# Beyond the Pixel: An End-to-End Deepfake Identification System Built with a Multi-Network Combination and Face-Focused Inspection

Swati Kashyap[1], Parth Panwar[2], Aaryan Raj Sinha[3], Sahil Singh Chauhan[4], Aditya Kumar[5]

[1]*Assistant Professor, Department of Computer Science and Engineering, Shivalik College of Engineering, Dehradun, India*
[2,3,4,5]*B.Tech. Student, Department of Computer Science and Engineering, Shivalik College of Engineering Dehradun, India*

Emails: swatikashyap510@gmail.com , parthpanwar80@gmail.com ,aaryanrajsinha2@gmail.com
sahilsinghchauhan36@gmail.com adityakrpjp1103@gmail.com
Corresponding Author Emial: swatikashyap510@gmail.com

***Abstract:*** Deepfake generation technology has advanced rapidly with the development of artificial intelligence, generative adversarial networks (GANs), diffusion models, and neural rendering systems. Modern tools can create highly realistic manipulated images, videos, and speech that are often difficult for viewers to distinguish from authentic content. While these technologies have valuable applications in entertainment, filmmaking, virtual reality, and education, they also pose significant risks, including misinformation, identity theft, cybercrime, political propaganda, and societal manipulation. As deepfake creation becomes easier, faster, and more accessible, the need for reliable and scalable detection methods has become increasingly urgent.

This work presents Beyond the Pixel, an end-to-end deepfake detection framework that combines neural-network-based face analysis with modern web deployment technologies. The proposed approach employs a face-focused workflow in which faces are detected and cropped using Multi-task Cascaded Convolutional Networks (MTCNN), followed by classification through an EfficientNet-B7-based model, DeepFake Classifier, trained on the Deepfake Detection Challenge (DFDC) dataset. By concentrating on facial regions rather than full frames, the method targets manipulation artifacts commonly found in skin textures, boundary regions, eye behavior, and facial expressions.

The framework supports both image and video analysis. Images undergo face cropping, normalization, and EfficientNet-B7 inference to generate confidence-based predictions. For videos, every fifth frame is sampled and analyzed individually, with frame-level predictions aggregated into a final confidence score. The system is implemented using FastAPI and PyTorch on the backend and React with TypeScript on the frontend. Results indicate that face-oriented pre-processing improves inference robustness by reducing background noise and aligning inputs with DFDC-style training data. The study also demonstrates practical deployment through scalable APIs and discusses implementation details, limitations, and future enhancements, including interpretable AI, live-stream analysis, uncertainty quantification, and cloud deployment.

***Keywords—*** *Deepfake Detection, EfficientNet-B7, MTCNN, DFDC Dataset, Computer Vision, Artificial Intelligence, FastAPI, PyTorch.*

## I. INTRODUCTION

Across the past decade, AI and deep learning progress has transformed how digital media is produced. Current generative techniques like GAN-driven models, VAEs, and diffusion-based frameworks can output realistic faces, voices, and short videos. These advances have enabled fresh opportunities in cinema, gaming, AR, education platforms, and digital communication. Yet in parallel with these positive uses, the very same capabilities have produced a harmful form of fabricated media called deepfakes. [3,9]

A deepfake is a digitally tampered image or video where a person's expressions, voice behavior, or identity is synthetically modified to mimic a real individual. As public deepfake-making software are easily available in the Internet, creating fake content has become easier even with limited computing power. As a consequence, deepfakes have become a serious issue for cybersecurity, digital forensics, journalism, policing, and especially social media.

Abuse of deepfake methods can trigger major social harm, such as election-related deception, monetary scams, stolen identities, spread of false reports, damage to reputation, and illicit impersonation. Altered clips featuring well-known people may sway voting outcomes and collective attitudes, and fabricated audio-visual material can cause extortion, fraudulent schemes, or digital attacks. Because current deepfakes can look extremely authentic, relying on people to check content with their naked eyes is no longer adequate for the verification of authenticity.

