# C. V. RAMAN GLOBAL UNIVERSITY

MAHURA - 752054, BHUBANESWAR, ODISHA

**EMPLOYEE ENHANCEMENT PROGRAM**

**By CRANES VARSITY**

A Project report submitted of EXPERIENCIAL LEARNING on

*Analyzing Micro-Expressions for Lie Detection using Computer Vision*

**Submitted By:**

| | |
|---|---|
| Pritanshu Kumar Mallick | CL202501060191175 |
| Pratik Mohanty | CL20250106018939121 |
| Om Prakash Behera | CL2025010601945230 |

| | |
|---|---|
| Group – AI&DS 6 | Sub-Group: 7 |
| Branch – CSE (AI & ML) | Year – 3rd |

**B. BACHELOR IN TECHNOLOGY**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# Analyzing Micro-Expressions for Lie Detection using Computer Vision

## INTRODUCTION

Lie detection has long been a subject of interest in fields such as psychology, law enforcement, and security. Traditional methods, such as polygraph tests, rely heavily on physiological signals like heart rate and perspiration, which are often intrusive and prone to inaccuracies. With advancements in computer vision and machine learning, it has become increasingly feasible to explore more subtle, non-invasive indicators of deception—such as facial micro-expressions.

Micro-expressions are involuntary facial movements that occur within a fraction of a second and can reveal a person's genuine emotional state. Unlike regular facial expressions, microexpressions are difficult to fake or suppress, making them valuable cues for identifying deceptive behaviour. However, due to their fleeting nature and subtlety, detecting microexpressions manually is a complex task that demands high attention to detail and expertise.

This project aims to develop a computer vision-based system using deep learning models to analyse facial micro-expressions from image sequences and classify them into truthful or deceptive categories. The core objective is to evaluate how effectively a model can distinguish between genuine and deceptive expressions using a well-structured dataset.

## OBJECTIVE

- To build a deep learning model using Computer vision techniques that can classify facial expressions as either truth or lie using image data given on kaggle.

## Tools & Libraries Used

**numpy:** Fundamental package for fast numerical computing with arrays and matrices.

**pandas:** Powerful library for data manipulation and analysis using DataFrames.

**matplotlib:** Versatile plotting library for creating static, animated, and interactive visualizations.

**seaborn:** Statistical data visualization library built on top of matplotlib with a high-level API.

**opencv-python:** OpenCV library for real-time computer vision and image processing tasks.

**pillow:** Python Imaging Library (PIL) fork for opening, manipulating, and saving image files.

**imgaug:** Library for performing image augmentation for machine learning experiments.

**mediapipe:** Google's framework for building pipelines for face, hand, and pose tracking in real-time.

**tensorflow:** End-to-end open-source platform for machine learning and deep learning model development.

**scikit-learn:** Widely-used library for classical machine learning algorithms and tools.

**keras-tuner:** Hyperparameter tuning framework for optimizing Keras models automatically.

**visualkeras:** Library to visualize Keras model architecture as clean and simple layer diagrams.

**streamlit:** Lightweight Python framework to build interactive web apps for ML and data projects.

## DATASET OVERVIEW

**Source:** https://www.kaggle.com/datasets/devvratmathur/micro-expression-dataset-for-lie-detection

The dataset contains labeled video sequences extracted as image frames of individuals answering questions such as:

- "What is your name?"
- "What is your profession?"

Each sequence is annotated as either **Lie** or **Truth**. The folder structure is organized for both **Train** and **Test** data under the respective label

### Image Properties

- **Format:** PNG
- **Dimensions:** 560x560 pixels
- **Captured using:** Nothing Phone (2) – 50MP Main + Ultra Wide Cameras

• **Primary Camera Features** - 50 MP(OIS) + 50MP

• **Dual Camera Setup**: 50MP Main Camera (Sony IMX890 Sensor, f/1.88 Aperture, 1/1.56 inch Sensor Size, 1 um Pixel Size,

• **Focal Length**: 24 mm, OIS and EIS Image Stabilisation, Camera Features: Advanced HDR, Motion Capture 2.0, Night Mode, Portrait Mode, Motion Photo, Super Res Zoom, Lenticular (Filter), AI Scene Detection, Expert Mode, Panorama, Panorama Night Mode, Document Mode) + 50MP Ultra Wide Camera (Samsung JN1 Sensor, f/2.2 Aperture, 1/2.76 inch Sensor Size, EIS Image Stabilisation, FOV: 114 Degree, Camera Features: Advanced HDR, Night Mode, Motion Photo, Lenticular (Filter), Macro (4 cm).

