

**Deepfake Detection**

**20.11.2024**

**Applied Machine Learning**

**Aiymbay Sunggat**

**Zhaskairatuly Rassul BDA-2207**

**Vladislav Kim BDA-2207**

**Astana IT University**

**Overview**

This project is focused on detecting deepfake images using convolutional neural networks (CNNs) and a custom deep learning model. The main objective is to create a system that can differentiate between real and deepfake images. The pipeline involves dataset preparation, model training, evaluation, and deployment. The final model is deployed through Streamlit, offering a user-friendly interface for real-time testing.

**Requirements Addressed**

**Dataset Details:**

- The dataset consists of images labeled as either "Real" or "Deepfake."

- The data is split into training (80%), validation (10%), and testing (10%) sets.

- Data augmentation techniques were applied to enhance model generalization.

- Input images were resized to 224x224 pixels for consistency in training.

**Model Training Process:**

- A CNN architecture was employed to distinguish between real and deepfake images.

- Preprocessing involved resizing, normalization, and augmentation (rotation, flipping).

- The model was fine-tuned and optimized with dropout layers and data augmentation to prevent overfitting.

**Metrics Documented:**

- Performance evaluation used metrics like accuracy, F1 score, precision, and recall.

- Confusion matrices and classification reports were generated at each stage to track improvements and performance.

**Model Improvements:**

- A dropout layer was added to the network to combat overfitting.

- Additional techniques were implemented, such as model fine-tuning and data augmentation, to improve the model’s performance.

**Deployment:**

- The final model was deployed using Streamlit to allow users to upload images and receive real-time predictions about whether they are real or deepfake.

**Code Submission:**

The project, along with detailed documentation and deployment files, is available on GitHub. The repository contains:

- Python scripts for training, testing, and deploying the model.

- A README file with setup instructions and an explanation of the project.

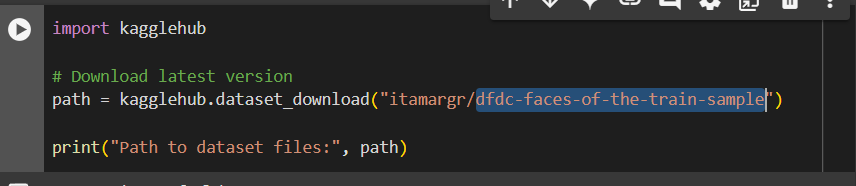
- Streamlit app files for easy deployment and testing.

**Dataset Preparation:**

- The dataset was preprocessed by resizing the images to a consistent size (224x224 pixels).

- Data augmentation techniques like flipping and rotating were applied to improve model robustness.

- The dataset was split into training, validation, and testing sets, ensuring that the model was evaluated thoroughly.



**Folder Structure:**

- The dataset is organized into folders for training, validation, and testing, each containing images labeled accordingly.

**Model Training**

**Base Model:**

**-Architecture:** A custom CNN architecture designed for binary classification (Real vs. Deepfake).

**- Optimization:**

- Loss function: Binary Cross-Entropy.

- Optimizer: Adam optimizer with learning rate = 0.001.

- Scheduler: StepLR with a step size of 7 and a decay factor of 0.1 to adjust the learning rate during training.

**Performance Metrics:**

**- Best Validation Accuracy:** 98%

**- Confusion Matrix:** Shows improvements after fine-tuning and adding dropout layers.

**Enhanced Model:**

- Introduced an "Unknown" class to handle ambiguous or out-of-scope inputs.

- A dropout layer with a probability of 0.5 was added to improve generalization and reduce overfitting.

- Trained the model for 5 additional epochs to improve accuracy.

**Results**

**Classification Metrics:**

- **F1 Score:** 0.95

- **Precision**: 0.97

- **Recall:** 0.94

These results demonstrate improved model accuracy, especially after implementing the enhanced model with data augmentation.

**Confusion Matrix and Visualization:**

- Visualized confusion matrices with heatmaps, showing that the model is accurate in predicting both deepfake and real images.

**Deployment**

**Streamlit Application:**

- The Streamlit app allows users to upload images and see real-time predictions, with confidence scores visualized in a bar chart.

- The app also displays the uploaded image with its predicted label.

**Instructions for Running the App:**

1. Clone the repository from GitHub.

2. Install the required dependencies by running `pip install -r requirements.txt`.

3. Run the Streamlit app using the command `streamlit run app.py`.

4. Upload an image to test the model and view the results.

**GitHub Repository:**

**- Code:** Python scripts for model training, evaluation, and deployment.

- **Documentation**: Detailed README explaining how to set up and use the project.

