exclusively to the training set to enhance generalization. The augmentation regimen included random horizontal flipping, small rotations, slight zooming, and brightness jittering. These transformations simulate variations in viewpoint, camera motion, and lighting, forcing the model to learn robust representations rather than memorizing superficial patterns.

Augmentation was performed on-the-fly during training, rather than permanently altering the stored dataset. This approach preserves the original frames for unbiased evaluation while providing fresh, varied examples at every epoch.

Model Architectures
Our detection framework employs a family of convolutional neural network models built upon the EfficientNetV2-B0 backbone. EfficientNetV2-B0 was chosen for its efficient compound scaling strategy, which balances network depth, width, and resolution to achieve strong accuracy with moderate computational cost. Each variant begins with the frozen pre-trained backbone, which is subsequently unfrozen for fine-tuning throughout the experiment.

Three architectural variants were defined, differing only in their dropout rates and the presence or absence of L2 weight regularization within the classification head:

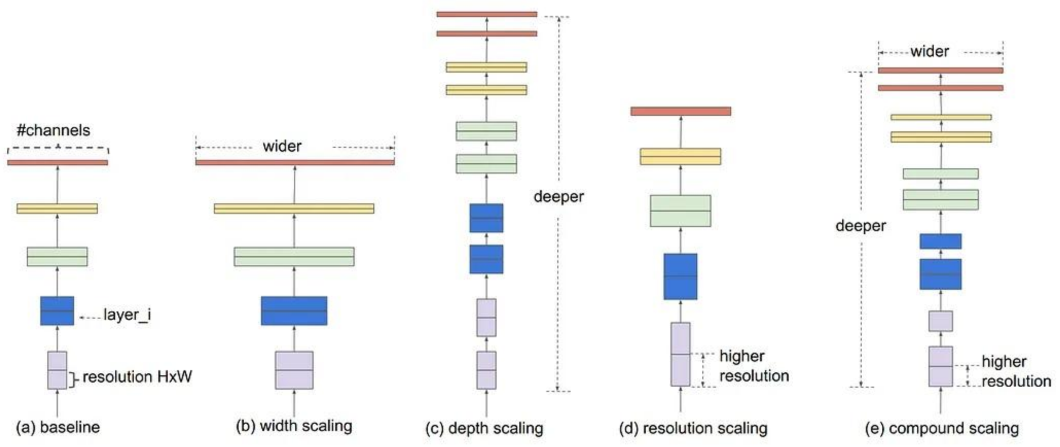


Fig: EfficientNetV2-B0

Model V1 (Moderate Regularization)
This variant uses moderately aggressive dropout layers surrounding a fully connected layer with 256 units. A small L2 penalty is applied to that layer's weights to encourage smaller parameter magnitudes.

Model V2 (Light Regularization)
This configuration employs lower dropout rates and omits L2 weight decay entirely. It features a more compact dense layer of 128 units.

Model V3 (Heavy Regularization)
Here, high dropout rates and a stronger L2 penalty are combined with a 256-unit dense layer, representing the most conservative approach toward overfitting.

All three models terminate in a single sigmoid-activated output neuron, yielding a probability estimate of a frame being fake.

Training Strategy
Each model is trained to minimize binary cross-entropy loss, using the Adam optimizer with a small initial learning rate. To accelerate training, mixed-precision computation was enabled, allowing portions of the network to run in lower precision without sacrificing model fidelity. Just-In-Time (JIT) compilation was also activated to fuse operations and further improve GPU throughput.

Our training regimen extends up to twenty epochs, though an early-stopping mechanism halts training if the loss does not improve for seven consecutive epochs. The best-performing weights are automatically restored upon termination. Although validation-loss monitoring is generally preferable, loss-based stopping on the training set was adopted here to maintain conceptual simplicity; in practice, one would likely monitor validation loss once a separate validation set is established.

Evaluation Protocol:
Evaluation occurs at two levels:

Frame-Level Assessment:
Each trained model infers a probability for every frame in the test set. We apply a threshold of 0.5 to produce binary predictions. Standard classification metrics are computed, including accuracy, precision, recall, F1-score, and area under the ROC curve (AUC). Confusion matrices at the frame level highlight the

distribution of true/false positives and negatives.

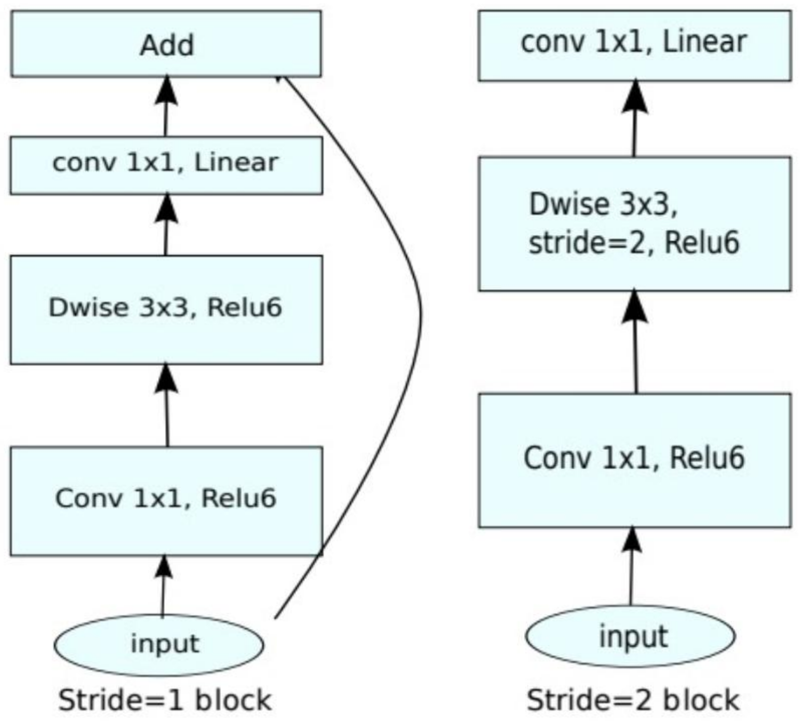


Fig: MobileNetV2

Video-Level Aggregation
Recognizing that real-world applications classify entire video clips; we group frame probabilities by their originating video and compute the mean probability across all ten frames. Applying the same 0.5 threshold yields a single predicted label per video. We then compute video-level accuracy, loss, ROC AUC, and confusion matrices. This two-tiered evaluation elucidates whether strong frame performance translates to robust video detection.

All evaluation metrics and intermediate results are logged. In addition, training history (loss and accuracy curves) is preserved to facilitate analysis of convergence behavior and potential overfitting.

Comparative Analysis and Visualization
To compare the three model variants, we compile their frame-level and video-level metrics into a summary table sorted by video accuracy. We then produce a series of visualizations:

Bar charts contrasting frame vs. video accuracy for each model

Comparative bar charts of frame vs. video loss

AUC comparison bars

Training curves overlaying all variants' accuracy and loss trajectories over epochs

These visual tools provide insights into how dropout and regularization influence both convergence speed and generalization to unseen videos.

Implementation Details and Reproducibility
The entire pipeline was implemented in TensorFlow 2.x using the Keras API, with OpenCV and NumPy for data handling, and scikit-learn for metric computations. We set fixed seeds in Python's random module, NumPy, and TensorFlow to ensure deterministic behavior. All package versions, training hyperparameters, and dataset splits were recorded in a configuration file, guaranteeing that experiments can be rerun precisely.

Future Work and Extensions:
Although the current methodology establishes a solid baseline, several enhancements are foreseen:

Face Detection and Alignment: Focusing on facial regions promises to improve sensitivity to local manipulations.

Temporal Modeling: Incorporating a dedicated temporal network (e.g., a Temporal Convolutional Network) on per-frame embeddings could better capture motion inconsistencies.

Model Ensembling: Averaging or voting among multiple trained variants may boost robustness.

Efficient I/O: Leveraging specialized video loaders and caching mechanisms can reduce data-loading bottlenecks.

On-Device Deployment: Techniques such as quantization and pruning would enable real-time detection on mobile or embedded platforms.

