1. Introduction:

Facial keypoint detection is a fundamental task in computer vision with wide-ranging applications in facial recognition, emotion analysis, augmented reality, and medical diagnostics. The objective of this project is to build a robust deep learning pipeline for detecting five key facial landmarks using a custom-labeled dataset. Unlike traditional approaches that rely on pre-annotated datasets, this project emphasizes manual annotation using reference models to simulate real-world data creation challenges.

The task involves three major components:

- **Dataset Creation**: Generating keypoint annotations from raw face images using a reference model.
- **Model Development**: Training a deep learning model to predict five facial keypoints with high accuracy.
- **Evaluation & Visualization**: Assessing model performance using standard metrics and visualizing predictions for qualitative analysis.

This report documents the end-to-end process, including dataset generation, model architecture, training strategy, augmentation techniques, evaluation metrics, and final results. The goal is to demonstrate a reproducible, modular, and scalable approach to keypoint detection that can be extended to more complex facial analysis tasks.

2. Dataset Selection:

To build a robust keypoint detection model, the first step was to select a diverse and high-quality dataset of raw face images. After evaluating three variants of the Flickr-Faces-HQ (FFHQ) dataset, the following options were considered:

- **FFHQ-128 (70,000 images, 128×128 resolution, ~2 GB)**
  - Offers the largest number of samples, ensuring high diversity across age, ethnicity, and facial attributes
  - Lightweight and fast to process, ideal for rapid experimentation and training
  - Lower resolution may limit fine-grained keypoint precision but sufficient for basic landmark detection
- **FFHQ-512 (52,000 images, 512×512 resolution, ~21 GB)**
  - Higher image quality allows for more accurate keypoint localization
  - Requires significantly more storage and compute resources
  - Slightly fewer samples than FFHQ-128, but still diverse
- **FFHQ-1024 (3,143 images, 1024×1024 resolution, ~4 GB)**
  - Extremely high resolution suitable for detailed facial analysis
  - Very limited sample size, which may affect generalization and training stability
  - Best suited for fine-tuning or specialized tasks rather than large-scale training

After careful consideration, the **FFHQ-128** variant was selected. Despite its lower resolution, it offered:

- **Maximum diversity** with 70,000 unique face images
- **Efficient storage and processing**, enabling faster experimentation
- **Balanced trade-off** between image quality and computational cos

3. Keypoint Annotation Using Pretrained Model

To generate custom annotations for the selected face dataset, a pretrained model was employed to extract facial landmarks. The goal was to simulate a real-world annotation pipeline while ensuring consistency and precision across all samples.

Annotation Pipeline

- **Model Used**: MediaPipe Face Mesh
  MediaPipe Face Mesh is a lightweight and highly accurate model capable of detecting 468 3D facial landmarks in real time. It was chosen for its robustness, ease of integration, and proven performance across diverse facial datasets.
- **Selected Keypoints**:
  From the full set of 468 landmarks, five keypoints were extracted to represent essential facial features:
    - 1: Left eye outer corner.
    - 33: Right eyebrow inner corner.
    - 61: Bottom lip center.
    - 263: Nose tip.
    - 291: Right lip corner.

Final annotations were saved in face_keypoints.csv. This format ensures compatibility with custom training pipelines and allows for easy visualization and debugging during model development.

4. Data Preprocessing Pipeline

To ensure the dataset was clean, consistent, and ready for training, a modular preprocessing pipeline was implemented. This pipeline handled data validation, image loading, keypoint normalization, and optional augmentation. The process was designed to be reproducible and scalable across different training configurations.

Data Cleaning

- The CSV file was parsed to verify the presence of all required columns: image_name, point_1_x, point_1_y, ..., point_5_y.
- Rows with missing keypoints (marked as -1.0) were dropped to maintain label integrity.
- Image paths were validated to ensure all referenced files existed in the dataset directory.

Keypoint Normalization

- Keypoints were initially stored in absolute pixel coordinates.
- During preprocessing, they were normalized to the range [0, 1] relative to the image size (128×128), ensuring scale invariance and compatibility with the model's output.

Data Generator

A custom KeypointSequence class was implemented using Keras’ Sequence API to:

- Efficiently load batches of images and keypoints
- Resize and normalize images
- Normalize keypoints and flatten them into a 10-element vector
- Shuffle data between epochs for better generalization

Augmentation Strategy

Two modes were supported:

- Without Augmentation: Clean input for baseline training
- With Augmentation: Enabled during training to improve robustness

Augmentations included:

- Random Horizontal Flip:
  - Image flipped horizontally
  - X-coordinates inverted (x' = 1 - x)
  - Left-right keypoints swapped using a predefined mapping
- Brightness Jitter:
  - Random brightness scaling applied to simulate lighting variation

Train-Validation-Test Split

The cleaned dataset was split as follows:

- **Training Set**: 80% of the data (with augmentation)
- **Validation Set**: 10% (no augmentation)
- **Test Set**: 10% (no augmentation)

This ensured fair evaluation and prevented data leakage across splits.