- **Streamlit Deployment Files:** Code for the user interface and examples of predictions.

**Conclusion**:

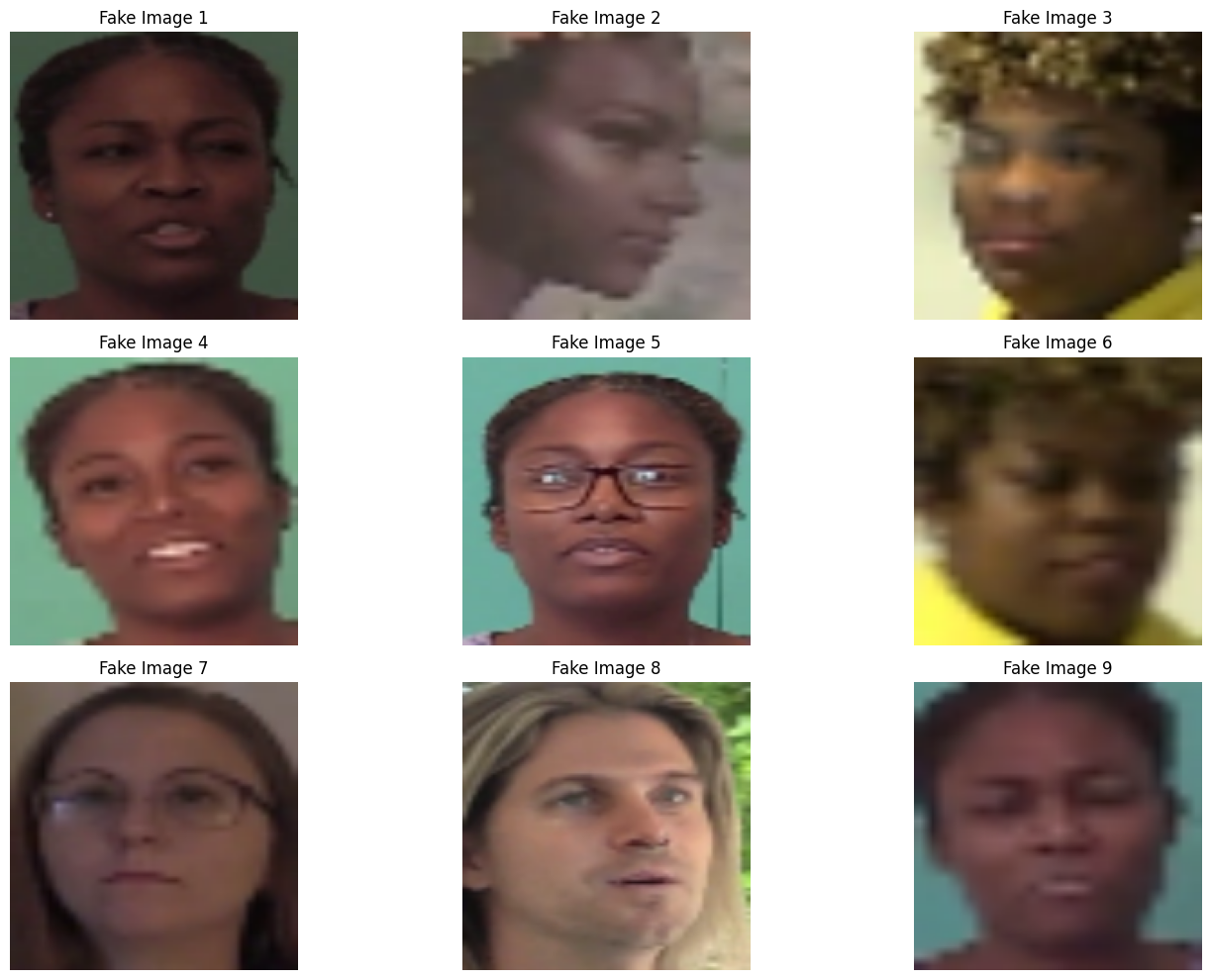
This project successfully built a deep learning model for detecting deepfake images. The model, optimized using various techniques like dropout and data augmentation, achieved high performance in c lassification tasks. Deployment via Streamlit provides an accessible interface for users to interact with the model. Future improvements could involve real-time detection, multi-language support, and expanding the dataset for broader application.