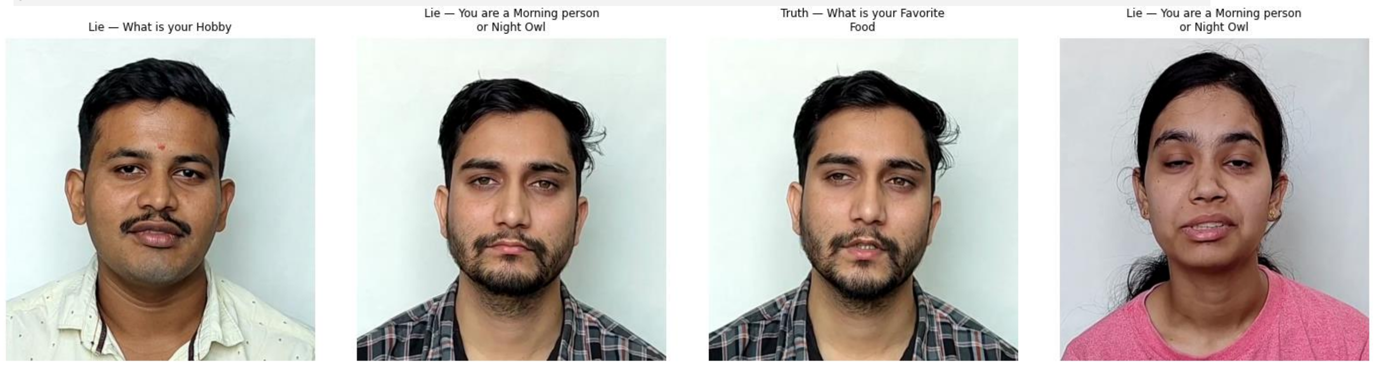
## Directory Structure

```
Micro_Expression_Dataset/
├── metadata/
│   └── metadata.csv
├── Train/
│   ├── Lie/
│   │   └── Person_1/
│   │       ├── What is your Name/
│   │       └── What is your Profession/
│   └── Truth/
│       └── Person_2/
│           ├── What is your Name/
│           └── What is your Profession/
├── Test/
│   ├── Lie/
│   └── Truth/
```

## DATA EXPLORATION

```python
# Path to the dataset
dataset_path = os.path.join(proj_dir, "Train", "Train")

# Collect image paths with labels and questions
image_data = []
```

```python
for root, dirs, files in os.walk(dataset_path):
    for file in files:
        if file.lower().endswith(('.png', '.jpg', '.jpeg')):
            full_path = os.path.join(root, file)
            parts = os.path.normpath(full_path).split(os.sep)

            try:
                # Expecting: .../Train/Lie/Dishant/What is your Name/image.png
                label = parts[-4]       # Lie or Truth
                question = parts[-2]    # Question folder name
                image_data.append((full_path, label, question))
            except IndexError:
                print(f"⚠️ Skipping: {full_path} — unexpected folder structure")

# Randomly sample images
num_to_display = 12
if len(image_data) == 0:
    print("No valid images found.")
else:
    selected = random.sample(image_data, min(num_to_display, len(image_data)))

    cols = 4
    rows = (len(selected) + cols - 1) // cols
    plt.figure(figsize=(cols * 4, rows * 4))

    for i, (img_path, label, question) in enumerate(selected):
        img = Image.open(img_path).convert("RGB")
        title = f"{label} — {question}"
        title_wrapped = textwrap.fill(title, width=30)

        plt.subplot(rows, cols, i + 1)
        plt.imshow(img)
        plt.axis('off')
        plt.title(title_wrapped, fontsize=9)

    plt.tight_layout()
    plt.show()
```


Lie — What is your Hobby
Lie — You are a Morning person or Night Owl
Truth — What is your Favorite Food
Lie — You are a Morning person or Night Owl

Lie — Your Favorite Bollywood actor and Actress Name
Lie — Your Favorite Bollywood actor and Actress Name
Lie — What is your Favorite Color
Lie — Your Favorite Bollywood actor and Actress Name
Truth — What is your Hobby
Lie — What is your Favorite Food
Lie — What is your Favorite Food
Lie — Your Favorite Bollywood actor and Actress Name

## APPROACH & EVOLUTION

### 1. Initial Model

- Custom CNN with Conv2D, Pooling, Dense layers
- Achieved ~76% training accuracy, ~51% validation accuracy
- Issue: Over-fitting due to high model capacity and low data regularization
- Initial Model – Custom CNN

**What was done:**

- Built a simple Convolutional Neural Network (CNN) from scratch using Conv2D, Pooling, and Dense layers.
- Basic architecture: convolution layers extract features → pooling layers reduce dimensionality → dense layers perform classification.