CHAPTER 6

6. Results:

6.1 Results for Methodology – 1

6.1.1 Metrics for Methodology - 1

Frame Size: 128 x 128

Learning Rate: 1e-4,5e-5, 1e-4

	Model-1(1e-4)	**Model-2(5e-5)**	**Model-3(1e-4)**
Frame Level Accuracy	89.0909	91.3182	51.5909
Video Level Accuracy	89.5455	92.2727	51.8182

Learning Rate: 1e-6,1e-6,1e-6

	Model-1(1e-6)	**Model-2(1e-6)**	**Model-3(1e-6)**
Frame Level Accuracy	54.2727	54.0909	47.3636
Frame Loss	0.938331	0.777166	0.754459
Video Level Accuracy	55.4545	54.5455	47.7273
Video Loss	0.811571	0.694213	0.752477

Learning Rate: 5e-5, 5e-5

	Model-1(5e-5)	**Model-2(5e-5)**
Frame Level Accuracy	90.5909	89.7727
Frame Loss	0.401680	0.408599
Video Level Accuracy	90.9091	90.9091
Video Loss	0.336524	0.348676

Learning Rate: 1e-3, 1e-3

	Model-1(1e-3)	**Model-2(1e-3)**
Frame Level Accuracy	50.00	50.00
Frame Loss	0.700806	0.693428
Video Level Accuracy	50.00	50.00
Video Loss	0.700806	0.693428

Frame Size: 64 x 64

Learning rate: 1e-3, 1e-3, 1e-3

	Model-1(1e-3)	**Model-2(1e-3)**	**Model-3(1e-3)**
Frame Level Accuracy	50.00	50.00	50.00
Frame Loss	0.693431	0.693201	8.059049
Video Level Accuracy	50.00	50.00	50.00
Video Loss	0.693431	0.694201	8.059048

Learning rate: 5e-4, 5e-4, 5e-4

	Model-1(5e-4)	**Model-2(5e-4)**	**Model-3(5e-4)**
Frame Level Accuracy	52.7727	50.00	50.00
Frame Loss	0.661747	0.693158	0.694881
Video Level Accuracy	51.3636	50.00	50.00
Video Loss	0.659420	0.693158	0.694880

Learning rate: 1e-5, 1e-5, 1e-5

	Model-1(1e-5)	**Model-2(1e-5)**	**Model-3(1e-5)**
Frame Level Accuracy	64.6818	63.0909	49.9091
Frame Loss	0.643989	0.752305	0.77328
Video Level Accuracy	70.4545	66.3636	50.9091
Video Loss	0.556671	0.613948	0.770642

Frame size: 32 x 32

Learning rate: 1e-4, 5e-5, 5e-5

	Model-1(1e-4)	**Model-2(5e-5)**	**Model-3(5e-5)**
Frame Level Accuracy	61.6818	57.2273	50.3182
Frame Loss	0.644382	0.714810	0.714810
Video Level Accuracy	62.2727	60.00	50.00
Video Loss	0.636678	0.661457	0.713760

6.1.2 Graphs for Methodology – 1

Model – 1:

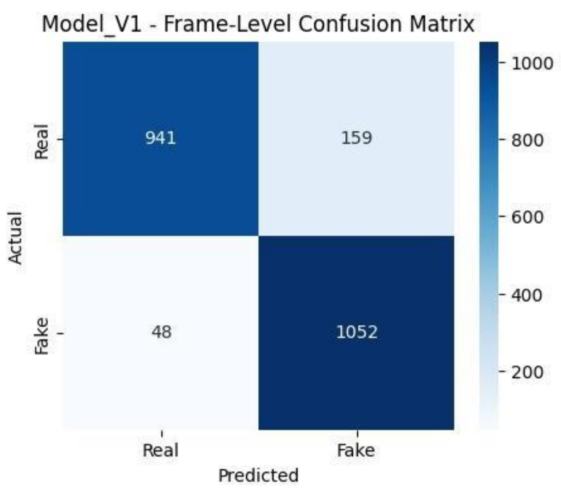


Fig: Frame Level confusion Matrix

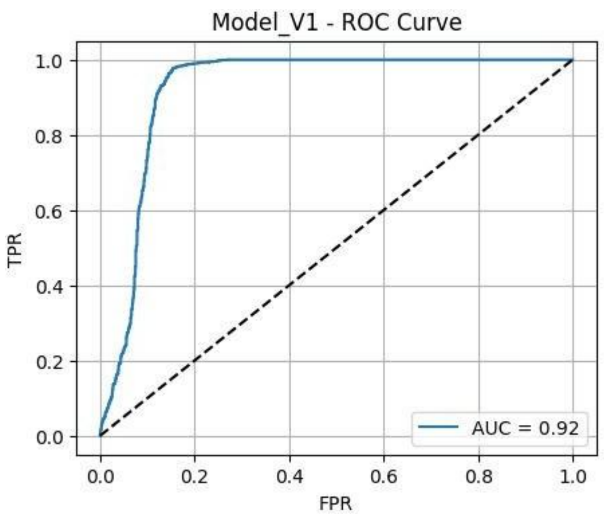


Fig: Frame level Roc Curve

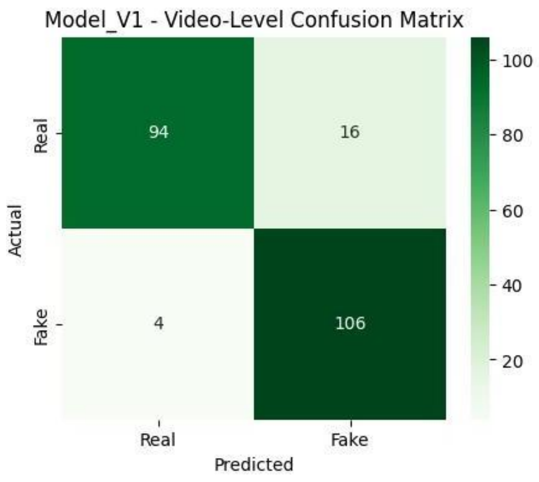


Fig: Video Level Confusion Matrix

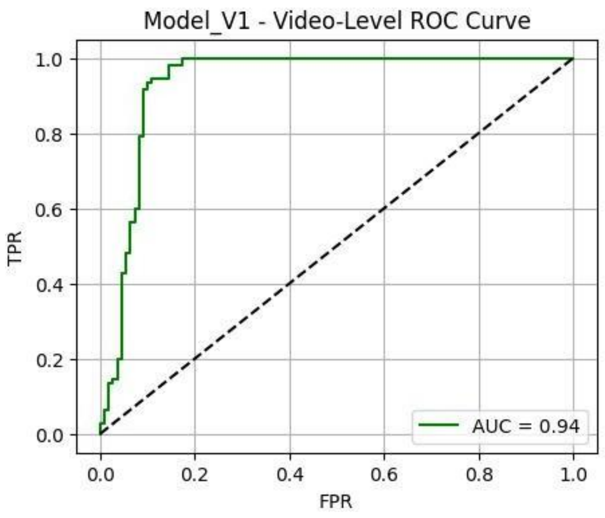
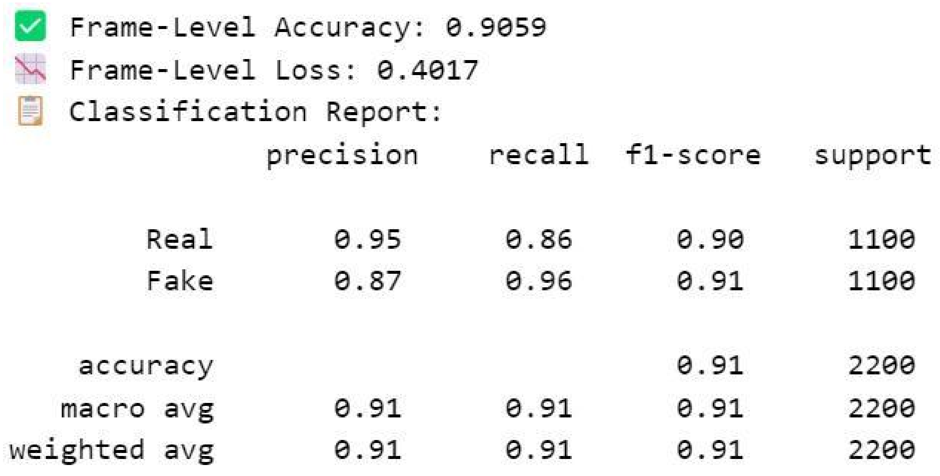


Fig: Video Level ROC Curve

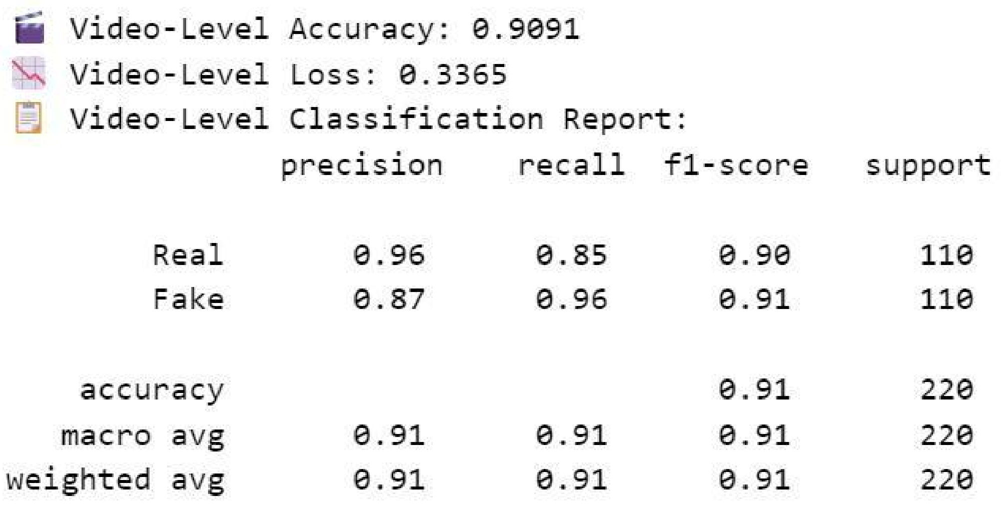


```
✅ Frame-Level Accuracy: 0.9059
📉 Frame-Level Loss: 0.4017
📋 Classification Report:
              precision    recall  f1-score   support

        Real       0.95      0.86      0.90      1100
        Fake       0.87      0.96      0.91      1100

    accuracy                           0.91      2200
   macro avg       0.91      0.91      0.91      2200
weighted avg       0.91      0.91      0.91      2200
```