Older image-forensics methods frequently break down when faced with AI-synthesized media, since deepfake generators can now enhance realism and minimize noticeable editing traces. As a result, new and smart solutions are urgently needed to spot small irregularities inside manipulated content. Among all current techniques, Deep learning forensic techniques have become among the strongest options for finding these edits, because they can absorb concealed visual cues from very large datasets. [2,3]

This paper's project, Beyond the Pixel, seeks to meet these issues by building a usable end-to-end deepfake detection framework that combines deep-learning inference with a contemporary web deployment design. The system uses a face-focused evaluation method, guided by the idea that counterfeit media often creates small discrepancies near facial areas, including skin detail patterns, edges around the eyes, transitions near the mouth, lighting inconsistencies, and artifacts from generated expressions. Instead of classifying entire image frames directly, the system first extracts the facial region using MTCNN-based face detection and then applies an EfficientNet-B7 classifier trained on DFDC-style facial crops. [1,4]The framework supports both image and video-based deepfake detection. For image analysis, the system processes uploaded images through facial extraction, preprocessing, normalization, and classification to produce confidence-based predictions. When analyzing videos, the system captures frames at regular intervals and runs face recognition decisions separately, then combines those outputs to produce a single confidence value for the entire clip.

By doing so, it reduces compute demands yet preserves consistent prediction quality. On the server side, FastAPI and PyTorch are used to build scalable REST inference endpoints, and on the client side, React with TypeScript provides an interactive, easy-to-use interface for operators. In general, the structure is intended to be compact, component-based, and practical to deploy in both research settings and real deployments. [10,11]

This project's main contributions are as follows:

- Creation of an end-to-end deepfake detection system that supports images and video.
- Combination of MTCNN-driven face cropping with EfficientNet-B7 for classification.
- Implementation of confidence-based prediction semantics for improved interpretability.
- Design of a practical React + FastAPI deployment workflow.
- Efficient frame-based video analysis using score aggregation.
- A scalable architecture suitable for future real-time deployment.

In this work, we expand on prior studies, the chosen methods, the design and deployment structure, the results seen in experiments, the constraints encountered, and the potential avenues for later improvement tied to the presented system.

## II. LITERATURE REVIEW

Within artificial intelligence, computer vision, and digital media forensics, detecting deepfakes has become an important line of inquiry. Around the globe, investigators have introduced a range of approaches to spot altered content produced through deep-learning setups, including Generative Adversarial Networks (GANs), autoencoders, and diffusion-oriented architectures. The studies listed next supply the technical and scientific basis on which the suggested system is built.

### *A. DeepFake Detection Challenge (DFDC) Dataset)*

Facebook AI, together with partnering researchers, released the DeepFake Detection Challenge (DFDC) dataset, which ranks among the biggest open-access resources for studying deepfake detection. It includes many thousands of authentic clips alongside altered ones, produced with a range of face replacement and facial reenactment approaches. DFDC was built with the aim of motivating creation of broadly applicable deepfake detectors that can spot manipulation techniques not encountered during training. Because it spans varied illumination, expression changes, viewpoint differences, compression effects, and tampering approaches, the dataset is well suited for developing resilient forensic models. The proposed system utilizes methodologies inspired by DFDC-trained architectures because the dataset provides realistic real-world deepfake scenarios. [3,9]

### *B. Selim Seferbekov's DFDC Solution*

Selim Seferbekov developed one of the highest-performing solutions in the DFDC competition. By integrating EfficientNet variants with sophisticated preprocessing workflows, his approach delivered robust accuracy when classifying altered face images. Instead of processing entire frames, the method

prioritized face-focused recognition, which helped the network concentrate on tampering cues near key facial areas. The project's training setup, its use of an EfficientNet-B7 model, and the chosen preprocessing approach draw motivation from that method. The positive outcome of this design showed that well-tuned CNN-driven models can reliably identify faint, artificial irregularities present in facial pictures.[4]

### C. *EfficientNet: Rethinking Model Scaling for CNNs*

Tan and Le introduced EfficientNet, a family of convolutional neural networks optimized using compound scaling techniques. Traditional CNN scaling methods typically increase either network depth, width, or image resolution independently, often resulting in computational inefficiency. EfficientNet proposed a balanced scaling strategy that improves accuracy while maintaining lower computational cost. The EfficientNet-B7 model used in this project provides strong feature extraction capability for high-resolution facial analysis and achieves excellent classification performance across multiple computer vision tasks. Its balance between inference efficiency and classification accuracy makes it highly suitable for practical deepfake detection systems.

