Development Document

# Project: DeepFake Detection from Images and Videos

#### Team: Starks

#### Date: January 24, 2023

## **I. Technical Stack**

**Programming Languages:**

* Python (primary language for backend development and model training)
* JavaScript (for frontend development and user interface interactions)
* Frameworks and Libraries:

**Backend:**

* Flask (web framework for building RESTful APIs)
* TensorFlow (deep learning framework for model development)
* NLTK (natural language processing library)
* Pandas and NumPy (data manipulation and analysis libraries)

**Frontend:**

* React (JavaScript framework for building user interfaces)
* D3.js (data visualization library)

**Cloud Platform**:

* Google Cloud Platform (potential use for cloud storage, compute resources, and deployment)

**II. AI Model Architecture**

**Chosen Model:**

* Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) architecture. Captures sequential patterns and context in tweets. Handles long-term dependencies in text data
* The project employees’ multiple models each having a different archetecture as well as different techniques to detect manipulations in the imputed video or image namely CNNs and RNNs (different or hybrid variants of …. <model names from chat gpt as of now>)

**III. Key Functionalities**

**Data Collection and Preprocessing:**

* Datasets for training and testing, such as Face Forensic++, Celeb-DF, DFDC (DeepFake Detection Challenge), FaceForensics, and WildDeepFake, are sourced from open platforms like Kaggle.
* Models that necessitate image inputs were trained using a tailored dataset compiled from the aforementioned datasets, which include labeled videos. Mediapipe, a Google library facilitating machine learning integration into applications, was utilized to concentrate on faces within the videos. Randomly selecting 20-30 frames from each video, solely containing faces as the region of interest, a corresponding metadata with labels was generated. Careful attention was given to ensure a well-balanced dataset, implementing resizing to maintain uniform image sizes.
* For video-based models, dataset preparation involves selecting 20-30 second clips from sources like Face Forensic++, Celeb-DF, DFDC, FaceForensics, and WildDeepFake. Frames are extracted at regular intervals, ensuring diverse facial expressions. Metadata with labels ensures a balanced mix of real and deepfake instances. Standardizing resolution and format guarantees uniformity, and temporal alignment aligns frames with metadata labels, creating a well-prepared dataset for video input in deepfake detection models.
* The aggregation model will be trained on dataset obtained from the historical performance of each individual model and corresponding weights will be chosen for each model to produce a combined result.

**Model Training:**

* The detectors are trained on their respective datasets and further fine-tuning and hyper parameters can be modified if required. The aggregation model is trained on historical performance data of respective models.
* Refinement and hyper parameter tuning for the aggregation model has to be done to improve its performance

**Prediction:**

* Receive
* Receives new image/video as input.
* Preprocess the input using the same pipeline as training data.
* Apply the trained model to predict output (e.g., Authentic, DeepFake).

**Visualization:**

* Display deepfake detection results through charts, graphs, and interactive visualizations. Showcase individual outputs from each model, along with the aggregated result. Justify classifications as deepfake or authentic and provide clear insights for novice users to easily understand the outcomes.

**User Interface:**

* Provide a user-friendly interface for:
* Viewing sentiment analysis results
* Filtering and sorting tweets based on sentiment, keywords, and other criteria
* Exploring sentiment trends over time
* Exporting results for further analysis

**Sample Links:**

* **GitHub Repository and Project Link:** https://github.com/Advaith-Sajeev/Starks-nestria\_hackathon