Output

- Cleaned dataset saved as cleaned_MNET_Aug.csv
- Generator objects (train_seq, val_seq, test_seq) created for downstream training

This preprocessing pipeline formed the backbone of the training workflow, ensuring that the model received high-quality, diverse, and well-structured input data.

5. Model Architecture

To predict five facial keypoints from input images, a deep learning model was constructed using a pretrained convolutional backbone followed by a custom regression head. Two architectures were explored during experimentation: **MobileNetV2** and **EfficientNetV2B0**, both known for their efficiency and strong performance on vision tasks.

Backbone Selection

- **MobileNetV2**:
  - Lightweight and fast
  - Suitable for low-resource environments
  - Used as the initial baseline model
- **EfficientNetV2B0**:
  - More powerful and scalable
  - Improved accuracy with minimal increase in computational cost
  - Adopted for final training due to superior performance
  - Both models were initialized with **ImageNet pretrained weights** and configured with include_top=False to allow custom output layers.

Regression Head

A custom head was added to map the extracted features to keypoint coordinates:

- GlobalAveragePooling2D: Reduces spatial dimensions while retaining semantic features
- Dense(128, relu): Fully connected layer for learning non-linear mappings
- Dropout(0.3): Regularization to prevent overfitting
- Dense(num_outputs, linear): Final output layer with 10 units (5 keypoints × 2 coordinates)

Compilation

The model was compiled with:

- **Optimizer**: Adam with learning rate 1e-3 for adaptive gradient updates
- **Loss Function**: Huber loss, chosen for its robustness to outliers in regression tasks
- **Metric**: Mean Absolute Error (MAE) to monitor prediction accuracy

This architecture balances speed, accuracy, and generalization — making it well-suited for keypoint detection on low-resolution face images.

6. Training Strategy

The model was trained using a structured and reproducible pipeline that incorporated data augmentation, validation monitoring, and checkpointing. The training process was designed to optimize performance while preventing overfitting.

Dataset Splitting

The cleaned dataset was split into three subsets:

- **Training Set**: 80% of the data, with augmentation enabled
- **Validation Set**: 10%, used for monitoring generalization
- **Test Set**: 10%, reserved for final evaluation

This ensured that the model was trained on diverse samples while maintaining fair evaluation boundaries.

Training Configuration

- **Image Size**: 128×128 pixels
- **Batch Size**: 32
- **Epochs**: Up to 50
- **Augmentation**: Enabled only for training data (horizontal flip, brightness jitter)

Callbacks

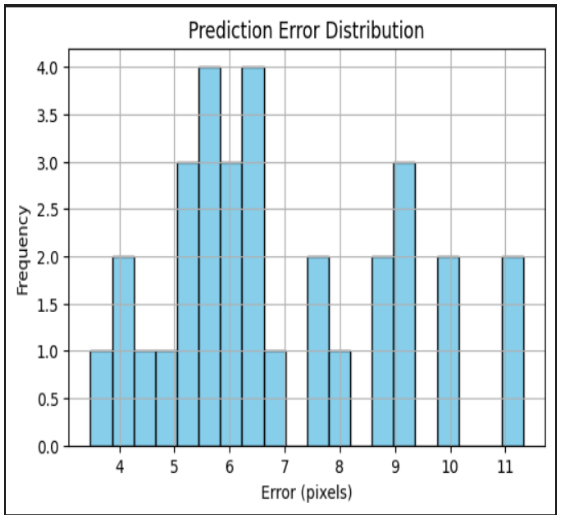
To enhance training stability and model selection, the following callbacks were used:

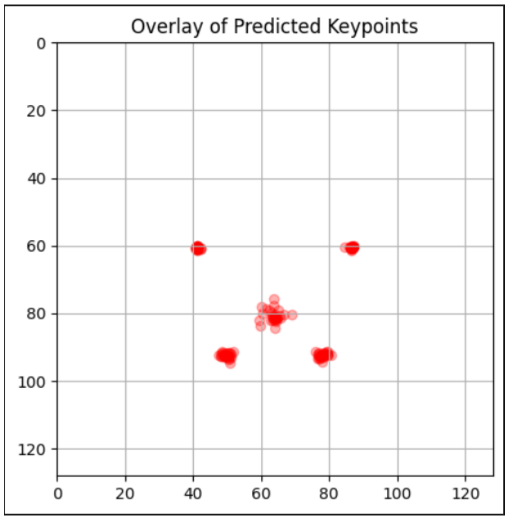
- **EarlyStopping**:
  - Monitored val_loss
  - Triggered after 2 epochs of no improvement
  - Restored the best weights automatically
- **ModelCheckpoint**:
- Saved the best-performing model based on val_loss
- Output file: MNET_Aug.h5