Fig: Frame Level Results for Model - 1



```
🎬 Video-Level Accuracy: 0.9091
📉 Video-Level Loss: 0.3365
📋 Video-Level Classification Report:
              precision    recall  f1-score   support

        Real       0.96      0.85      0.90       110
        Fake       0.87      0.96      0.91       110

    accuracy                           0.91       220
   macro avg       0.91      0.91      0.91       220
weighted avg       0.91      0.91      0.91       220
```

Fig: Video Level Results for Model - 1

Model – 2:

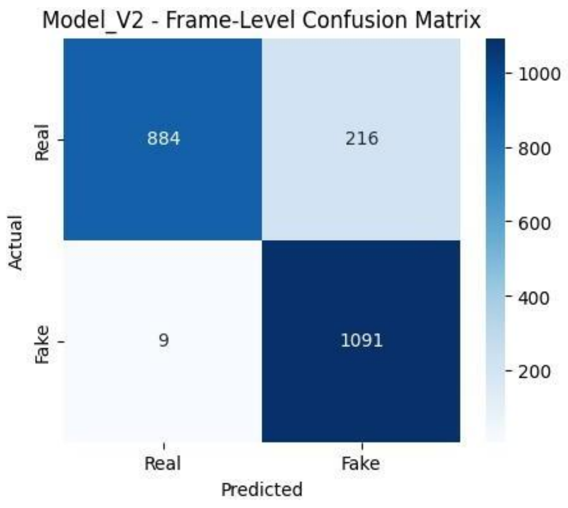


Fig: Frame level Confusion Matrix

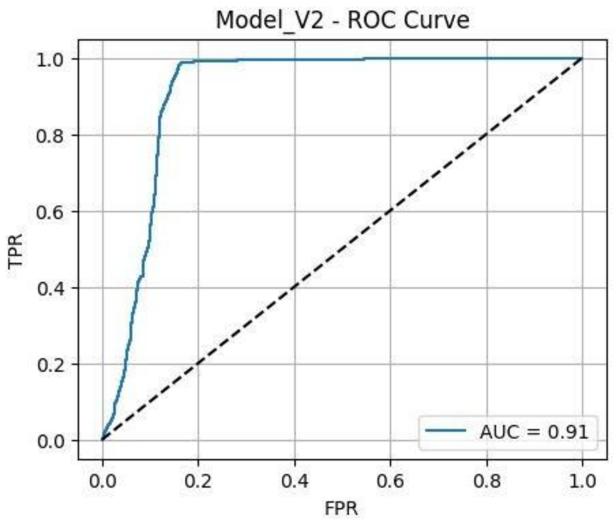


Fig: Frame level ROC Curve

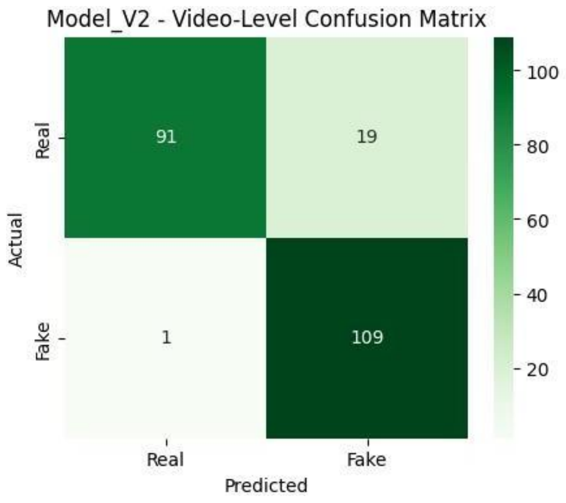


Fig: Video Level Confusion Matrix

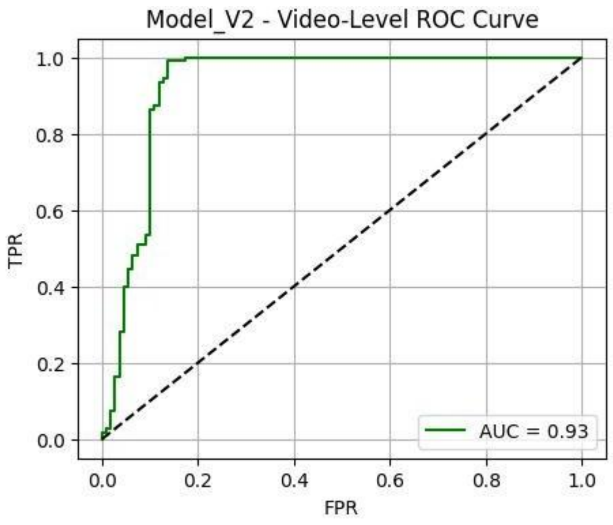
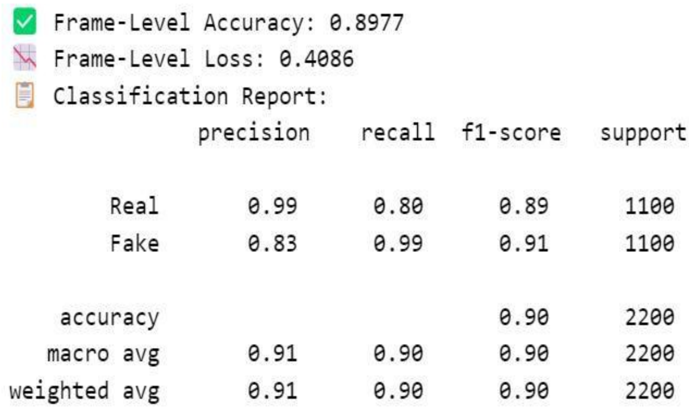


Fig: Video Level ROC Curve

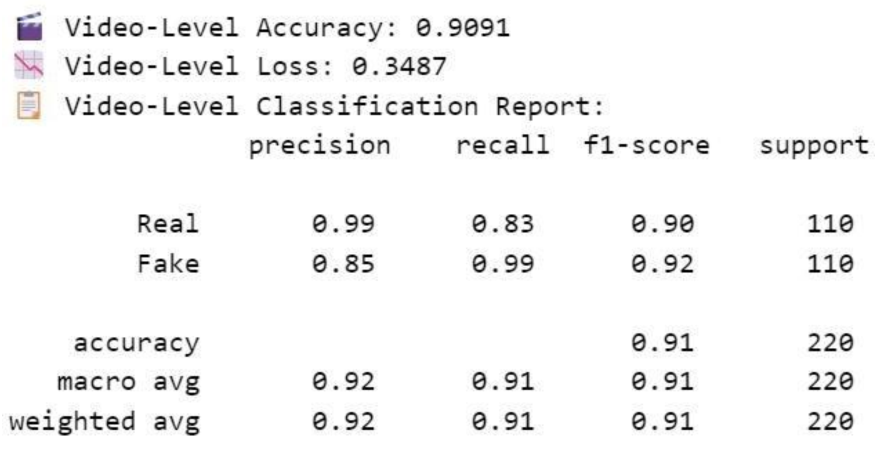


```
✅ Frame-Level Accuracy: 0.8977
📉 Frame-Level Loss: 0.4086
📋 Classification Report:
              precision    recall  f1-score   support

        Real       0.99      0.80      0.89      1100
        Fake       0.83      0.99      0.91      1100

    accuracy                           0.90      2200
   macro avg       0.91      0.90      0.90      2200
weighted avg       0.91      0.90      0.90      2200
```

Fig: Frame Level Results for Model - 2



```
🎬 Video-Level Accuracy: 0.9091
📉 Video-Level Loss: 0.3487
📋 Video-Level Classification Report:
              precision    recall  f1-score   support

        Real       0.99      0.83      0.90       110
        Fake       0.85      0.99      0.92       110

    accuracy                           0.91       220
   macro avg       0.92      0.91      0.91       220
weighted avg       0.92      0.91      0.91       220
```

