FPT Education

FPT UNIVERSITY

REAL-TIME DEEPFAKE DETECTION

Group 3 member:

Anh Khoi

Hoang Kiet

Bich Tuyen

Minh Chanh

Nguyen Trinh

Vu Huy

Thien Phu

FPT Education

FPT UNIVERSITY

Quay lại chương trình làm việc

Week 3: Building a Shallow Neural Network

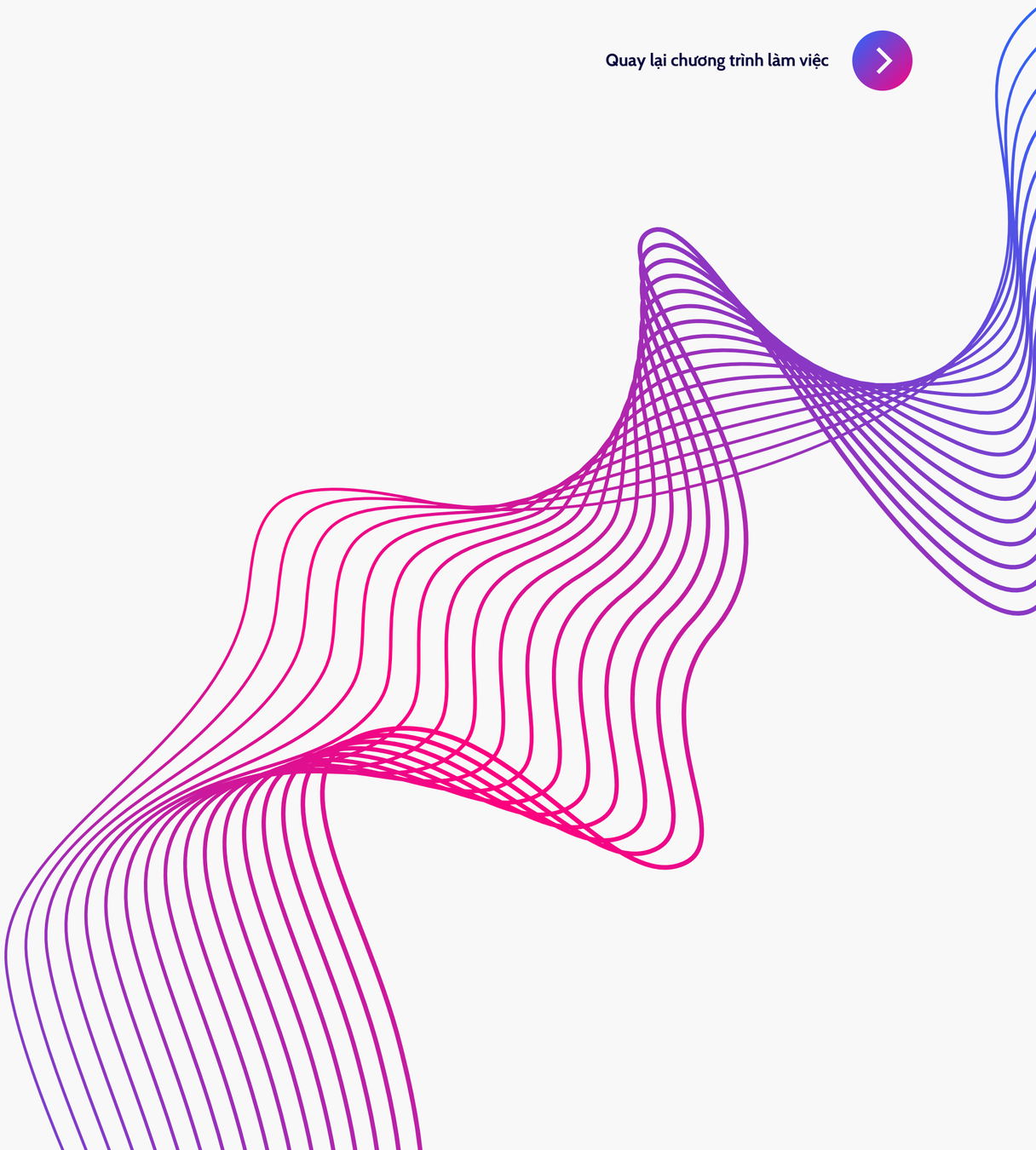
4.1 Introduction
4.2 Model Architecture
4.3 Hyperparameters
4.4 Model Training
4.5 Model Evaluation
4.6 Results

FPT Education
FPT UNIVERSITY

Quay lại chương trình làm việc

Week 4: Designing the CNN

4.1 Introduction
4.2 Model Architecture
4.3 Hyperparameters
4.4 Model Training
4.5 Model Evaluation
4.6 Results



FPT Education
FPT UNIVERSITY

4.1 Introduction

Explored different CNN architectures: VGG16, ResNet50, and MobileNet for deepfake detection.
Chose ResNet50 due to:

- Deep architecture for capturing complex patterns.
- Residual connections to prevent the vanishing gradient problem.
- Strong feature extraction capabilities for improved classification.

Objective: Train ResNet50 for binary classification using TensorFlow/Keras.