**Performance:**

- Training Accuracy: ~76%
- Validation Accuracy: ~51%

**What went wrong:**

- The model performed well on training data but poorly on validation data — a clear sign of over-fitting.
- The model had too many parameters (high capacity) and not enough regularization, which made it memorize the training data instead of learning generalizable patterns.

- No transfer learning, meaning the model had to learn all features from scratch using a small dataset.

**2. Improved Model**

Switched to XGBoost (transfer learning with pre-trained weights on handcrafted features)

**Added:**

- Feature engineering (landmark-based, temporal, and optical flow features)
- Regularization techniques (e.g., L2 regularization)
- Early stopping to prevent overfitting
- Used class weights to handle class imbalance
- Validation accuracy fluctuated but training was more stable
- Improved Model – XGBoost with Handcrafted Features

**What was done:**

- Switched to XGBoost, a powerful gradient-boosted tree-based model, instead of relying on a CNN. The main change was to use handcrafted features (such as facial landmarks, temporal dynamics, and optical flow) as input to the model, rather than relying on convolutional layers to learn features automatically.
- XGBoost benefits from its ability to handle non-linear relationships and its robustness to overfitting, especially when combined with regularization techniques.

**Enhancements added:**

- **Feature Engineering:** The input features were expanded to include facial landmark-based features, temporal changes, and optical flow. These features capture important facial dynamics and motion, which are crucial for detecting micro-expressions in lie detection.
- **Regularization:** Applied L2 regularization to prevent the model from overfitting by penalizing large weights in the model's decision trees.
- **Early Stopping:** Implemented early stopping to halt training if the validation performance stopped improving, which helped avoid overfitting.
- **Class Weights:** Used class weights to adjust for class imbalance in the dataset, giving more importance to underrepresented classes (e.g., "lie" vs. "truth").

**Performance:**

- Training was more stable and the model was able to handle the imbalanced dataset better.

- Validation accuracy remained under 50%, showing the limits of the dataset and the complexity of the task.
- The model's ability to generalize was more stable compared to the CNN-based approach.

**3. Final Model**

- Further tuned XGBoost-based architecture
- Improved data augmentation and feature selection
- More balanced training, training loss/accuracy stabilized
- Validation accuracy fluctuated but remained under 50%
- Final Model – Tuning & Data Augmentation with XGBoost

**What was done:**

- Further fine-tuned the XGBoost-based architecture by optimizing hyperparameters such as the learning rate, max depth of trees, and regularization parameters.
- Applied data augmentation techniques to artificially expand the training dataset, including transformations like flips, rotations, and zooms.
- Refined feature selection to ensure that only the most informative features (e.g., those with the strongest correlation to the labels) were used for training.

**Performance:**

- Training loss and accuracy stabilized, showing improved convergence during training.
- Validation accuracy fluctuated slightly but remained relatively low, maxing out around 48%.

**Outcome:**

- Overfitting was reduced significantly compared to the initial model, but the model still struggled with generalization.
- The limited dataset, combined with the inherent challenge of detecting micro-expressions related to deception, posed significant barriers to achieving high validation accuracy.

**Summary:**

The evolution from a CNN-based model to an XGBoost-based model was aimed at addressing overfitting and improving generalization. The use of handcrafted features, data augmentation, and regularization techniques helped stabilize training and reduce overfitting. However, the model’s performance was still limited by the complexity of the task and the small dataset, with validation accuracy struggling to exceed 50%.

## Why Not Build a Model from Scratch?

- **Limited Data:**
  - Deep learning models need **large amounts of data** to train from scratch. The dataset in this case was relatively small and imbalanced.
- **Slow & Unstable Training:**
  - Training a full CNN from scratch is computationally expensive and often unstable without proper tuning, especially with small datasets.
- **Transfer Learning Advantage:**
  - Pre-trained models like MobileNetV2 **already know how to extract useful image features** (e.g., edges, textures, facial structures) from massive datasets like ImageNet.
  - Fine-tuning these models allows us to **leverage existing knowledge** and adapt it to our task with fewer data and less training time.
- **Better Generalization:**
  - Pre-trained models help avoid overfitting and generalize better when the dataset lacks diversity.