Fig: Video Level Results for Model - 2

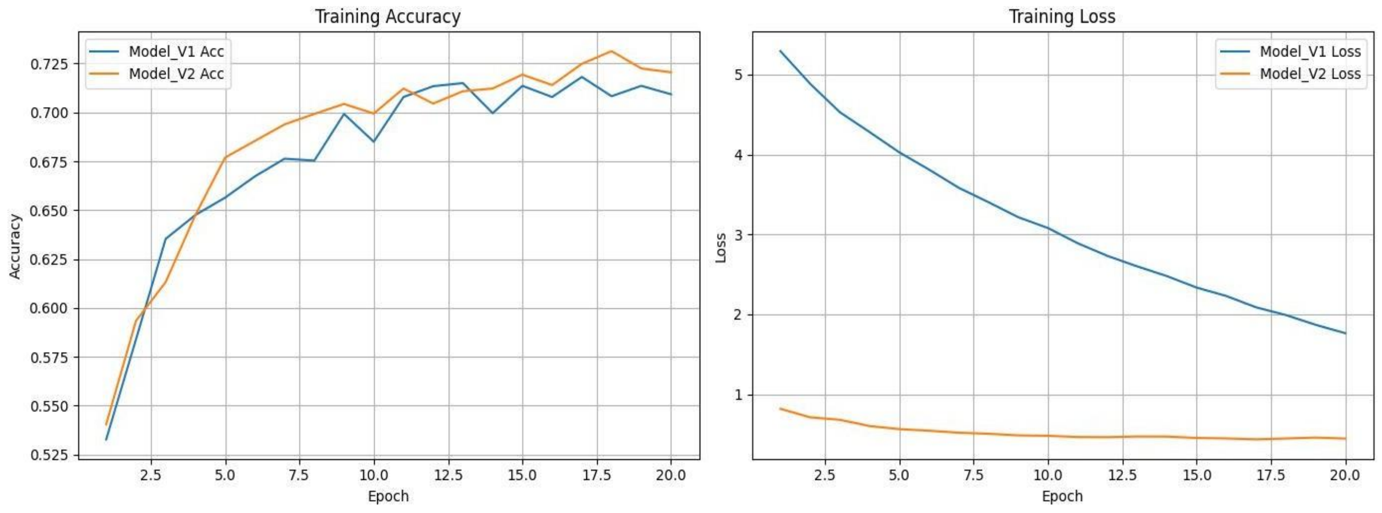


Fig: Training Accuracy and Training Loss Comparison of Model – 1, Model - 2

6.2 Results for Methodology – 2

6.2.1 Metrics for Efficient Net

Frame Size: 128 x 128

Learning rate: 1e-4,1e-4,1e-4

	Model_V1(1e-4)	**Model_V2(1e-4)**	**Model_V3(1e-4)**
Frame Level Accuracy	56.61	58.64	53.39
Frame Loss	1.0301	1.3692	0.6900
Video Level Accuracy	57.63	59.32	54.24
Video Loss	0.8945	1.0203	0.6890

Frame Size: 128 x 128

Learning rate: 5e-5,5e-5,5e-5

	Model_V1(1e-4)	**Model_V2(1e-4)**	**Model_V3(1e-4)**
Frame Level Accuracy	50.85	59.66	52.71
Frame Loss	1.4020	1.2083	0.7123
Video Level Accuracy	50.85	61.02	52.54
Video Loss	1.2430	0.9432	0.7091

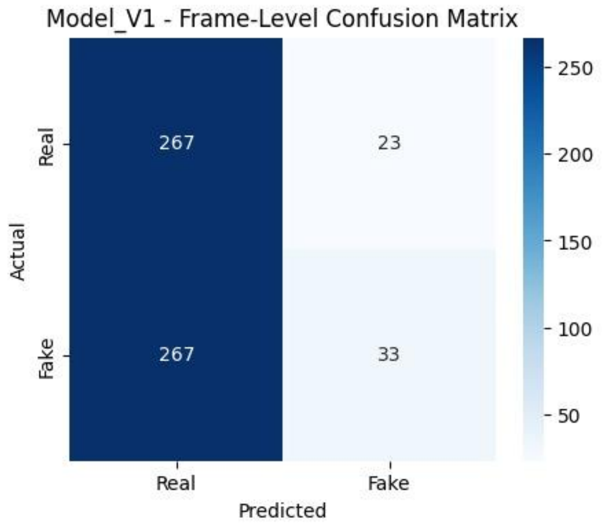


Fig: Frame Level confusion Matrix for Model_V1

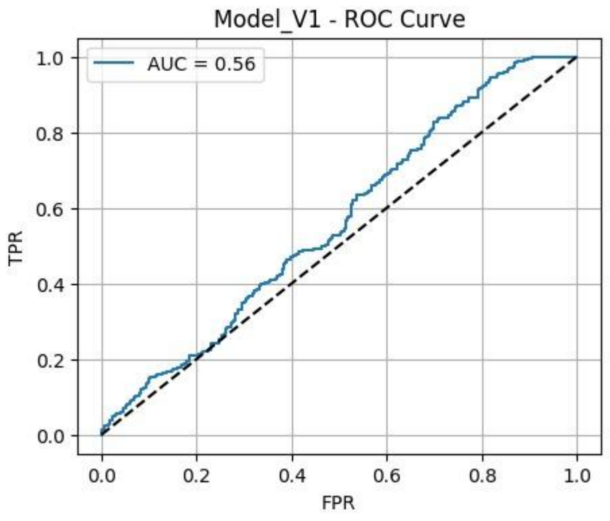


Fig: Frame Level ROC Curve for Model_V1

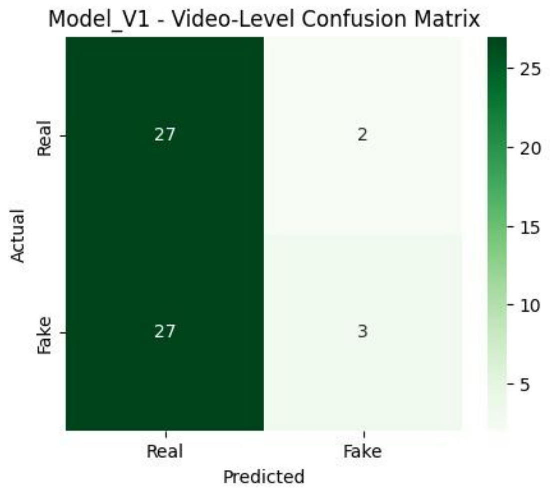


Fig: Video Level confusion Matrix for Model_V1

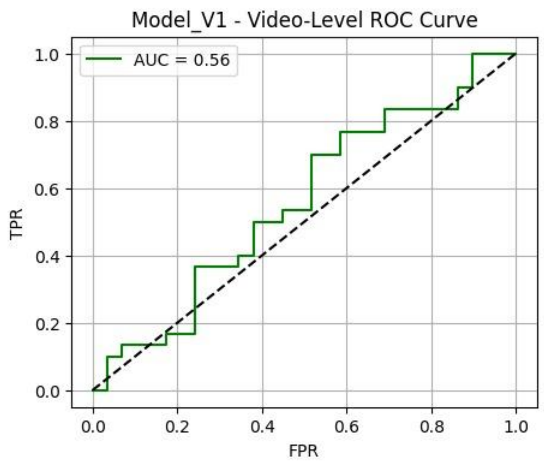
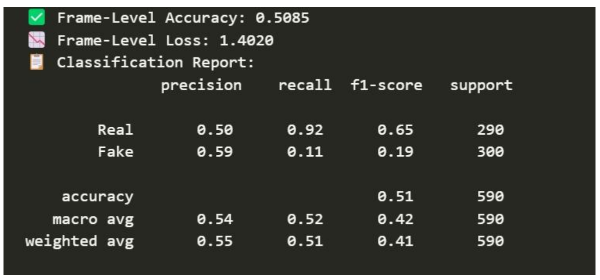


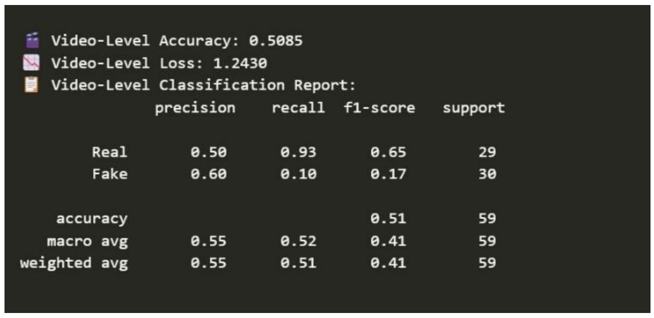
Fig: Video Level ROC Curve for Model_V1



```
✅ Frame-Level Accuracy: 0.5085
📉 Frame-Level Loss: 1.4020
📋 Classification Report:
              precision    recall  f1-score   support

        Real       0.50      0.92      0.65       290
        Fake       0.59      0.11      0.19       300

    accuracy                           0.51       590
   macro avg       0.54      0.52      0.42       590
weighted avg       0.55      0.51      0.41       590
```