FPT Education
FPT UNIVERSITY

4.2 Model Architecture

Input: 224x224 RGB images.
Feature Extraction:

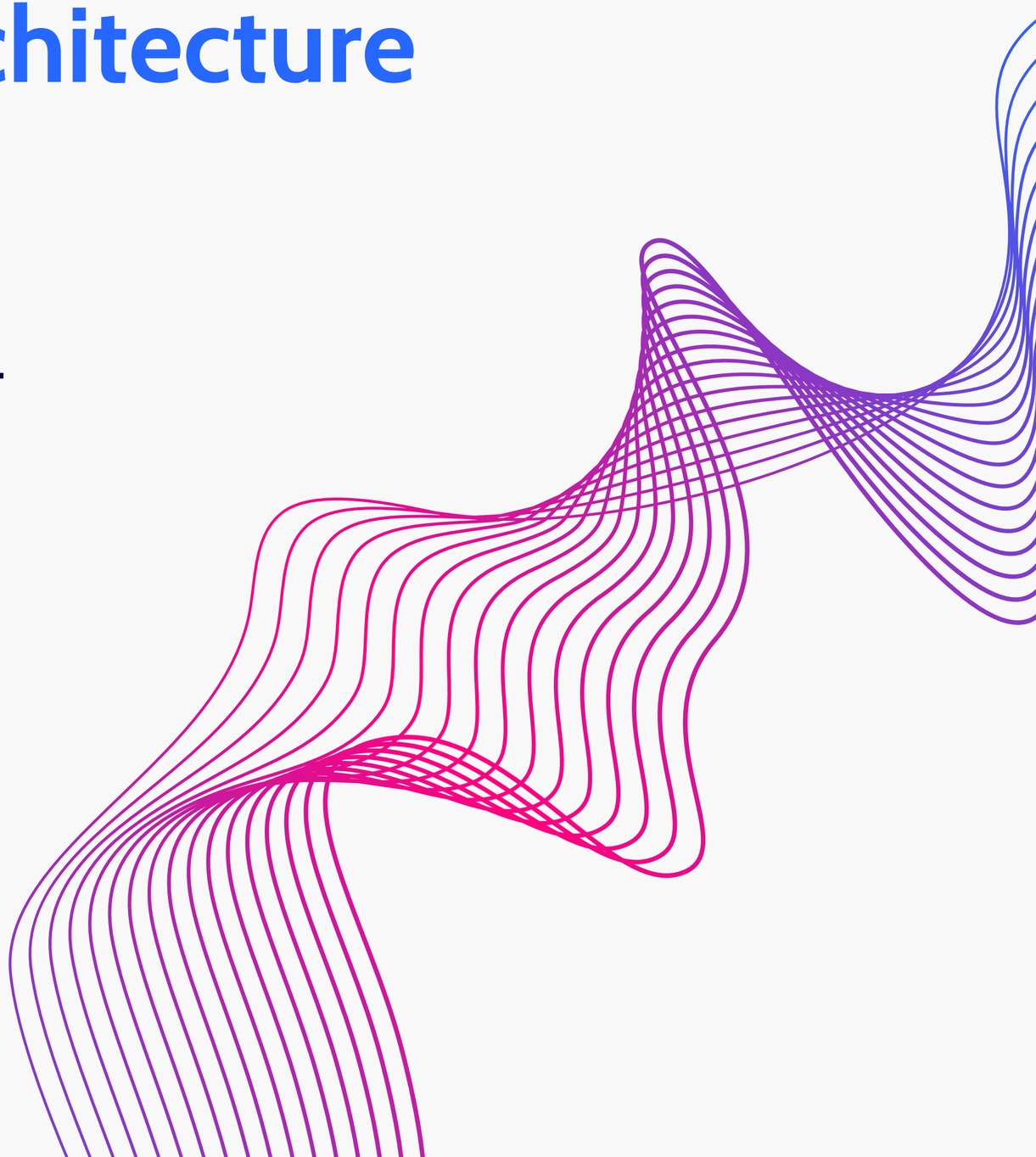
- Convolutional layers with ReLU activation.
- Max-pooling layers for dimensionality reduction.
- Residual blocks (ResNet50-inspired) to enhance gradient flow.

Fully Connected Layers:

- 1024-neuron dense layer with ReLU activation for feature aggregation.
- Dropout layer (0.5) to prevent overfitting.