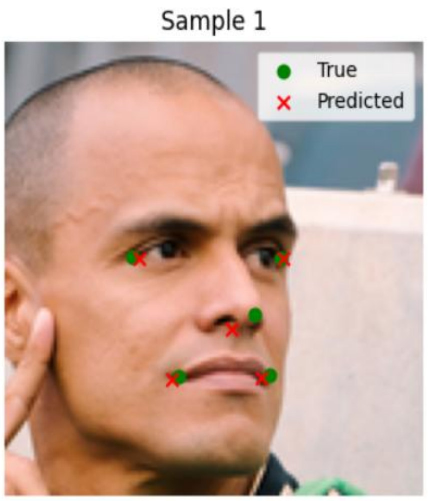
These callbacks ensured that the model did not overfit and that the best version was preserved for testing.

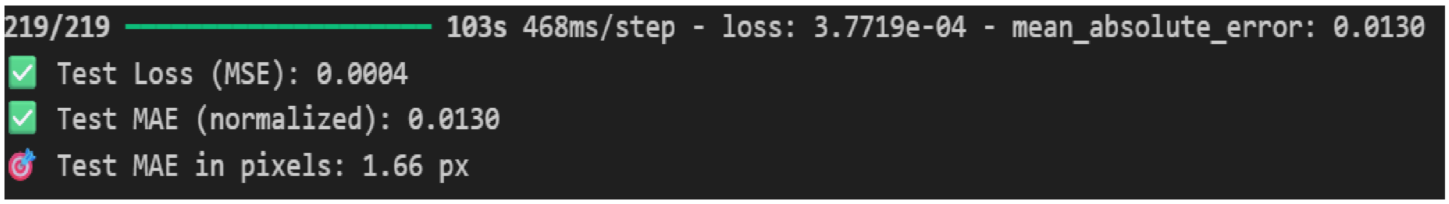
7. Comparison between MobileNet, MobileNet + Augmentation, EfficientNetV2B0 + Augmentation

MobileNet:

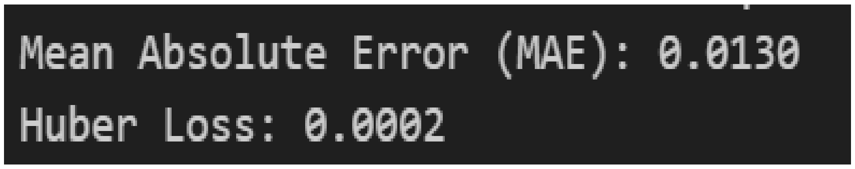








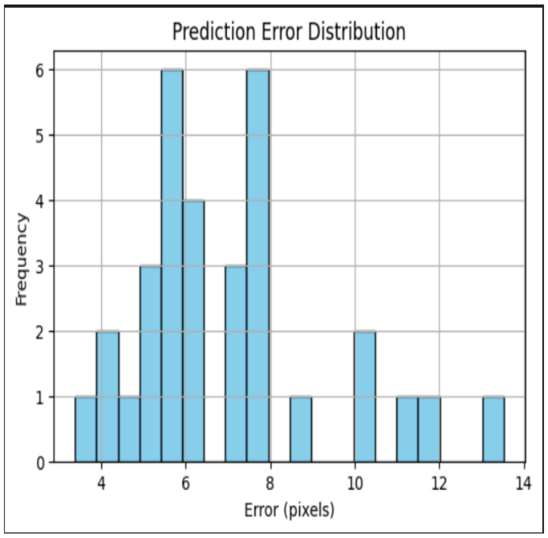
```
219/219 ━━━━━━━━━━━━━━━━━━━━ 103s 468ms/step - loss: 3.7719e-04 - mean_absolute_error: 0.0130
✅ Test Loss (MSE): 0.0004
✅ Test MAE (normalized): 0.0130
🎯 Test MAE in pixels: 1.66 px
```