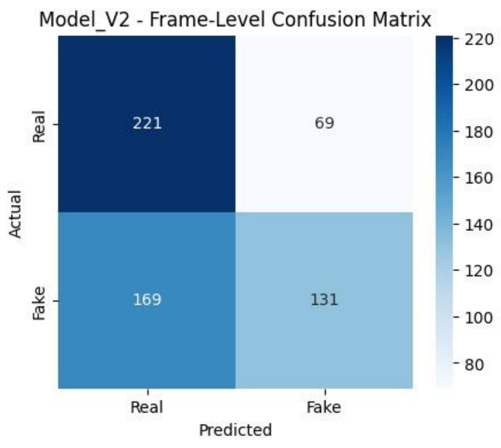


```
🎬 Video-Level Accuracy: 0.5085
📉 Video-Level Loss: 1.2430
📋 Video-Level Classification Report:
              precision    recall  f1-score   support

        Real       0.50      0.93      0.65        29
        Fake       0.60      0.10      0.17        30

    accuracy                           0.51        59
   macro avg       0.55      0.52      0.41        59
weighted avg       0.55      0.51      0.41        59
```

Fig: Frame Level Results for Model_V1 **Fig:** Video Level Results for Model_V1



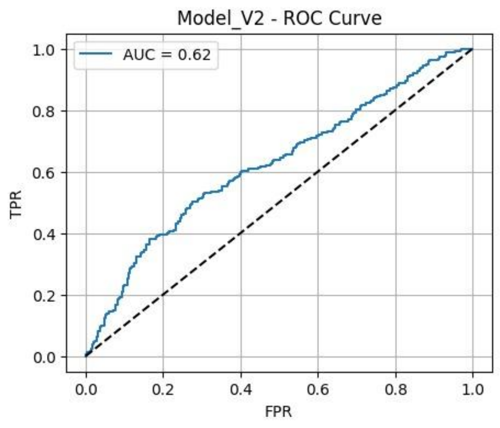
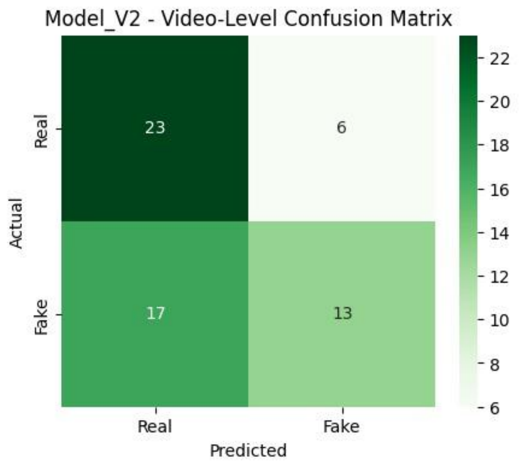


Fig: Frame Level confusion Matrix for Model_V2 **Fig:** Frame Level ROC Curve for Model_V2



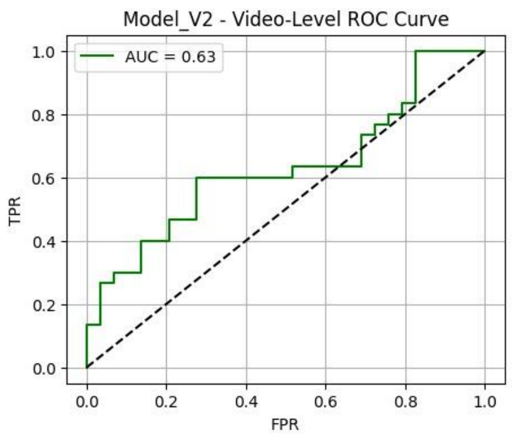
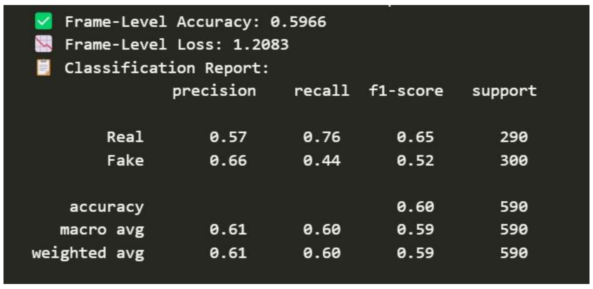


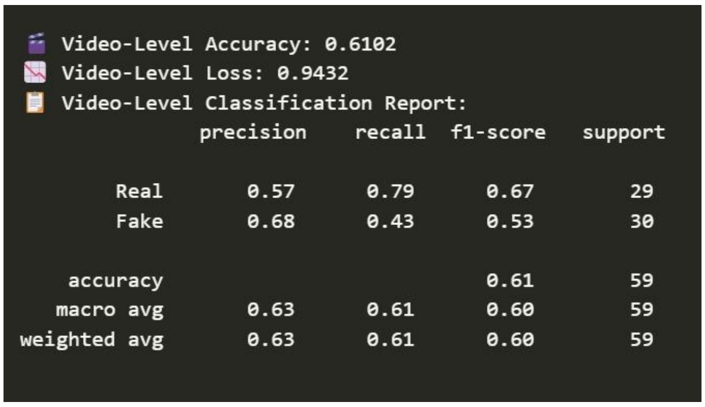
Fig: Video Level confusion Matrix for Model_V2 **Fig:** Video Level ROC Curve for Model_V2



```
✅ Frame-Level Accuracy: 0.5966
📉 Frame-Level Loss: 1.2083
📋 Classification Report:
              precision    recall  f1-score   support

        Real       0.57      0.76      0.65       290
        Fake       0.66      0.44      0.52       300

    accuracy                           0.60       590
   macro avg       0.61      0.60      0.59       590
weighted avg       0.61      0.60      0.59       590
```

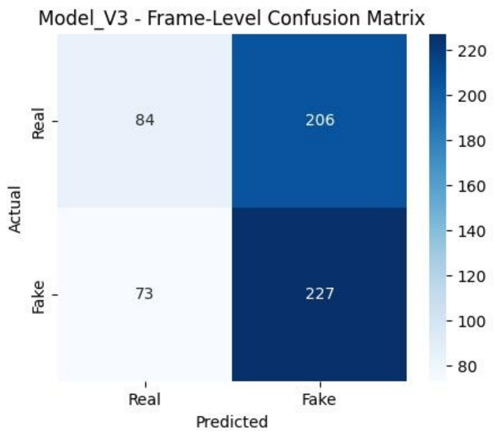


```
🎬 Video-Level Accuracy: 0.6102
📉 Video-Level Loss: 0.9432
📋 Video-Level Classification Report:
              precision    recall  f1-score   support

        Real       0.57      0.79      0.67        29
        Fake       0.68      0.43      0.53        30

    accuracy                           0.61        59
   macro avg       0.63      0.61      0.60        59
weighted avg       0.63      0.61      0.60        59
```

Fig: Frame Level Results for Model_V2 **Fig:** Video Level Results for Model_V2



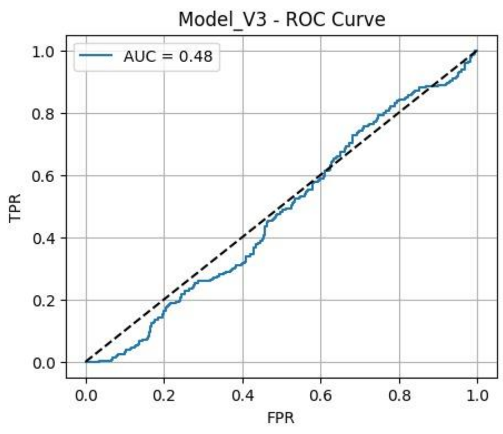
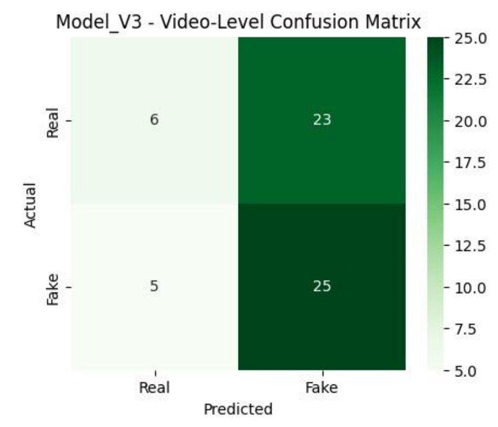


Fig: Frame Level confusion Matrix for Model_V3 **Fig:** Frame Level ROC Curve for Model_V3



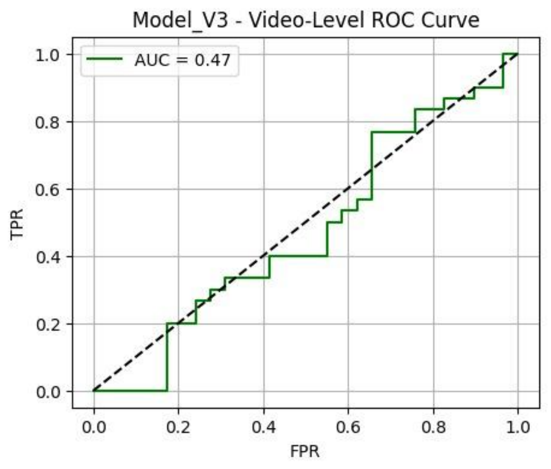
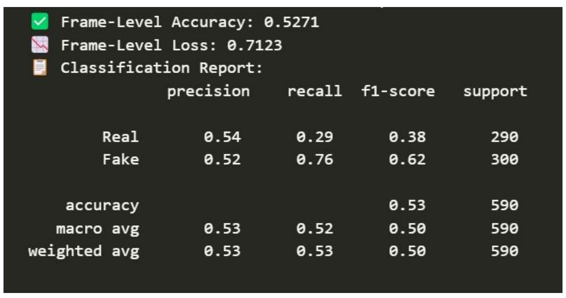