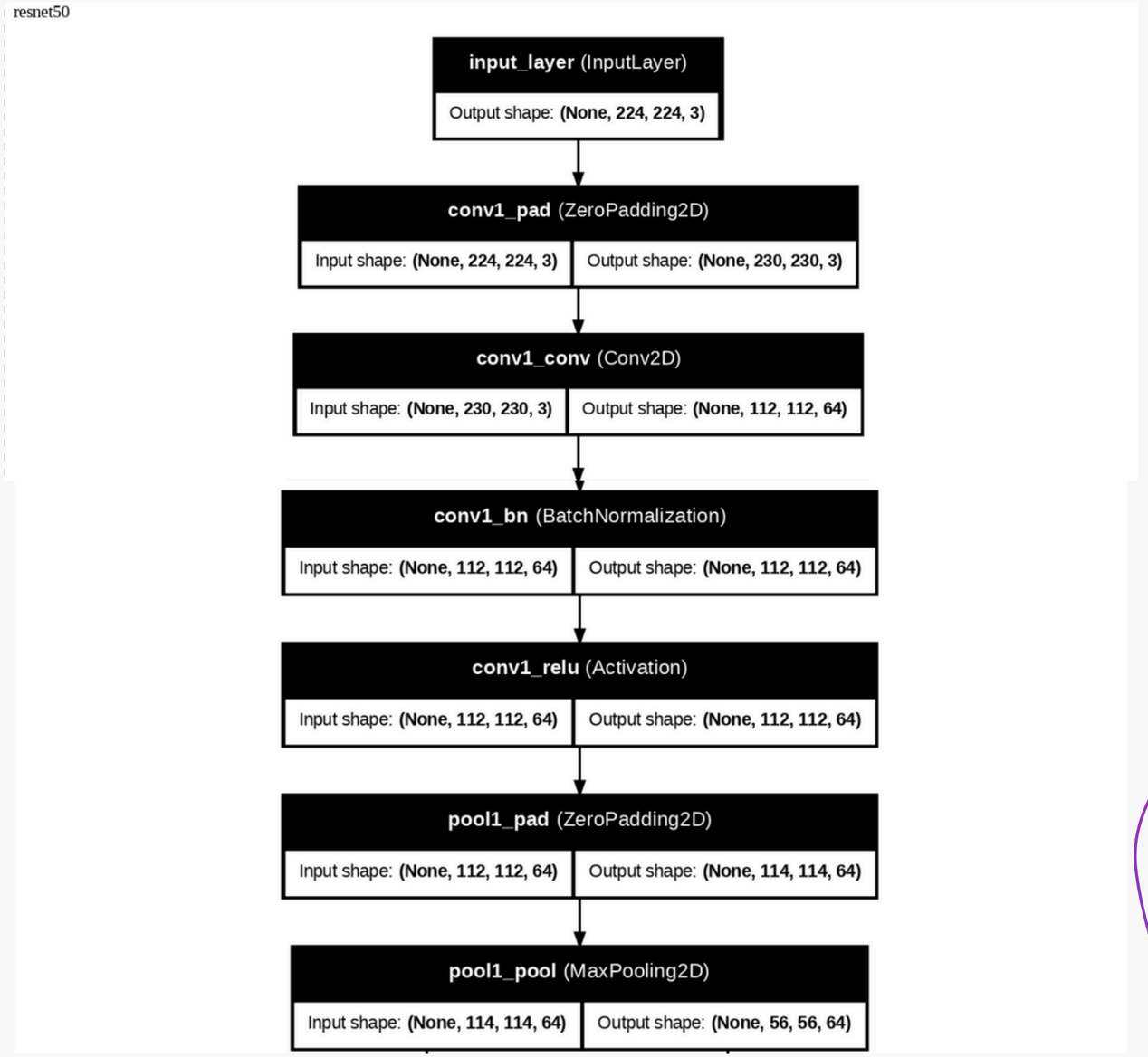
Output Layer:

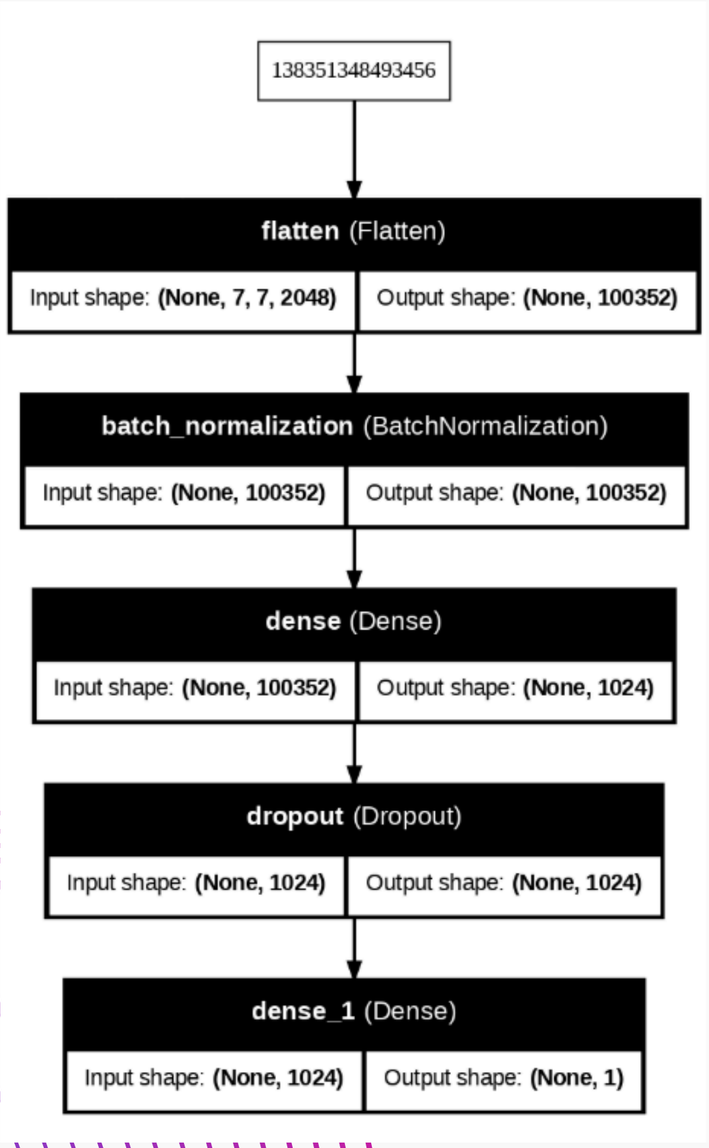
- Sigmoid activation function for binary classification.