### D. *FaceForensics++*

Face Forensics++ serves as a leading benchmark collection and experimental foundation for spotting altered facial imagery. The authors showed that concentrating analysis on the face markedly raises deepfake detection performance, since artifacts from edits tend to cluster along face edges, near the eyes, across the mouth surface, and within skin transition zones. In that work, several deep-learning designs for detecting tampered media were evaluated, and isolating the face via cropping was identified as an essential preprocessing stage in forensic pipelines. The introduced Beyond the Pixel approach adopts the same idea, applying face extraction driven by MTCNN prior to running the classifier.[2]

### E. *MTCNN Face Detection Framework*

Known as the Multi-task Cascaded Convolutional Network, MTCNN is broadly used for detecting and aligning faces. It relies on a cascaded, multi-step structure to pinpoint facial areas precisely despite changes in illumination, shifts in pose, partial obstruction, and different face angles. In deepfake evaluation, dependable face localization matters because flawed extraction can sharply degrade classification results. The described system uses MTCNN as the main preprocessing component so that facial alignment is consistent before EfficientNet-B7 is applied. As a result, the overall detection workflow becomes more robust and dependable. [5,12]

### F. *Albumentations Image Processing Framework*

Albumentations is a fast library for image augmentation and preprocessing built for deep-learning workloads. It offers functions including resizing, standardization, cropping, conversion to tensors, and augmentation while emphasizing efficient computation. For inference-driven deepfake detection, fixed preprocessing is particularly important because the model anticipates inputs distributed like those seen

during training. Within the presented approach, Albumentations resizes the extracted face crops to 380 × 380 and then normalizes pixel values prior to inference. This setup helps keep model responses steady and strengthens consistency in predictions.[6]

### G. *Deep Learning-Based Media Forensics*

Progress in media forensics indicates that deep-learning models can uncover concealed visual inconsistencies that people typically struggle to notice by hand. Investigations have found that forged media frequently includes faint issues such as uneven facial textures, lighting that appears implausible, irregular blinking, mismatched expressions, and blending defects around generated portions. Convolutional neural networks can learn these latent cues from large datasets, which supports automated identification of manipulated material. Collectively, this body of work underpins the use of CNN-style models in deepfake forensics. [2,7]

### H. *Video-Based Deepfake Detection Systems*

A number of current deepfake detectors target video specifically, leveraging temporal-consistency analysis, motion estimation, recurrent networks, and transformer-based models. By examining links across adjacent frames, these methods can flag unnatural changes, blinking that does not match expectations, and irregular facial motion. Even though temporal strategies can boost detection performance, they also demand far greater computational capacity. The proposed Beyond the Pixel framework adopts a more computationally efficient frame-sampling strategy where every fifth frame is analyzed independently and aggregated into a clip-level confidence score. This balances computational efficiency with practical inference performance.

### I. *Interpretable AI for Deepfake Identification*

In recent studies, attention has shifted toward explainable AI approaches within forensic tools. Models built for interpretability can generate visual aids—like attention visualizations, Grad-CAM style heat displays, and analyses of activated features—to clarify why particular content was judged genuine or fabricated. When systems can justify decisions, confidence, openness, and clarity increase in forensic workflows powered by AI. Even though the present version lacks interpretability components, later additions of attention-driven visual explanation tools could strengthen analyst comprehension and enhance forensic dependability.[2]

### J. *Research on Live Deepfake Identification*

Current directions in the field place growing emphasis on detecting deepfakes in real time for scenarios such as live broadcasts, online meetings, and camera-based use cases. To meet these needs, designs must stay lightweight, leverage GPU tuning, support batch-style inference flows, and keep latency minimal. Ongoing work aims to cut computational burden without sacrificing high recognition performance. Because Beyond the Pixel is built with scalability in mind, it can later support real-time use through GPU-powered execution and refined frame-handling methods.[8]

## III. METHODOLOGY

### A. *Proposed System Overview*

Beyond the Pixel, the suggested solution, is built as an end-to-end deepfake detection stack that can examine both still images and video by applying deep-learning facial analysis. The approach centres on locating face areas in submitted content and then classifying them with an EfficientNet-B7–derived network trained using the DeepFake Detection Challenge dataset.[3]

Within the framework, a DFDC-style process is used: facial areas are first obtained and inspected, and only then is a final label produced. Its preprocessing steps and classification plan draw from DFDC-established designs and typical pipelines, helping the model pick up fine manipulation traces present in facial areas. [3,4]

Rather than standard classifiers that evaluate an entire frame, this design adopts a face-focused detection method. This choice comes from the finding that deepfake edits often create visible mismatches near the face, including around eyes, in skin detail, across mouth motion, within lighting shifts, and along blending borders. By restricting analysis to facial content, the model boosts decision stability and avoids extra interference from background regions.