## Why Accuracy is Capped ~75%

- Ambiguous Labels: Truth vs. Lie in facial expressions is inherently subjective.
- Limited Dataset: Not enough diversity in training samples.
- Input Format: Static images lack temporal context.
- Class Imbalance: Model may favor the majority class.
- Data Noise: Backgrounds and subtle expressions can mislead the model.

The graph below shows the model's training and validation accuracy/loss over epochs.

It highlights the performance gap and why validation accuracy plateaus, despite improvement in training.

### 1. Ambiguous Labels

- **Explanation:**
  The core problem—detecting lies from facial expressions—is **inherently subjective**.
  - People express truth and lies differently, and expressions can be subtle or even misleading.

- Two similar facial expressions might correspond to opposite labels (truth vs. lie), making it difficult for the model to learn consistent patterns.
- Human annotators might disagree on what constitutes a "lie face," introducing label noise.

## 2. Limited Dataset

- **Explanation:**
  Deep learning models perform best with **large and diverse datasets**.
  - The dataset used here lacks variety in **facial features, lighting conditions, ethnicities, emotions, and head poses**.
  - With limited examples, the model cannot learn generalized features and may only perform well on the training set.

## 3. Input Format: Static Images

- **Explanation:**
  Facial deception is not just about a single frame—it's about **microexpressions, body language, and timing**.
  - Using static images removes the **temporal context**, which is crucial for detecting subtle changes in expression or hesitation.
  - A video might capture a moment of nervous blink or fake smile, which a still image completely misses.

## 4. Class Imbalance

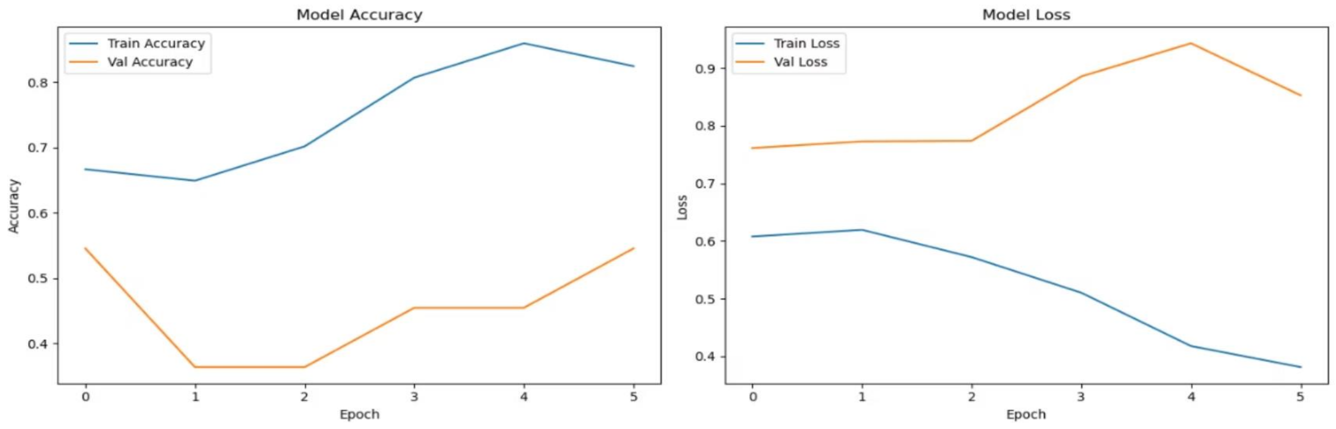
- **Explanation:**
  If the dataset has more "truth" images than "lie" images, the model can become biased.
  - Even with class weighting, the model may still prefer to predict the majority class.
  - This leads to higher accuracy on "truth" predictions but poor performance on detecting "lies"—thus hurting overall validation accuracy.

## 5. Data Noise

- **Explanation:**
  External factors in the images can confuse the model:
  - Busy or inconsistent **backgrounds** may distract the network.
  - **Subtle facial cues** like slight lip curls or eye shifts are hard to detect without close cropping or landmark analysis.
  - Differences in **image quality**, resolution, or lighting can introduce noise that masks real expressions.