FPT Education
FPT UNIVERSITY

4.2 Model Architecture





FPT Education
FPT UNIVERSITY

4.3 Hyperparameters

- Learning rate: 0.0001
- Batch size: 32
- Training epochs: 10
- Loss function: Binary cross-entropy
- Optimizer: Adam (for efficient convergence)

4.4 Model Training

- Dataset: Real and fake images
- Data Augmentation: Rotation, flipping, shifting (to improve generalization)
- Normalization: Pixel values scaled to [0,1]
- Data Split: 80% training, 20% validation
- Training: 10 epochs, with loss and accuracy monitoring to prevent overfitting

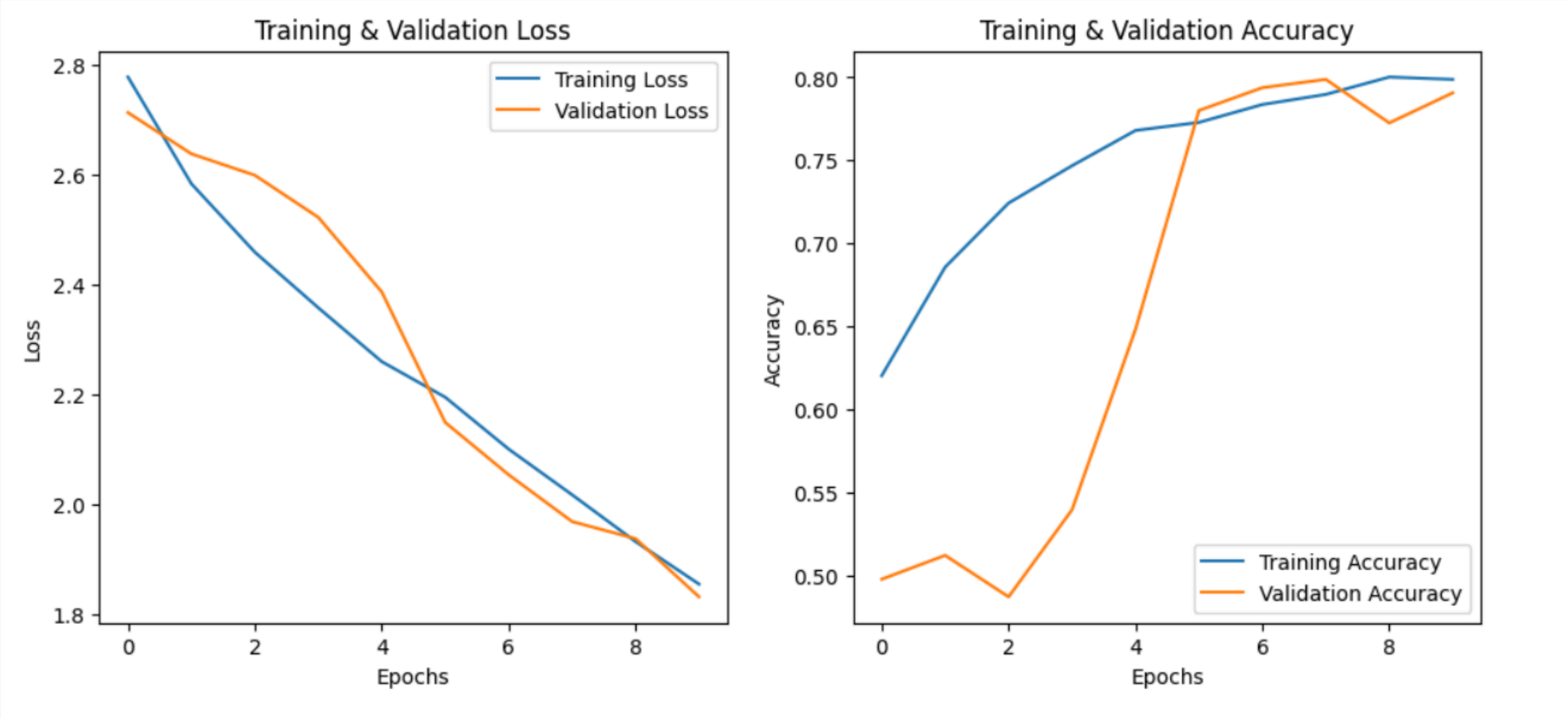
FPT Education
FPT UNIVERSITY

4.5 Model Evaluation

- Final validation accuracy: 79.62%
- Performance Analysis: Confusion matrix, precision, recall, and F1-score
- Training Loss: Decreased over time, confirming successful learning

4.6 Results

- Accuracy improvement: 53.61% → 79.62% over ten epochs
- Validation accuracy: Stable, indicating no severe overfitting


Training & Validation Loss
Training Loss
Validation Loss
Loss
Epochs
2.8
2.6
2.4
2.2
2.0
1.8
0
2
4
6
8
Training & Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
Epochs
0.80
0.75
0.70
0.65
0.60
0.55
0.50