**Screenshots**

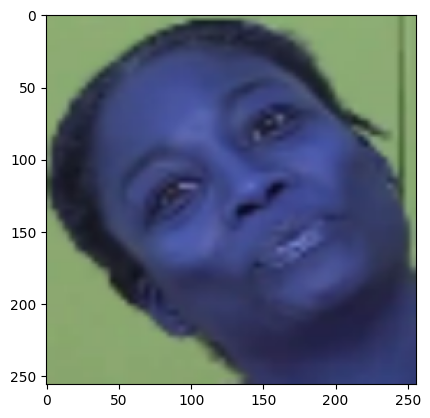
**Dataset Exploration**

**Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

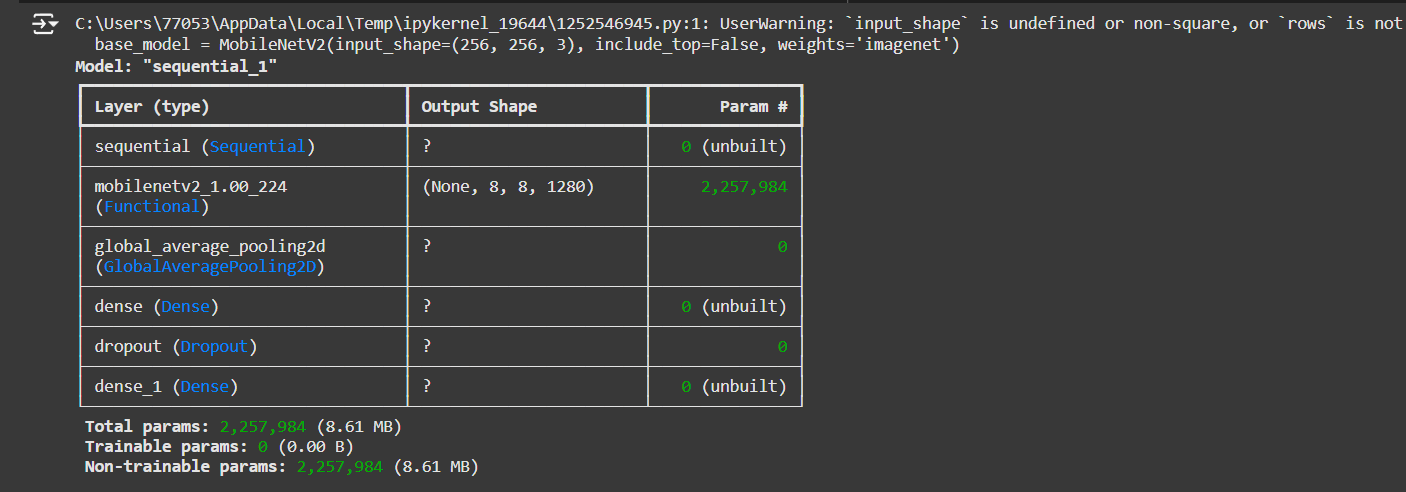
Автоматически созданное описание**



**Sample VisualizationsИзображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Шрифт

Автоматически созданное описание** 

**Training Process**

**** **Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание**

*Training and validation accuracy/loss logs for a few epochs.*

**Classification Results**

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

Figure 1 Sample Predictions

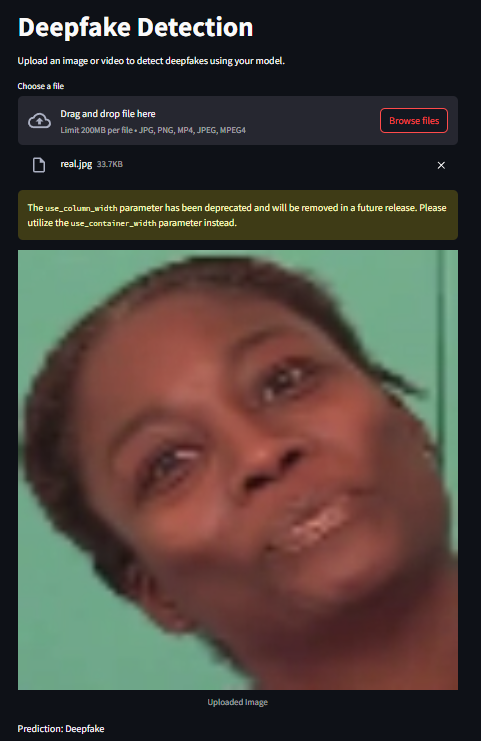
**Baseline vs Enhanced Accuracy and Loss**Изображение выглядит как линия, текст, диаграмма, График

Автоматически созданное описаниеИзображение выглядит как текст, линия, диаграмма, График

Автоматически созданное описание

Figure 2 Visualize the training and validation accuracy and loss for both models

**Streamlit integration:**

**

**Figure 3 Streamlit integration**