The complete workflow consists of the following major stages:

1. Media Upload
2. Face Detection using MTCNN
3. Facial Cropping and Preprocessing
4. Deep Learning-Based Classification
5. Confidence Score Generation
6. Frontend Visualization

### B. *System Architecture*

The architecture of the proposed framework consists of three major modules:

- Frontend Interface
- Backend API Server
- Deep Learning Inference Engine

The frontend is developed using React and TypeScript, allowing users to upload media files and visualize predictions. On the server side, FastAPI is used to manage uploads, run preparation steps, route inference calls, and return API outputs. The inference engine uses PyTorch and EfficientNet-B7 for deepfake classification. [1,7,10,11]

The EfficientNet-B7 classifier is based on DFDC-style training methodologies and facial preprocessing techniques inspired by high-performing DeepFake Detection Challenge solutions.

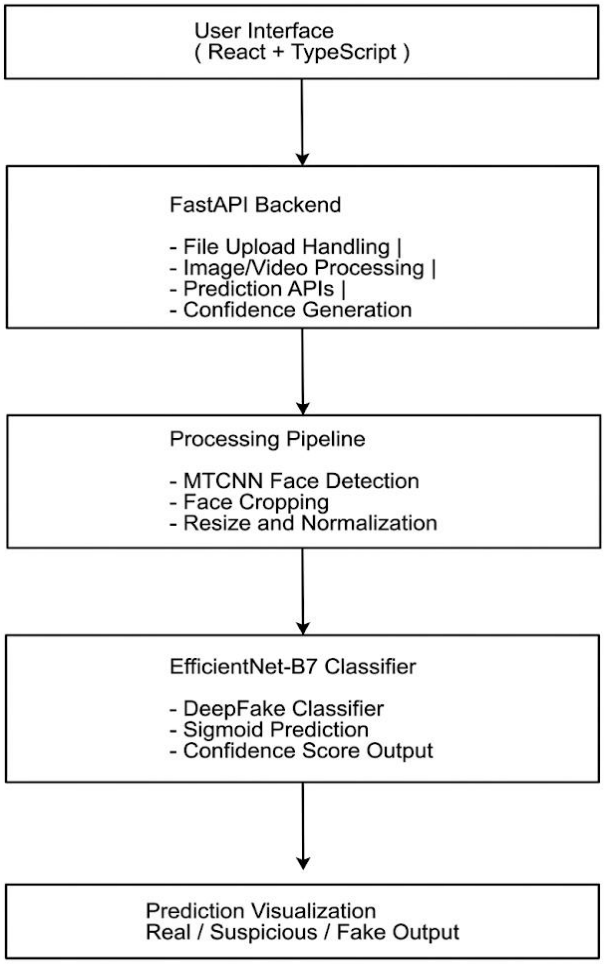


Fig1: *Overall Architecture Flow*

### C. *Face Detection and Facial Extraction*

At the start of the detection workflow, a Multi-task Cascaded Convolutional Network (MTCNN) is used to extract the face region. The detector's role is to locate the main face area and separate it from the submitted media.[5]

MTCNN is selected due to strong performance across changing scenarios, such as:

- Different lighting environments
- Partial occlusions
- Multiple face orientations
- Variable facial expressions

Before running classification, the system crops the detected face since tampering traces tend to appear around facial features rather than in the surrounding scene.