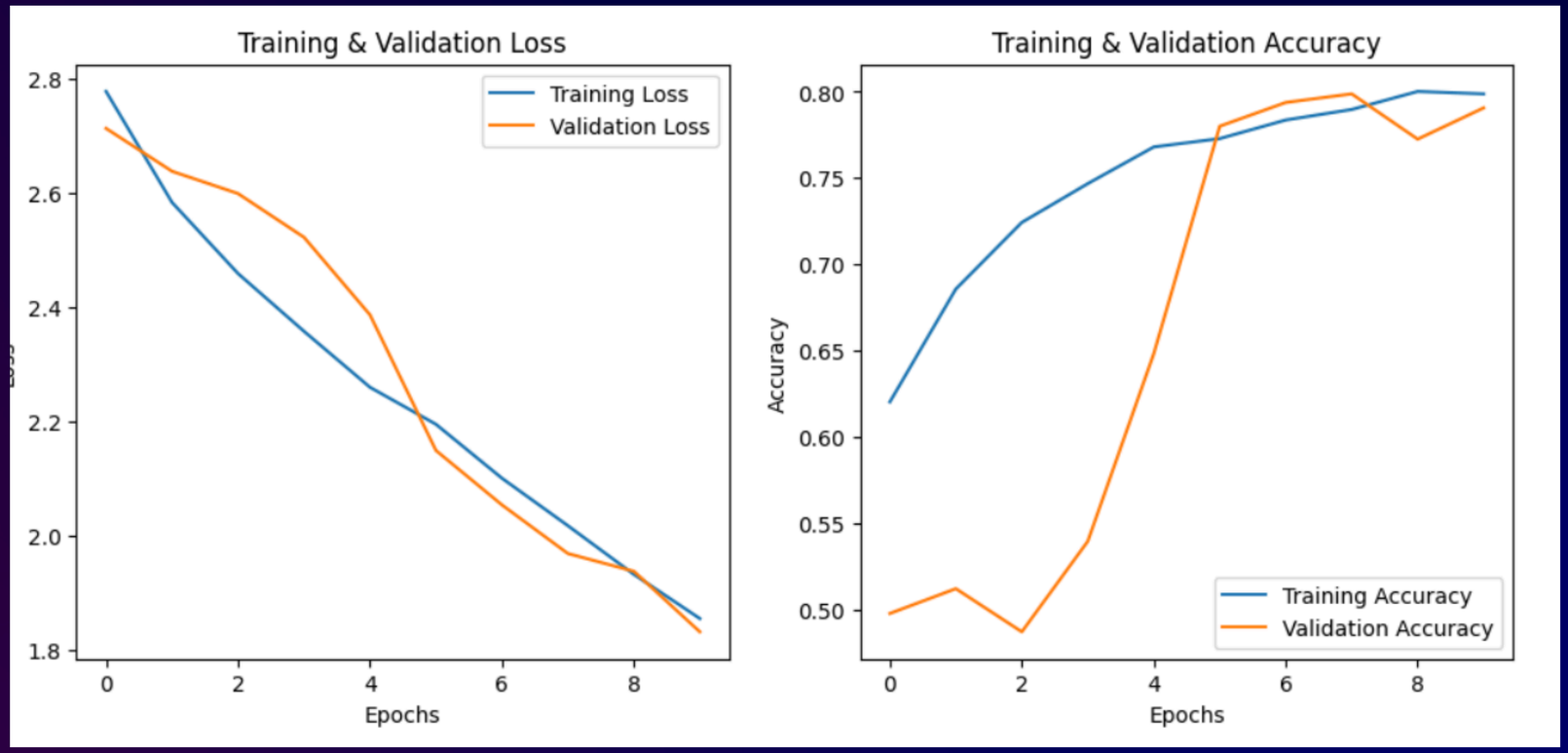
Week 5: Model Training and Optimization

Regularization, BatchNorm, Dropout, and optimization algorithms. Tasks:
- Train the CNN on the full dataset with basic hyperparameters.
- Implement BatchNorm to stabilize and accelerate training.
- Add Dropout layers to mitigate overfitting.
- Test optimization algorithms like SGD and Adam.
- Save model checkpoints and logs for a

Week 5: Model Training and Optimization

Category	Details
🔹 **Model Used**	Pre-trained **ResNet50** Input size **(224, 224, 3)** Applied **data augmentation**
🔹 **Batch Normalization**	Added **BatchNormalization()** after **GlobalAveragePooling2D** Stabilizes and accelerates training
🔹 **Dropout**	Added **Dropout(0.5)** after **Dense** layers Helps reduce overfitting
🔹 **Optimization Algorithms**	**Adam**: Learning rate = **0.0001** **SGD**: Learning rate = **0.001**, momentum = **0.9** Tested both to compare performance
🔹 **Callbacks Used**	**ModelCheckpoint**: Saves best model weights **EarlyStopping**: Stops training if no improvement **ReduceLROnPlateau**: Reduces learning rate when stagnation occurs **TensorBoard**: Logs training metrics for analysis

Week 5: Model Training and Optimization



Week 6: HYPERPARAMETER TUNING

Optimizing Deepfake Detection Model

- Optimizing the ResNet50 model for real-time deepfake detection.
- Goal: Improve accuracy, reduce loss, and enhance model generalization.
- Hyperparameter tuning techniques used: Bayesian Optimization & Hyperband.

FPT Education
FPT UNIVERSITY

Week 6: HYPERPARAMETER TUNING

Key Hyperparameter Tuning Methods

Method	Description	Advantages	Disadvantages
Random Search	Randomly selects hyperparameter combinations within the defined range.	- Simple and easy to implement. - Can find a good configuration faster than Grid Search.	- No guarantee of finding the best configuration. - Can be resource-intensive.
Bayesian Optimization	Uses past evaluations to predict better hyperparameter choices.	- Finds optimal configurations faster than Random Search. - Reduces unnecessary trials.	- Requires more computation. - Can get stuck in local optima.
Hyperband	Runs multiple models in parallel and stops underperforming ones early.	- Saves computational resources significantly. - Speeds up the search process.	- May prematurely stop promising models.
Grid Search (Not in Keras Tuner)	Systematically explores all possible hyperparameter combinations.	- Guarantees finding the best configuration within the search space.	- Very slow and resource-heavy. - Not feasible for large hyperparameter spaces.