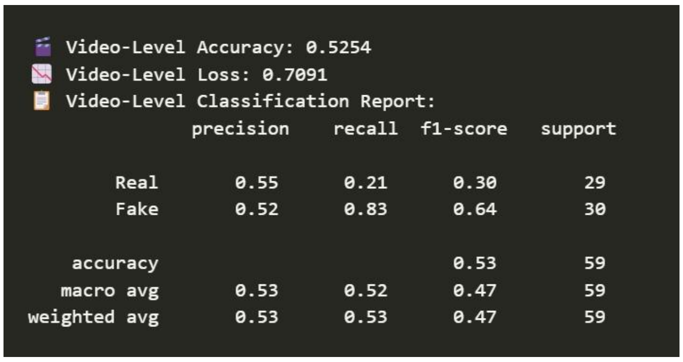
Fig: Video Level confusion Matrix for Model_V3 **Fig:** Video Level ROC Curve for Model_V3



```
✅ Frame-Level Accuracy: 0.5271
📉 Frame-Level Loss: 0.7123
📋 Classification Report:
              precision    recall  f1-score   support

        Real       0.54      0.29      0.38       290
        Fake       0.52      0.76      0.62       300

    accuracy                           0.53       590
   macro avg       0.53      0.52      0.50       590
weighted avg       0.53      0.53      0.50       590
```



```
🎬 Video-Level Accuracy: 0.5254
📉 Video-Level Loss: 0.7091
📋 Video-Level Classification Report:
              precision    recall  f1-score   support

        Real       0.55      0.21      0.30        29
        Fake       0.52      0.83      0.64        30

    accuracy                           0.53        59
   macro avg       0.53      0.52      0.47        59
weighted avg       0.53      0.53      0.47        59
```

Fig: Frame Level Results for Model_V3 **Fig:** Video Level Results for Model_V3

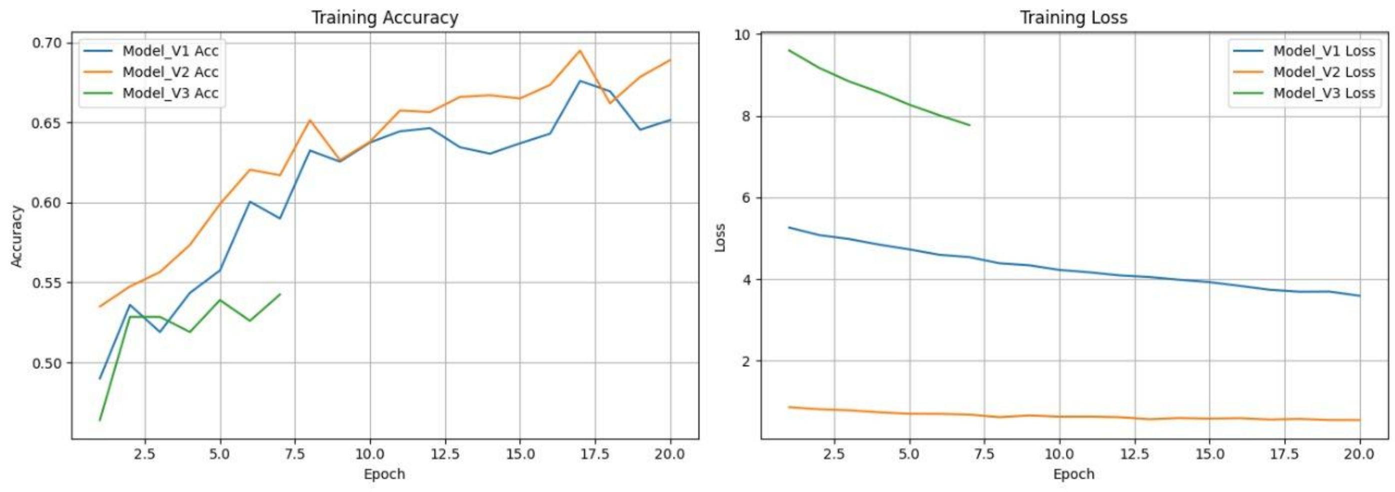


Fig: Training Accuracy and Training Loss Comparison of Model_V1, Model_V2, Model_V3

6.2.2 Metrics for Mobile Net

Frame Size: 128 x 128

Learning rate: 1e-4,1e-4,1e-4

	Model_V1(1e-4)	**Model_V2(1e-4)**	**Model_V3(1e-4)**
Frame Level Accuracy	88.95	88.14	58.09
Frame Loss	0.3135	0.3482	0.7221
Video Level Accuracy	90.91	88.95	56.82
Video Loss	0.2748	0.3115	0.6978

Frame Size: 128 x 128

Learning rate: 5e-5,5e-5,5e-5

	Model_V1(5e-5)	**Model_V2(5e-5)**	**Model_V3(5e-5)**
Frame Level Accuracy	87.50	87.23	52.50
Frame Loss	0.3477	0.3842	0.7496
Video Level Accuracy	89.55	89.09	50.91
Video Loss	0.2878	0.3320	0.7205

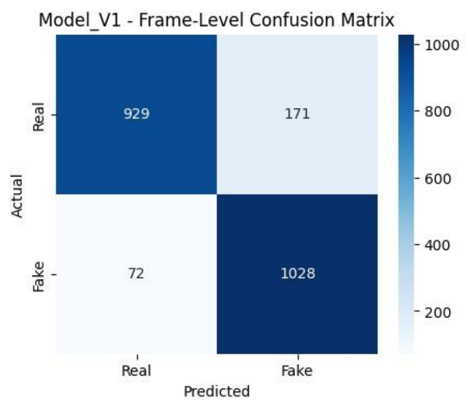


Fig: Frame Level confusion Matrix for Model_V1

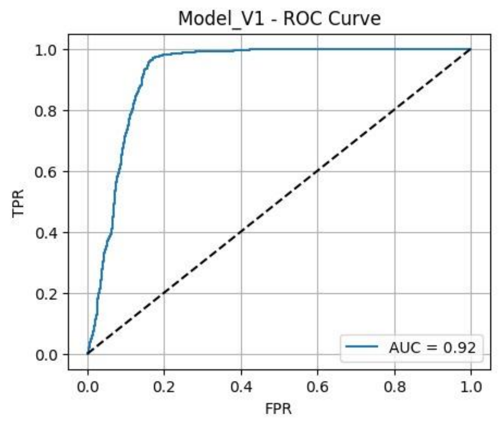


Fig: Frame Level ROC Curve for Model_V1

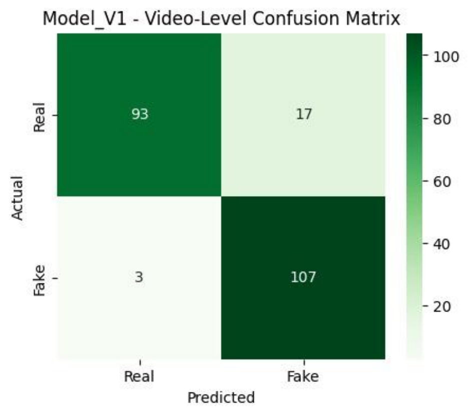


Fig: Video Level confusion Matrix for Model_V1

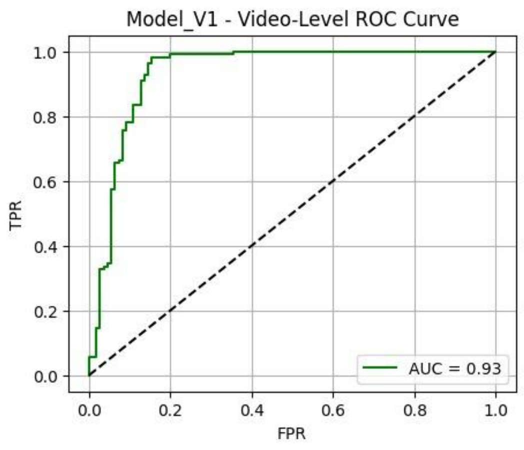
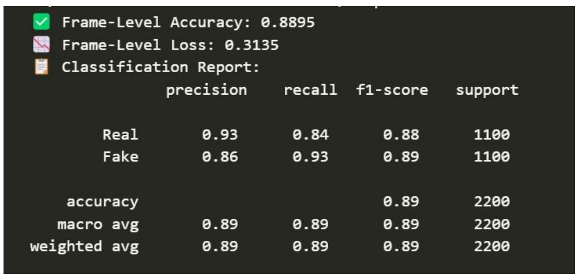