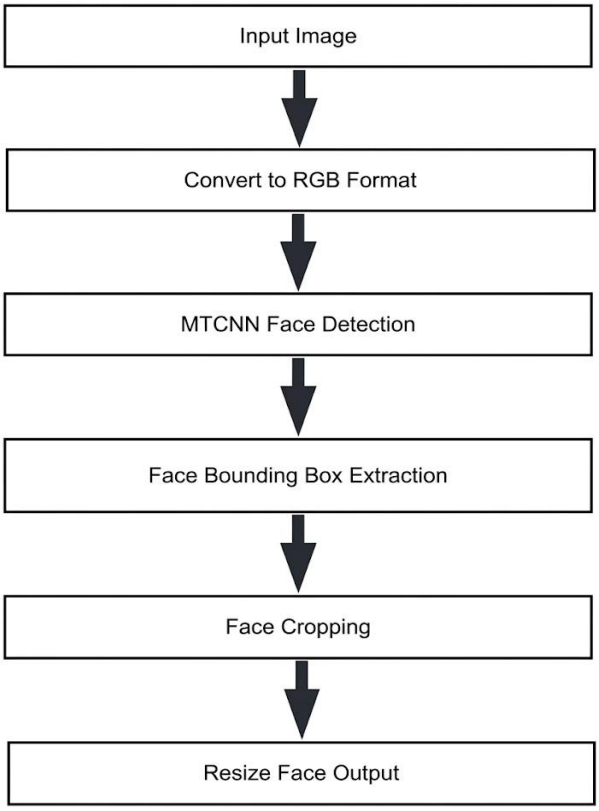


Fig2: *Face Extraction Pipeline*

In this setup, the facenet-pytorch implementation of MTCNN is incorporated into the preprocessing stage.[12]

When no suitable face is found, the system prevents low-confidence outputs and responds with an error rather than producing an untrustworthy classification.

### D. *Image Preprocessing*

Once the face is extracted, the cropped face image is prepared for classification through preprocessing. Careful preparation is required because EfficientNet-B7 expects inputs that match the data distribution used during training.[1]

The preprocessing pipeline includes:

- Face resizing to 380 × 380 resolution
- Pixel normalization
- Tensor conversion
- Batch formatting for inference

For fast preprocessing and normalization, the project relies on the Albumentations library.

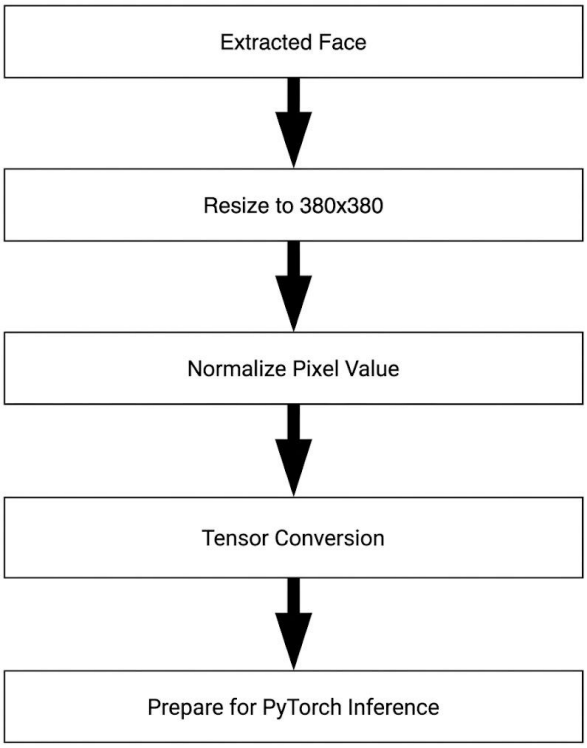


Fig3: *Preprocessing Workflow*

These steps increase stability and help keep inference results consistent and repeatable.

### *E. Deep Learning Classification Model*

At the centre of the system's classification component is EfficientNet-B7, a top-performing member of the EfficientNet model line.[1]

A PyTorch-based, custom Deepfake Classifier design is used in the project. Both the network layout and the preprocessing approach draw inspiration from DFDC competition codebases, especially EfficientNet-centered methods seen in strong DeepFake Detection Challenge entries. The trained model aims to detect manipulation cues including imperfect blending, abnormal textures, lighting inconsistencies, and artificial facial warping that often occur in DFDC-type altered media. Pretrained EfficientNet-B7 parameters are used to initialize the network, and it is then fine-tuned for deepfake detection objectives. [3,4]

Table1: *Model Configuration*

| **Parameter** | **Value** |
|---|---|
| Model | EfficientNet-B7 |
| Framework | Pytorch |
| Input Resolution | 380 x 380 |
| Loss Function | Binary Cross Entropy |
| Optimizer | SGD with Momentum |
| Momentum | 0.9 |
| Weight Decay | 1e-4 |
| Batch Size | 12 |
| Epochs | 40 |

### F. *Image Detection Workflow*

The image detection module processes uploaded images independently. Prediction Interpretation The System generates three confidence-based labels:

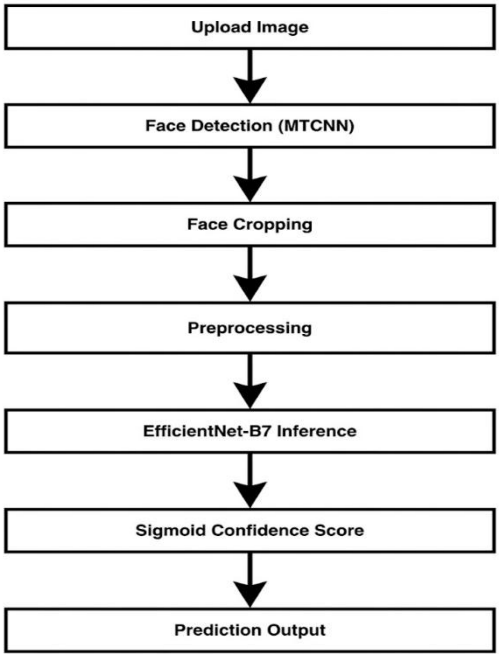


Fig4: *Image Detection Flowchart*

Table2: *Prediction score*

| Confidence Score | Prediction |
|---|---|
| < 0.40 | Likely Real |
| 0.40 – 0.60 | Suspicious |
| > 0.60 | Likely Fake |

This approach improves interpretability compared with strict binary classification.

### G. *Video Detection Methodology*

Video-based deepfake detection is computationally more expensive than image analysis because videos contain multiple frames. To improve efficiency, the proposed framework uses a frame-sampling strategy.[8]

Instead of processing every frame, the system extracts every fifth frame from the uploaded video and performs independent facial classification on each sampled frame.

This frame-sampling strategy is consistent with practical DFDC-style inference methodologies where computational efficiency and stable prediction aggregation are important for large-scale video analysis. [3,4]

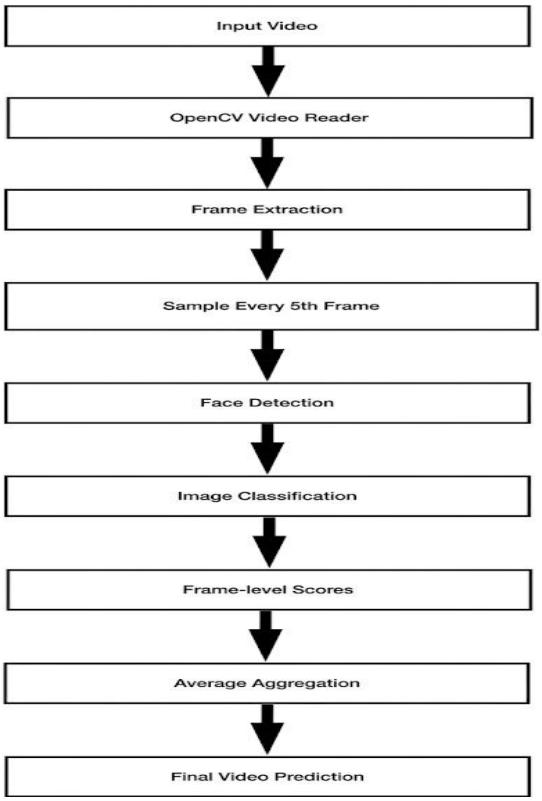


**Fig5:** ***Video Detection Pipeline***

By applying this approach, compute cost drops markedly, yet prediction quality remains consistent.

*H. Frontend Workflow*

Built with React 18, TypeScript, and Vite, the client side delivers a current, fast, and adaptable interface. It contains:

- Panel for uploading images
- Panel for uploading videos
- Cards that show results
- A view for confidence levels
- Indicators for progress while waiting
- A mechanism to manage errors

Using REST endpoints, the client connects to the server and supports an easy, operator-friendly flow.

*I. Backend API Workflow*

Using FastAPI, the server-side foundation delivers inference through scalable REST endpoints.

On the server, these operations are carried out:

- Managing incoming file uploads
- Handling temporary saved files
- Detecting faces and preparing inputs

- Running the model to infer outcomes
- Producing prediction score values
- Formatting replies as JSON

### J. *Confidence-Based Decision System*

Rather than returning strictly yes/no outputs, the framework provides confidence values. This raises clarity and helps users judge edge cases with greater ease.

Predictions are grouped by confidence into:

- Real
- Suspicious
- Fake

Overall, this method boosts real-world forensic usefulness and curbs excessively certain outputs.

## IV. IMPLIMENTATION

### A. *Implementation Overview*

Beyond the Pixel is implemented by building a scalable, efficient end-to-end deepfake detection system that can examine both still images and video. Deep learning media-forensics methods are combined with up-to-date web tooling to deliver an accessible service for identifying tampered media content.