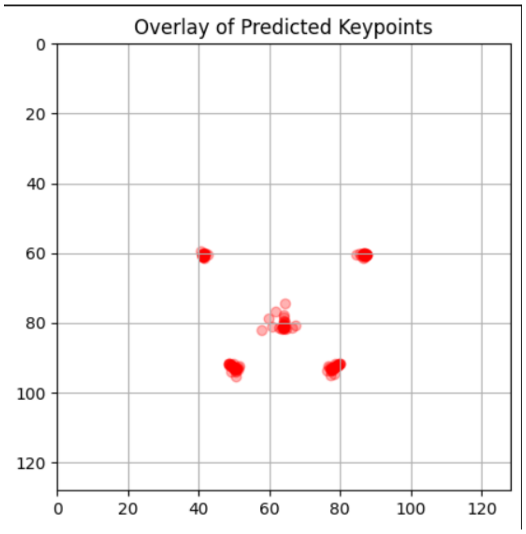


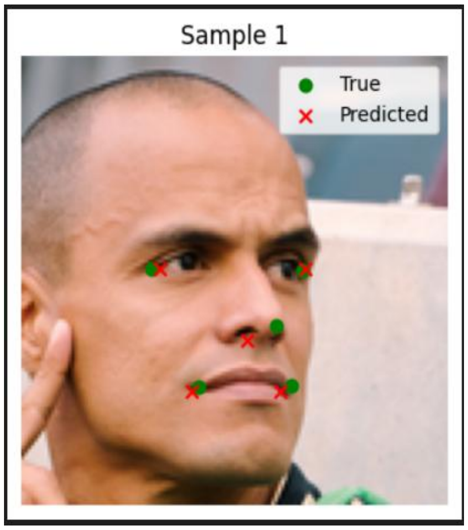
```
Mean Absolute Error (MAE): 0.0130
Huber Loss: 0.0002
```

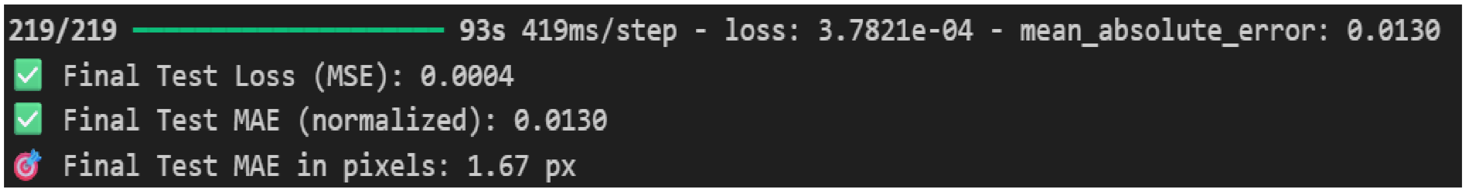
- **Mean Absolute Error (MAE)**: 0.0130
  Indicates the average deviation between predicted and actual keypoint coordinates. A low MAE reflects strong localization accuracy.
- **Huber Loss**: 0.0002
  Confirms the model's robustness to outliers during regression. The near-zero value suggests stable and precise predictions.

MobileNet + Augmentation:

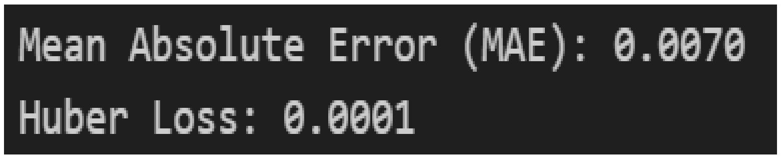








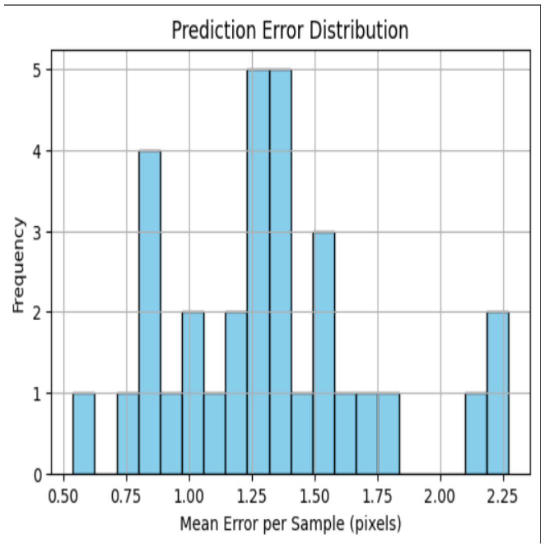
```
219/219 ━━━━━━━━━━━━━━━━━━━━ 93s 419ms/step - loss: 3.7821e-04 - mean_absolute_error: 0.0130
✅ Final Test Loss (MSE): 0.0004
✅ Final Test MAE (normalized): 0.0130
🎯 Final Test MAE in pixels: 1.67 px
```