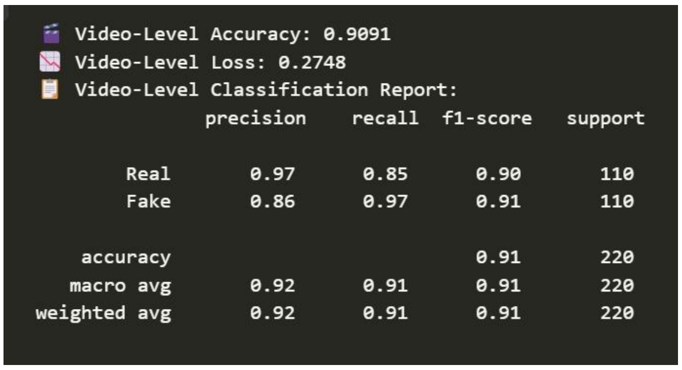
Fig: Video Level ROC Curve for Model_V1



```
✅ Frame-Level Accuracy: 0.8895
📉 Frame-Level Loss: 0.3135
📋 Classification Report:
              precision    recall  f1-score   support

        Real       0.93      0.84      0.88      1100
        Fake       0.86      0.93      0.89      1100

    accuracy                           0.89      2200
   macro avg       0.89      0.89      0.89      2200
weighted avg       0.89      0.89      0.89      2200
```



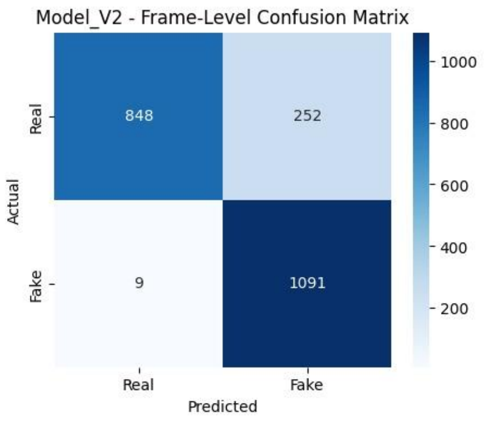
```
🎬 Video-Level Accuracy: 0.9091
📉 Video-Level Loss: 0.2748
📋 Video-Level Classification Report:
              precision    recall  f1-score   support

        Real       0.97      0.85      0.90       110
        Fake       0.86      0.97      0.91       110

    accuracy                           0.91       220
   macro avg       0.92      0.91      0.91       220
weighted avg       0.92      0.91      0.91       220
```

Fig: Frame Level Results for Model_V1

Fig: Video Level Results for Model_V1



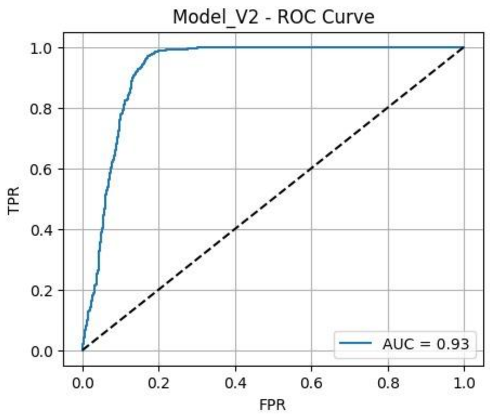
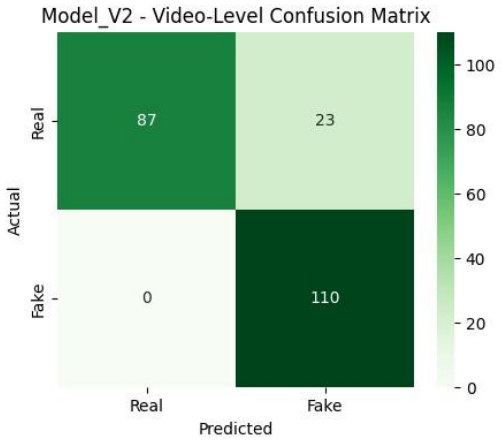


Fig: Frame Level confusion Matrix for Model_V2 **Fig:** Frame Level ROC Curve for Model_V2



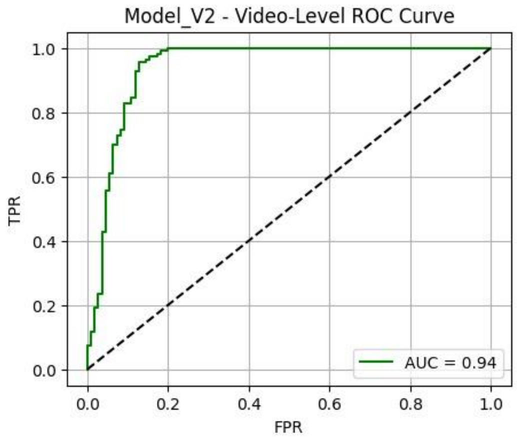
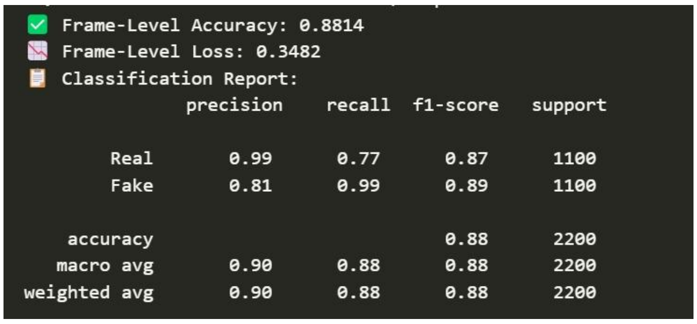


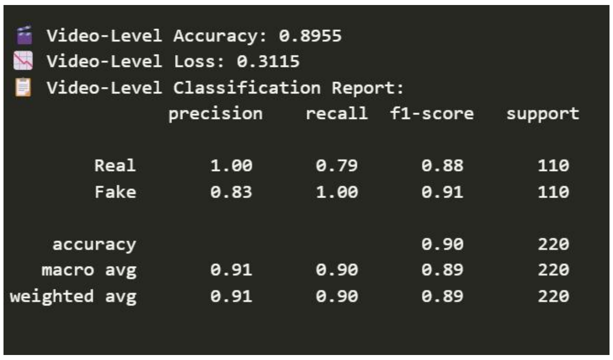
Fig: Video Level confusion Matrix for Model_V2 **Fig:** Video Level ROC Curve for Model_V2



```
✅ Frame-Level Accuracy: 0.8814
📉 Frame-Level Loss: 0.3482
📋 Classification Report:
              precision    recall  f1-score   support

        Real       0.99      0.77      0.87      1100
        Fake       0.81      0.99      0.89      1100

    accuracy                           0.88      2200
   macro avg       0.90      0.88      0.88      2200
weighted avg       0.90      0.88      0.88      2200
```

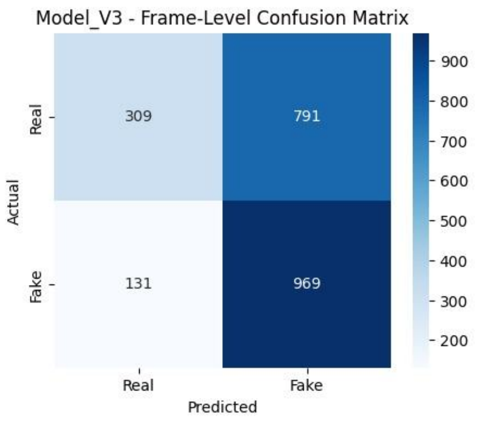


```
🎬 Video-Level Accuracy: 0.8955
📉 Video-Level Loss: 0.3115
📋 Video-Level Classification Report:
              precision    recall  f1-score   support

        Real       1.00      0.79      0.88       110
        Fake       0.83      1.00      0.91       110

    accuracy                           0.90       220
   macro avg       0.91      0.90      0.89       220
weighted avg       0.91      0.90      0.89       220
```

Fig: Frame Level Results for Model_V2 **Fig:** Video Level Results for Model_V2



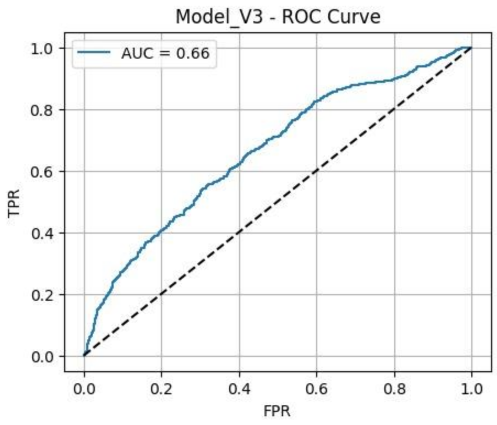


Fig: Frame Level confusion Matrix for Model_V3 **Fig:** Frame Level ROC Curve for Model_V3