The implementation is heavily inspired by methodologies used in the **DeepFake Detection Challenge (DFDC)** and incorporates architectural concepts from EfficientNet-based DFDC competition solutions. This work uses a face-focused categorization approach in which facial areas are cropped prior to inference, aiming to strengthen prediction consistency and limit interference from the surrounding scene.

The full system is organized into three primary tiers:

- Frontend Tier
- Backend Tier
- Deep Learning Inference Tier

User input and output display are managed by the frontend, preprocessing and API exchanges are handled by the backend, and the inference tier runs deepfake classification with EfficientNet-B7.[1]

### B. *DFDC-Based Detection Framework*

The suggested solution applies a DFDC-motivated deepfake detection workflow tailored for examining altered facial content. Its implementation approach draws from top-ranking entries in the DeepFake Detection Challenge competition.

The DFDC dataset contains thousands of manipulated and authentic videos generated using different facial synthesis techniques. These datasets are used to train deep learning models capable of identifying subtle manipulation artifacts.

The current project adopts several DFDC-inspired concepts including:

- Face-centered preprocessing
- EfficientNet-B7 classification
- Frame-based video inference
- Confidence-based prediction
- Facial crop normalization
- Score aggregation for videos

The EfficientNet-B7 classifier used in the project is designed to identify forensic inconsistencies such as:

- Facial blending artifacts
- Skin texture irregularities
- Illumination mismatches
- Eye and mouth region anomalies
- Expression synthesis inconsistencies
- Boundary transition distortions

The use of DFDC-inspired preprocessing improves the model's ability to generalize on manipulated media. [3,4]

### C. *Frontend Implementation*

The frontend of the system is implemented using **React 18**, **TypeScript**, and **Vite**. The frontend's aim is to deliver a fast-reacting, interactive UI for uploading media and viewing prediction results.

Frontend Capabilities

Within the frontend, the build provides:

- An image uploading mechanism
- A video uploading mechanism
- Support for drag and drop
- A confidence value readout
- On-the-fly prediction drawing
- A system for handling errors
- A layout that adapts to screen size
- Communication with a REST interface

The frontend connects to the FastAPI backend through non-blocking HTTP calls.

Table3: *Frontend Technologies*

| Technology | Purpose |
|---|---|
| React 18 | User Interface |
| TypeScript | Type Safety |
| Vite | Frontend Build Tool |
| CSS | Styling |
| Fetch API / Axios | Backend Communication |

*D. Backend Implementation*

FastAPI, a lightweight Python framework intended for scalable API creation, is used to build the backend foundation.[11]

The backend serves as the link joining the frontend to the deep learning inference component.

Backend Duties

On the backend side, it carries out:

- Managing uploaded files
- Storing media temporarily
- Preparing images prior to processing
- Pulling frames from videos
- Detecting faces
- Running the model for inference
- Producing confidence values
- Packaging replies as JSON
- Handling failures

The backend is organized in modules and enables inference on either CPU or GPU.

The backend also integrates:

- CORS middleware
- Device selection (CPU/GPU)
- Temporary directory management
- File cleanup system

*E. Face Detection Implementation*

For detecting and extracting faces, the project relies on MTCNN (Multi-task Cascaded Convolutional Networks).

Implemented via the facenet-pytorch toolkit, MTCNN is tasked with identifying the main face area in images and in video frames. [5,12]

This face-focused detection approach draws from DFDC preprocessing pipelines, since artifacts from manipulation typically cluster in facial areas.

Face Detection Steps

In this implementation, it executes:

1. Converting images to RGB
2. Finding facial positions
3. Deriving bounding rectangles
4. Cropping out the face region
5. Getting ready for alignment

*F. Image Preprocessing Implementation*

Once the face is extracted, the cut-out face image is processed prior to inference. For normalization and tensor setup, the project employs the Albumentations preprocessing toolkit.

*G. Deep Learning Model Implementation*

The deepfake detection component uses EfficientNet-B7 as its backbone inside the PyTorch environment.

The project uses a custom DeepFakeClassifier architecture inspired by high-performing DFDC competition solutions.

**Model Architecture**

The implementation uses:

- EfficientNet-B7 encoder
- Binary classification head
- Sigmoid activation output
- PyTorch inference engine

The classifier is initialized using pretrained EfficientNet weights and fine-tuned for manipulated media detection.