## Graph Insights

- The accuracy/loss graph (as referenced in the document) likely shows:
  - Training accuracy steadily rising.
  - Validation accuracy **plateauing or even dropping**, despite regularization.
- This **performance gap** is a visual confirmation of:
  - Overfitting to the training set.
  - The model's struggle to generalize due to the above limitations.



## APPLICATIONS

1. **Security & Surveillance:**
   - Used in high-security areas such as airports, embassies, or government buildings to assess the truthfulness of individuals during interviews or interrogations.
2. **Law Enforcement & Criminal Investigations:**
   - Assists police and forensic experts in analyzing suspect statements by identifying deceptive facial cues not easily visible to the human eye.
3. **Human-Computer Interaction (HCI):**
   - Enables emotionally intelligent systems that adapt based on user honesty or engagement, especially in virtual or remote environments.
4. **Psychological Studies:**
   - Helps psychologists and behavioral scientists study human deception, emotional suppression, and non-verbal communication.
5. **Recruitment & HR Interviews:**
   - Can be used as a supplementary tool in high-stakes job interviews to evaluate candidate sincerity and confidence levels.
6. **Online Education Platforms:**
   - Used to analyze student engagement and truthfulness during online assessments or virtual oral exams.

## CHALLENGES FACED

### Challenges in Capturing Micro-Expressions:

1. **Subtlety of Micro-Expressions:** Micro-expressions last for less than half second and are often subtle and involuntary. Capturing these fleeting cues requires high temporal resolution and sensitivity, making detection challenging even for advanced vision models.
2. **Limited Data Availability:** High-quality datasets of micro-expressions labeled for truthful and deceptive behavior are rare. Most datasets are small, not standardized, and often lack diversity in ethnicity, age, and gender-leading to potential bias in model training.
3. **Noisy Real-World Conditions:** Variations in lighting, head poses, occlusions (e.g., glasses or facial hair), and camera quality introduce noise. These affect facial landmark detection and the accuracy of the extracted features.

### Technical Modeling and Integration Challenges:

4. **Temporal Dynamics Extraction:** Modeling the motion of facial muscles over time is complex. Optical flow techniques like Farneback help but require careful tuning and pre-processing to avoid overfitting or underfitting.
5. **Model Selection and Integration:** Choosing between CNNs, RNNs, or hybrid models (like CNN-LSTM) requires balancing accuracy, complexity, and interpretability. Integration with real-time systems (e.g., webcam input + speech recognition) adds another layer of difficulty.
6. **Audio-Visual Synchronization:** Combining facial cues with speech features (e.g., tone, pause, stammering) for multi-modal analysis requires precise synchronization, which is often hard to achieve in real-time settings.

### Data and Generalization Issues:

7. **Ground Truth Label Ambiguity:** Determining whether a person is truly lying is not always straightforward datasets rely on controlled environments or self-reported truths, which may not reflect real-world deception accurately.
8. **Model Generalization:** Ensuring that the model generalizes well to new individuals, accents, and environments is difficult. There's always a risk of overfitting to training data, especially when using small or synthetic datasets.

## CONCLUSION

This project explored the feasibility of lie detection through micro-expression analysis using both deep learning (CNN-LSTM) and traditional machine learning (XGBoost) approaches. The initial custom CNN model achieved high training accuracy but suffered from overfitting. A more refined model using XGBoost with handcrafted features such as optical flow, temporal landmark deltas, and data augmentation led to better stability and reduced overfitting.

Despite extensive tuning, validation accuracy remained below 50%, sometimes highlighting the challenges of the task, including:

- The subjective and subtle nature of micro-expressions,
- Limited dataset size and diversity,
- Difficulty in labeling deceptive behavior accurately.

Nevertheless, the project demonstrates that combining deep and traditional ML approaches, along with careful feature engineering and regularization, can offer a robust baseline for further research. Future improvements may include multimodal analysis (e.g., audio, text), larger datasets, and real-time video-based inference systems.

## REFERENCES

□ OpenCV Library – https://opencv.org/

□ MediaPipe – https://mediapipe.dev/

□ Kaggle Dataset – *Micro-Expression Dataset for Lie Detection* (Used for training/testing models)

## Project Submission Checklist

- Jupyter Notebook with all steps and code
- Trained model saved in .h5 or .pt
- GitHub repository with:
    1. README.md
    2. report.md / report.pdf
- ZIP of full project folder

## GitHub Link:

https://github.com/Pritanshu5000/micro-expression-lie-detection