FPT Education
FPT UNIVERSITY

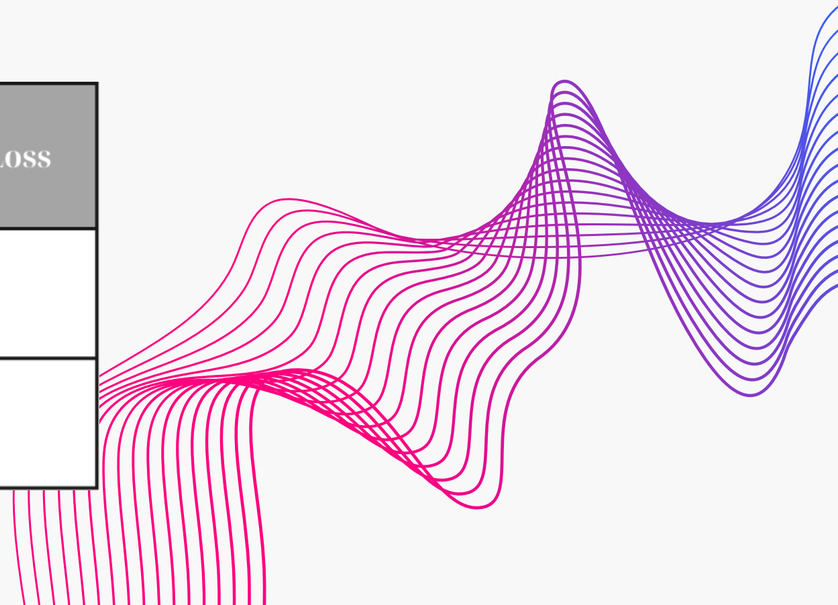
Week 6: HYPERPARAMETER TUNING

Techniques Implemented

- Pretrained ResNet50 Fine-Tuning: Last 20 layers trainable.
- Regularization Methods: L2 regularization, dropout, batch normalization.
- Learning Rate Optimization: ReduceLROnPlateau for dynamic learning rate adjustment.
- Early Stopping: Prevents overfitting by stopping training when validation loss plateaus.

Comparison of ResNet50 Before and After Tuning

Model Version	Train Accuracy	Validation Accuracy	Train Loss	Validation Loss
ResNet50 (Baseline)	85.2%	81.5%	0.45	0.38
ResNet50 (Tuned)	92.3%	86.8%	0.18	0.30



FPT Education
FPT UNIVERSITY

Week 7: Real-Time System Integration

Introduction

- Importance: Real-time deepfake detection is crucial for preventing the misuse of synthetic media.
- Focus: Integrating the trained deepfake detection model into a real-time system using OpenCV.
- Goal: Efficiently process video streams while maintaining high detection accuracy.

FPT Education
FPT UNIVERSITY

Week 7: Real-Time System Integration

Extracting Frames for Deepfake Detection

Steps:

- Use read() from cv2.VideoCapture() to extract frames.
- Apply facial detection using cv2.CascadeClassifier() (Haar cascades) or deep learning-based detectors (MTCNN).
- Preprocess images: Resize, normalize, and convert to tensor format for deepfake detection model input.

Week 7: Real-Time System Integration

FPT Education
FPT UNIVERSITY

Testing with Live Video and Recorded Files

Live Webcam Input:

- Create cv2.VideoCapture(0) for webcam.
- Continuously read and display frames with cv2.imshow().
- Implement exit mechanism (q key press).

Evaluating Recorded Videos:

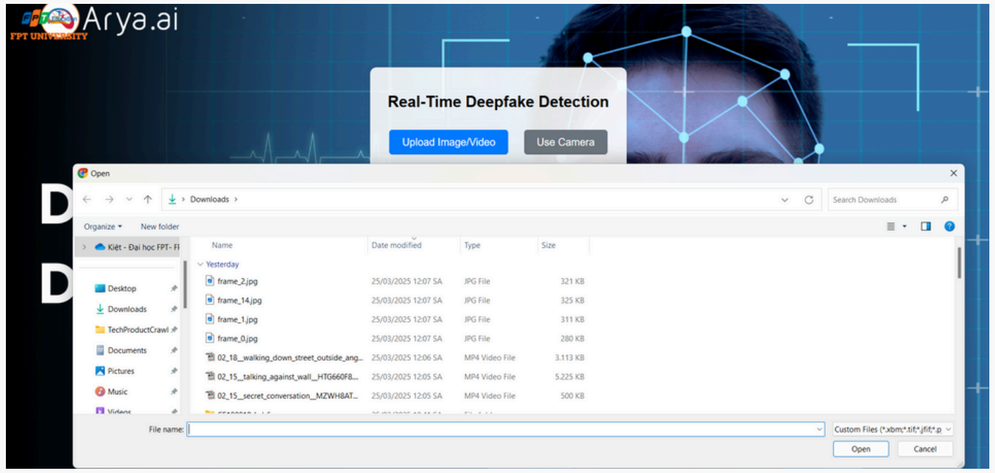
- Read from video files using cv2.VideoCapture("video.mp4").
- Test with different resolutions and lighting conditions.
- Analyze and compare results with live input.