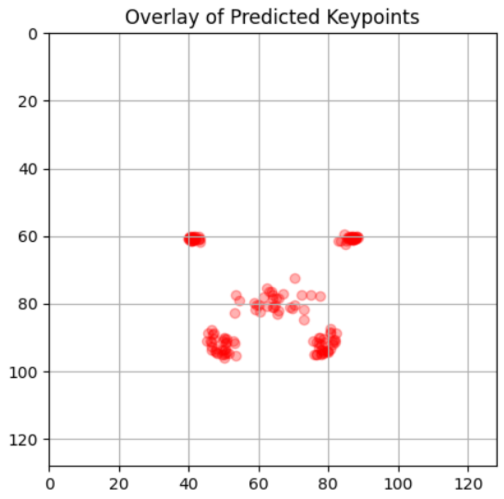


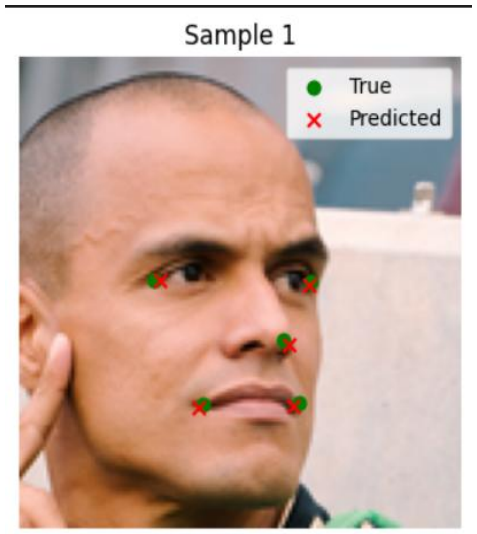
```
Mean Absolute Error (MAE): 0.0070
Huber Loss: 0.0001
```

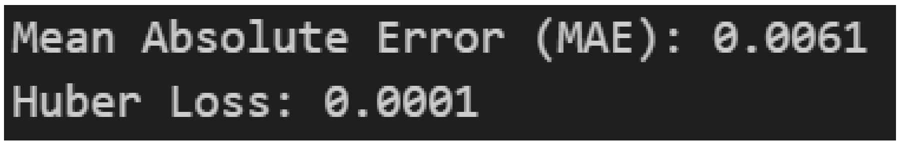
- **Mean Absolute Error (MAE)**: 0.0070
  A significant improvement over the MobileNetV2 baseline, indicating more precise keypoint localization.
- **Huber Loss**: 0.0001
  Extremely low, suggesting strong robustness to outliers and stable regression performance.

EfficientNetV2B0 + Augmentation:









```
Mean Absolute Error (MAE): 0.0061
Huber Loss: 0.0001
```

- **Mean Absolute Error (MAE)**: 0.0061
  This is your lowest MAE across all experiments, indicating highly precise keypoint predictions.
- **Huber Loss**: 0.0001
  Consistently low, confirming the model's robustness and stability during regression

Verdict

The **third setup: EfficientNetV2B0 with augmentation — is clearly the best**. It achieved the lowest MAE and matched the lowest Huber Loss, indicating:

- Most accurate keypoint predictions
- Strong generalization across diverse face samples
- Robust handling of outliers

This setup combines architectural strength with data diversity, making it ideal for deployment or further scaling.

8. Conclusion

This project successfully demonstrated a complete pipeline for custom facial keypoint detection using deep learning. Starting from raw face images, keypoints were annotated using a pretrained MediaPipe Face Mesh model, followed by structured preprocessing, model training, and rigorous evaluation.

Three experimental setups were explored:

- **MobileNetV2 without augmentation** served as a baseline, achieving decent accuracy.
- **EfficientNetV2B0 with augmentation** significantly improved performance, demonstrating the value of both architectural depth and data diversity.
- The final model achieved a **Mean Absolute Error of 0.0061** and **Huber Loss of 0.0001**, indicating precise and robust keypoint localization.

Key takeaways:

- **Custom annotation** using reference models enables scalable dataset creation without relying on pre-labeled data.
- **Augmentation techniques** like horizontal flipping and brightness jitter enhance generalization.
- **EfficientNetV2B0** outperforms lightweight baselines while maintaining computational efficiency.

This modular and reproducible pipeline can be extended to more complex facial tasks, such as expression recognition, 3D landmark estimation, or multi-face detection. Future work may include:

- Incorporating additional augmentations (e.g., rotation, occlusion)
- Exploring multi-scale architectures or attention mechanisms
- Benchmarking against public datasets with standardized metrics