**DFDC-Influenced Classification Strategy**

A DFDC-style design is used for the classifier, treating face crops as separate inputs rather than classifying entire frames.

The network is trained to recognize tampering cues, including:

- Uneven skin detail

- Warping from facial compositing
- Lighting mismatches
- Unnatural eye-motion patterns
- Edge and seam artifacts
- Artificial-looking expressions

With this method, the model generalizes better across altered media examples.

Table4: *Model Configuration*

| Parameter | Value |
|---|---|
| Model | EfficientNet-B7 |
| Framework | PyTorch |
| Input Size | 380 × 380 |
| Batch Size | 12 |
| Optimizer | SGD |
| Momentum | 0.9 |
| Weight Decay | 1e-4 |
| Epochs | 40 |
| Loss Function | Binary Cross Entropy |

### *H. Confidence-Based Prediction System*

Rather than producing only yes/no labels, the system outputs predictions with confidence values. This makes results easier to understand and helps avoid overly certain decisions.

Table5: *Prediction Categories*

| Confidence Score | Output |
|---|---|
| < 0.40 | Likely Real |
| 0.40 – 0.60 | Suspicious |
| > 0.60 | Likely Fake |

## V. RESULT

### *A. Experimental Overview*

Beyond the Pixel was assessed on deepfake examples from both images and videos, using a DFDC-like detection pipeline. The study examined how well face-focused preprocessing, EfficientNet-B7 classification, and frame-wise aggregation work for detecting manipulated media.

Uploaded content was processed through the FastAPI backend, and confidence-oriented outputs were presented in the React frontend.

### B. *Image Detection Results*

With MTCNN used to pull out faces and EfficientNet-B7 applied for classification, the image component successfully identified manipulated face pictures.

Consistency in predictions improved when the model emphasized facial crops, since it targeted regions that often show artifacts, including:

- the zones surrounding the eyes
- the borders by the mouth
- texture across the skin
- spots where compositing happens

### C. *Video Detection Results*

To reduce computational demand, the video component pulled frames using OpenCV and assessed a single frame for each set of five.

Stability in outputs rose because the method pooled information across multiple frames.

The system demonstrated:

- steady prediction behavior
- more efficient video processing
- reduced computing needs
- practical inference in real-world settings

### D. *Performance Analysis*

Under the proposed DFDC-inspired method, detection stability improved by applying:

- face-centered preprocessing
- EfficientNet-B7 for classification
- Albumentations for preprocessing
- confidence-directed prediction selection
- frame-level score aggregation

Connecting a FastAPI backend to a React frontend made the rollout workflow accessible and able to grow.

### E. *Overall Discussion*

Results from testing suggest that a strong deepfake detection approach for both images and videos is achieved by combining MTCNN-driven face cropping, Albumentations-based preprocessing,

EfficientNet-B7 classification, and DFDC-like techniques.

- The proposed approach provides:

- more dependable inference
- scalable deployment
- practical, field-focused forensics
- support for real-world deepfake analysis scenarios

## REFERENCES

[1] Tan, M., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *Proceedings of the International Conference on Machine Learning (ICML)*.

[2] Rossler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., & Nießner, M. (2019). FaceForensics++: Learning to detect manipulated facial images. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[3] Dolhansky, B., Bitton, J., Pflaum, B., Lu, J., Howes, R., Wang, M., & Canton Ferrer, C. (2020). The DeepFake Detection Challenge (DFDC) dataset. *arXiv preprint arXiv:2006.07397*.

[4] Seferbekov, S. (2020). *DeepFake detection challenge solution* [GitHub repository].

[5] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multi-task cascaded convolutional networks. *IEEE Signal Processing Letters, 23*(10), 1499–1503.

[6] Buslaev, A., et al. (2020). Albumentations: Fast and flexible image augmentations. *Information, 11*(2).

[7] Paszke, A., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*.

[8] Bradski, G. (2000). The OpenCV library. *Dr. Dobb's Journal of Software Tools*.

[9] Facebook AI. (n.d.). DeepFake Detection Challenge dataset. https://ai.facebook.com/datasets/dfdc/

[10] React. (n.d.). *React documentation*. https://react.dev/

[11] FastAPI. (n.d.). *FastAPI documentation*. https://fastapi.tiangolo.com/

[12] Timesler. (n.d.). *facenet-pytorch documentation*. https://github.com/timesler/facenet-pytorch