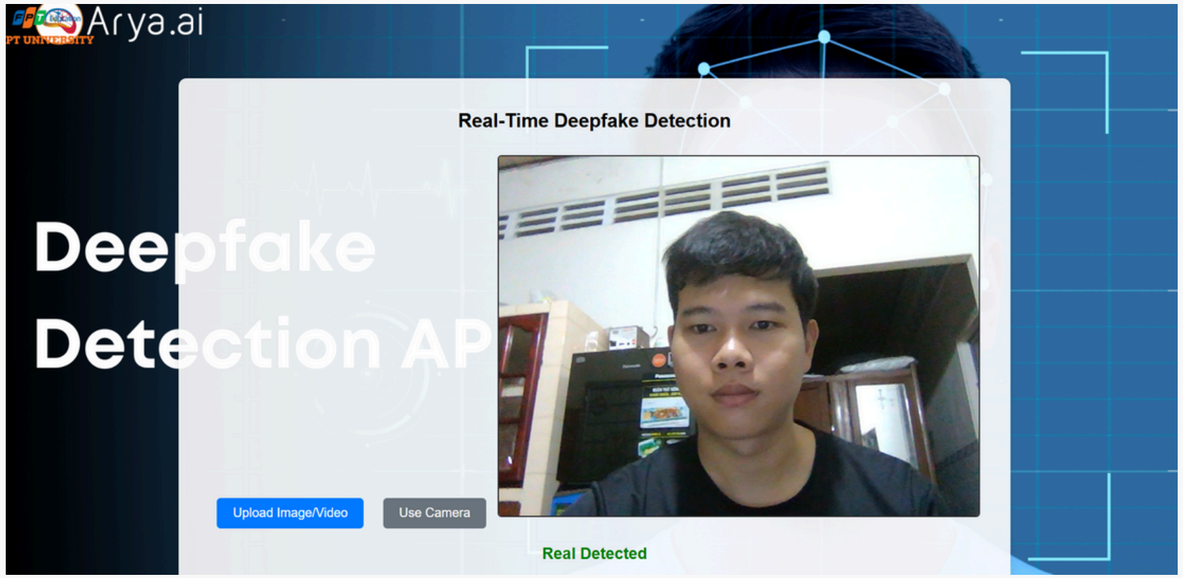
Analyzing System Latency & Detection Accuracy:

- Measure frames per second (FPS) to assess latency.
- Apply detection algorithms and compute accuracy metrics.
- Compare performance between live and recorded inputs.