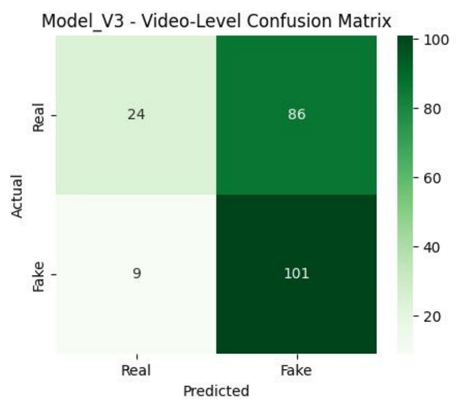


Fig: Video Level confusion Matrix for Model_V3

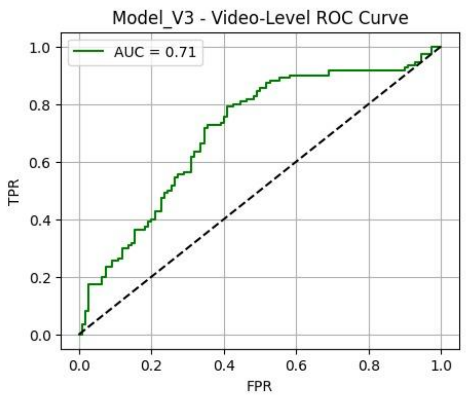
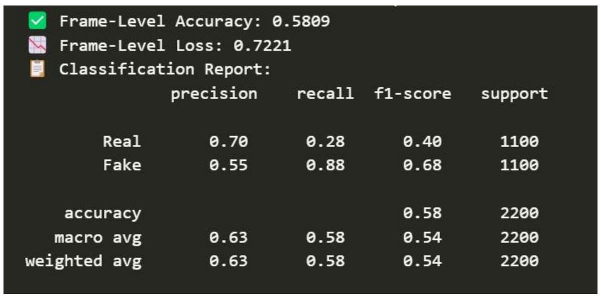


Fig: Video Level ROC Curve for Model_V3

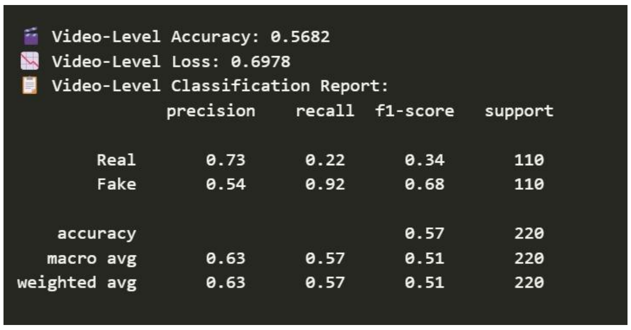


```
✅ Frame-Level Accuracy: 0.5809
📉 Frame-Level Loss: 0.7221
📋 Classification Report:
              precision    recall  f1-score   support

        Real       0.70      0.28      0.40      1100
        Fake       0.55      0.88      0.68      1100

    accuracy                           0.58      2200
   macro avg       0.63      0.58      0.54      2200
weighted avg       0.63      0.58      0.54      2200
```

Fig: Frame Level Results for Model_V3



```
🎬 Video-Level Accuracy: 0.5682
📉 Video-Level Loss: 0.6978
📋 Video-Level Classification Report:
              precision    recall  f1-score   support

        Real       0.73      0.22      0.34       110
        Fake       0.54      0.92      0.68       110

    accuracy                           0.57       220
   macro avg       0.63      0.57      0.51       220
weighted avg       0.63      0.57      0.51       220
```

Fig: Video Level Results for Model_V3

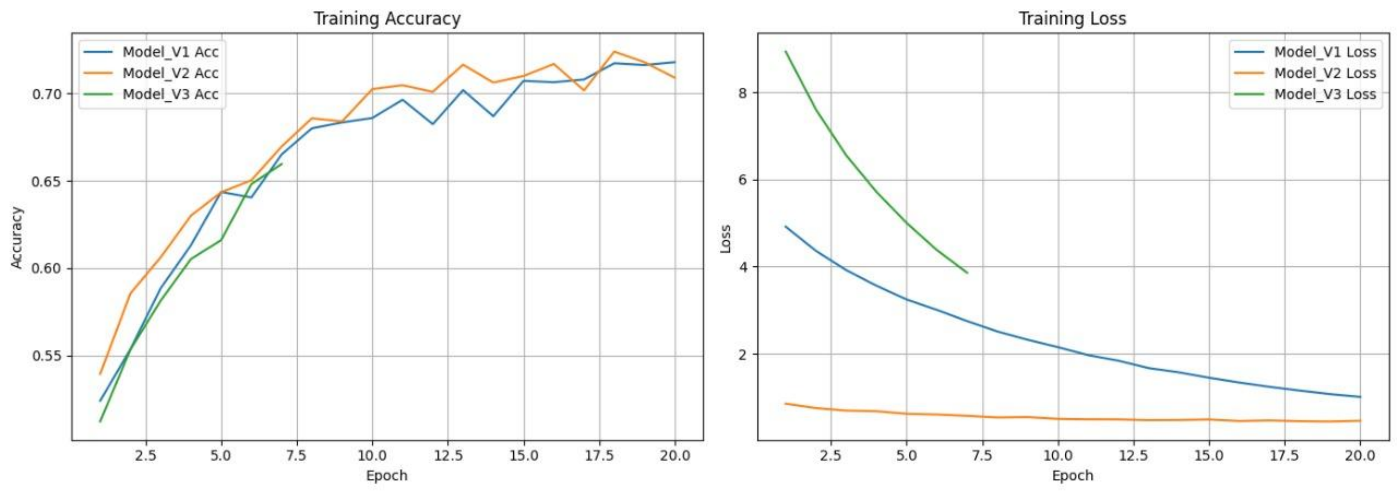


Fig: Training Accuracy and Training Loss Comparison of Model_V1, Model_V2, Model_V3

References

1. Vashistha, M., Jain, S., Pandey, S., Pradhan, A., & Tarwani, S. (2024). A comparative analysis of machine learning and deep learning approaches in deepfake detection. 2017 IEEE Region 10 Symposium (TENSYMP), 1–8. https://doi.org/10.1109/tensymp61132.2024.10752209

2. Karandikar, A. (2020). Deepfake video detection using convolutional neural network. International Journal of Advanced Trends in Computer Science and Engineering, 9(2), 1311–1315. https://doi.org/10.30534/ijatcse/2020/62922020

3. 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), Hong Kong, China, 1–7. https://doi.org/10.1109/WIFS.2018.8630761

4. Jung, T., Kim, S., & Kim, K. (2020). DeepVision: DeepFakes detection using human eye blinking pattern. IEEE Access, 8, 83144–83154. https://doi.org/10.1109/access.2020.2988660

5. Adnan, S. R., & Abdulbaqi, H. A. (2022). Deepfake video detection based on convolutional neural networks. Deepfake Video Detection Based on Convolutional Neural Networks, 65–69. https://doi.org/10.1109/icdsic56987.2022.10075830

6. Gharde, D., A, M. P., N, S., & S, S. K. (2022). Detection of Morphed Face, Body, Audio signals using Deep Neural Networks. 2022 IEEE 7th International Conference for Convergence in Technology (I2CT), 1–6. https://doi.org/10.1109/i2ct54291.2022.9825423

7. Patel, Y., Tanwar, S., Bhattacharya, P., Gupta, R., Alsuwian, T., Davidson, I. E., & Mazibuko, T. F. (2023). An improved dense CNN architecture for deepfake image detection. IEEE Access, 11, 22081–22095. https://doi.org/10.1109/access.2023.3251417

8. Kaushal, A., Singh, S., Negi, S., & Chhaukar, S. (2022). A Comparative Study on Deepfake Detection Algorithms. 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 854–860. https://doi.org/10.1109/ICAC3N56670.2022.10074593

9. Rana, M. S., Nobi, M. N., Murali, B., & Sung, A. H. (2022). Deepfake Detection: A Systematic Literature Review. IEEE Access, 10, 25494–25513. https://doi.org/10.1109/access.2022.3154404

10. Kaur, A., Hoshyar, A. N., Saikrishna, V., Firmin, S., & Xia, F. (2024). Deepfake video detection: challenges and opportunities. Artificial Intelligence Review, 57(6). https://doi.org/10.1007/s10462-024-10810-6

11. Bonettini, N., Cannas, E. D., Mandelli, S., Bondi, L., Bestagini, P., & Tubaro, S. (2021). Video Face manipulation detection through ensemble of CNNs. 2022 26th International Conference on Pattern Recognition (ICPR). https://doi.org/10.1109/icpr48806.2021.9412711

12. Rossler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., & Niessner, M. (2019). FaceForensics++: Learning to Detect Manipulated Facial Images. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), 1–11. https://doi.org/10.1109/iccv.2019.00009