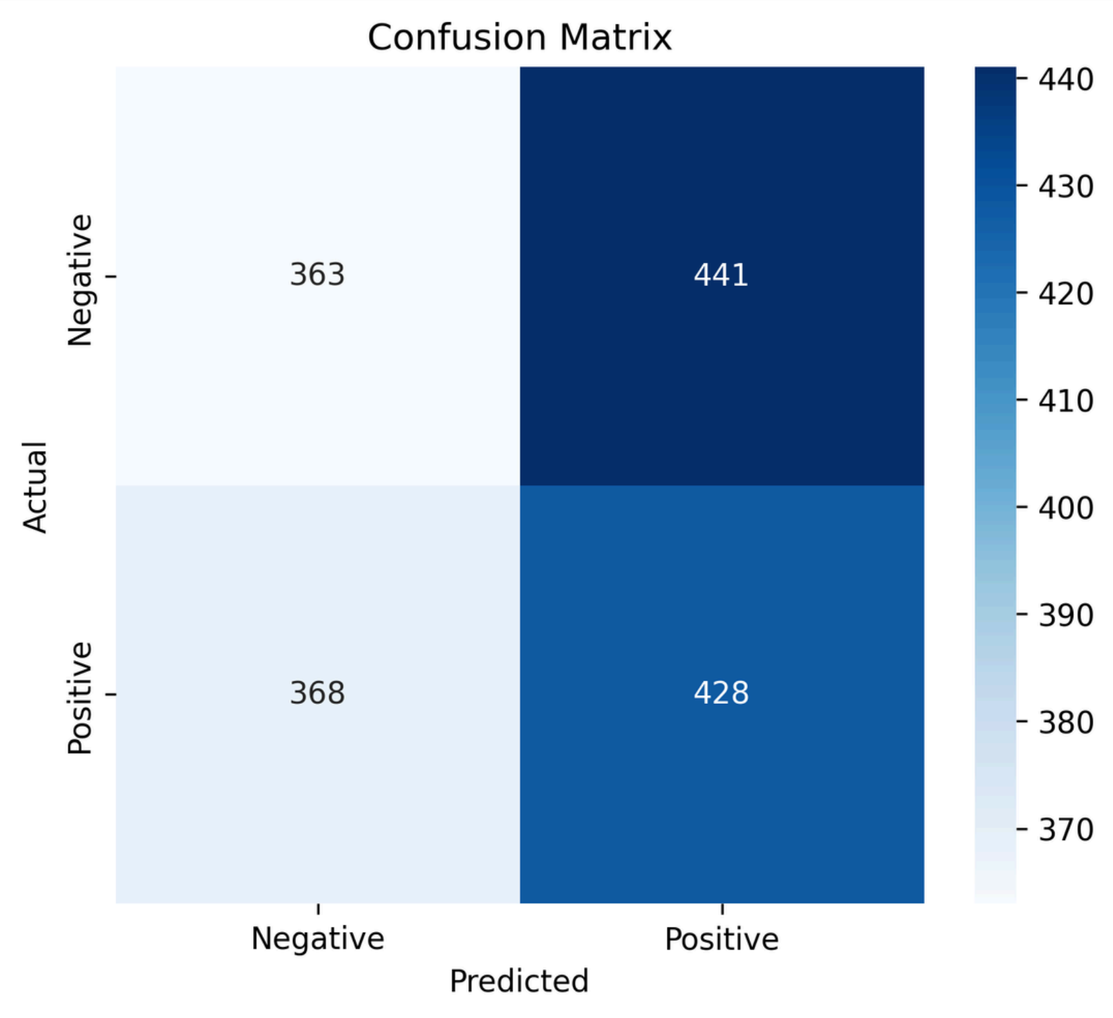
Week 7: Real-Time System Integration

Results and Evaluation





Week8 - Advance Testing



	precision	recall	f1-score	support
0	0.50	0.45	0.47	804
1	0.49	0.54	0.51	796
accuracy			0.49	1600
macro avg	0.49	0.49	0.49	1600
weighted avg	0.49	0.49	0.49	1600

C. Overall Accuracy

The overall accuracy of the model is calculated as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

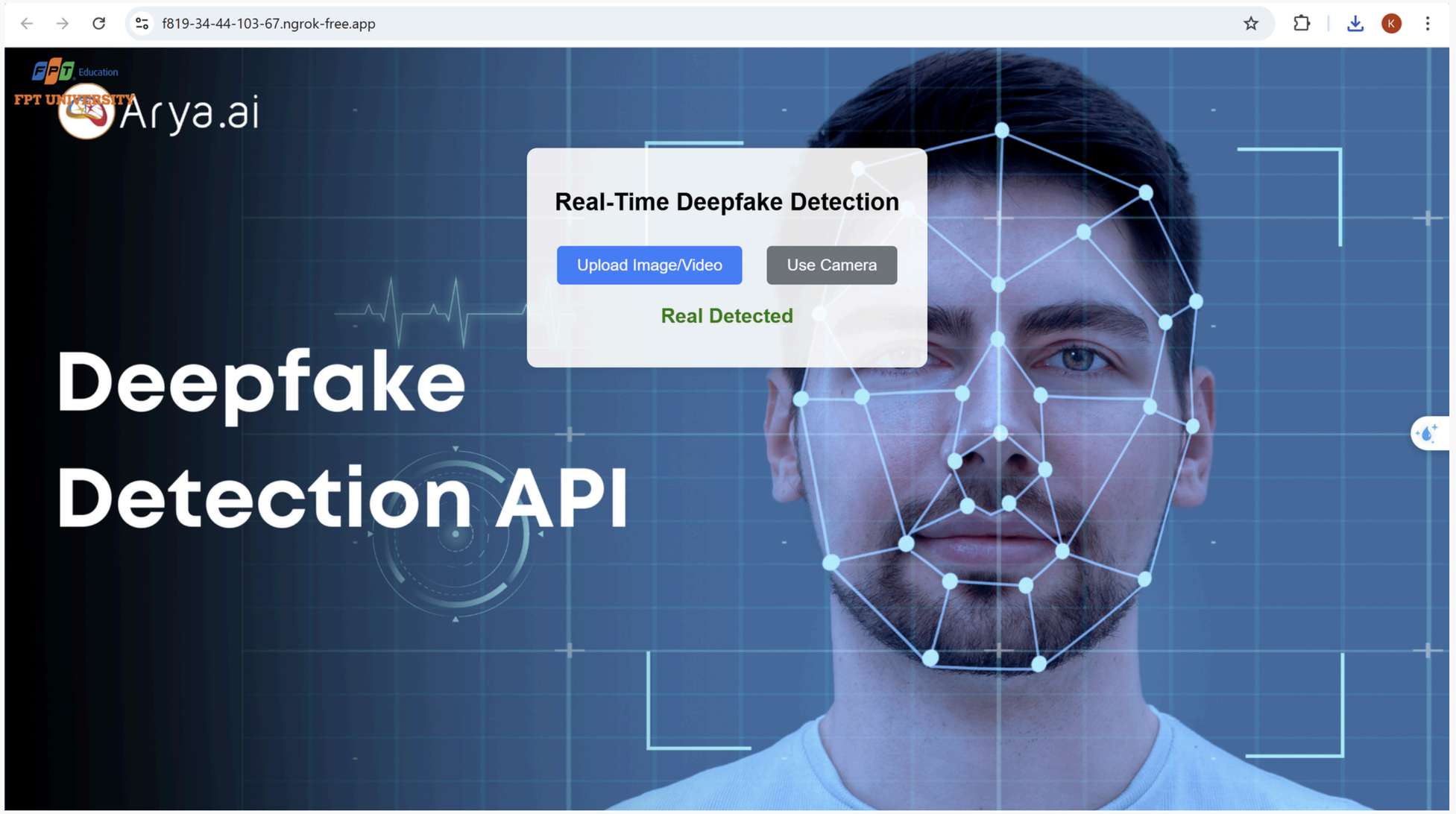
$$= \frac{428 + 363}{428 + 363 + 441 + 368}$$

$$= \frac{791}{1600} \approx 0.4944$$

0: Fake - 1: Real

Week8 - Advance Testing

FPT Education
FPT UNIVERSITY



FPT Education
FPT UNIVERSITY