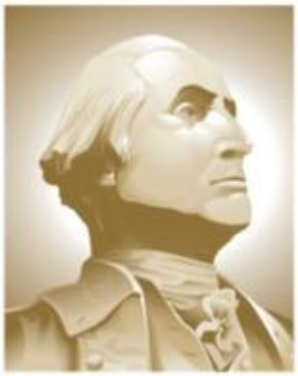


THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

DATS 6303: Deep Learning Project Report

Human Action Recognition

Group Number: 2

Akshit Reddy Palle
Ritu Patel
Dhatrija Sukasi
Ayush Meshram
Shaik Mohammad Mujahid Khalandar

Supervised by
Dr Amir Jafari

ABSTRACT

In this project, we address the challenge of multi-class Human Action Recognition (HAR) from static RGB images, an increasingly important task in applications such as surveillance systems, sports analytics, behavioral monitoring, and human–computer interaction. The variability of human poses, background clutter, object occlusion, and class similarity in real-world imagery makes robust action classification particularly difficult. To tackle these challenges, we investigate and compare the performance of five deep learning architectures: InceptionV3, ResNet-50, EfficientNet-B1, VGG-16, and a custom-built CNN classifier. Each model is trained and evaluated on a labeled public dataset containing 15 distinct human activities, using stratified data preprocessing, advanced augmentation strategies and ImageNet-based transfer learning to accelerate convergence.

Model evaluation is conducted using macro-averaged accuracy, precision, recall, and F1-score, supported by confusion matrices to analyze per-class performance, especially among visually similar seated activities such as *sitting, using laptop,* and *texting*. Performance differences across architectures are examined with respect to feature extraction depth, parameter efficiency, representational capacity, and training stability. Anticipated difficulties such as class imbalance, pose ambiguity, and overfitting are mitigated using label smoothing, frozen convolutional backbones, adaptive learning rate scheduling, and regularization techniques. The objective of this work is to design a reliable and scalable classification pipeline that systematically benchmarks several widely used CNN architectures and identifies the most effective model for static-image human action recognition in diverse, real-world environments.

CONTENTS

Section	Title	Page
1	Introduction	
2	Problem Statement	
3	Related Work	
4	Dataset Description	
5	Exploratory Data Analysis	
6	Solution and Methodology	
7	Results	
8	Conclusion	
9	Future Work	
10	References	

1. INTRODUCTION

Human Action Recognition (HAR) is an important area of computer vision that focuses on identifying what people are doing from visual data. It is widely used in real-world applications such as safety monitoring, smart cameras, sports analytics, assistive healthcare, and human computer interaction. Although many HAR systems rely on video, recognizing actions from single still images remains valuable. It enables faster, more lightweight solutions that can work on limited hardware or in environments where recording continuous video is impractical.

However, understanding human activity from a single frame is challenging. Many actions share very similar poses for example, *sitting, texting,* and *using a laptop* can look nearly identical depending on hand placement or camera angle. Lighting, background distractions, occlusion, and variation in posture make the problem even more complex. These challenges create a strong case to explore how different deep learning models interpret subtle visual cues among human actions.

In this project, we build and compare five deep learning architectures—InceptionV3, ResNet-50, EfficientNet-B1, VGG-16, and a custom CNN to understand their strengths in handling complex and visually similar human actions. We use transfer learning with ImageNet weights, apply advanced data augmentation (such as MixUp and Mosaic), and evaluate models using macro accuracy, precision, recall, F1-score, and confusion matrices. By analyzing how each architecture performs and where they fail, we aim to identify which model offers the most reliable and scalable solution for image-based human action recognition.

Ultimately, our goal is not only to build a strong HAR system, but also to provide insight into which type of deep learning architecture is best suited for real-world action classification from single images.

2. PROBLEM STATEMENT

Human Action Recognition (HAR) aims to classify activities such as running, clapping, or texting from visual data. When actions must be recognized from single still images, the task becomes more challenging because the model must rely entirely on pose, hand placement, and surrounding objects without motion clues. Factors like similar postures, lighting changes, camera angles, and background distractions make certain activities—like sitting, texting, and using a laptop—difficult to differentiate.

Traditional feature-based methods struggle with this complexity, which is why deep learning models are better suited for learning these subtle spatial patterns. In this project, we develop an image-based HAR system and compare architectures such as InceptionV3, ResNet-50, EfficientNet-B1, VGG-16, and a custom CNN to determine which model most effectively recognizes 15 human actions from static images.

3. RELATED WORK

Human Action Recognition (HAR) has evolved from traditional handcrafted features such as HOG and SIFT toward deep learning models that automatically learn spatial representations from images. Early

CNN-based approaches showed significant improvements over classical feature engineering, especially for recognizing actions with high variation in pose, lighting, and background.

Recent work highlights the effectiveness of transfer learning for still-image HAR, where models pretrained on ImageNet—such as **VGG-16, ResNet-50, InceptionV3, and EfficientNet**—have been fine-tuned to classify complex actions from static scenes. Residual networks (e.g., ResNet) improve gradient flow, Inception modules capture multi-scale spatial features, and EfficientNet scales parameters efficiently for improved accuracy-per-compute. These architectures consistently outperform shallow or handcrafted methods in both accuracy and generalization.

Motivated by these findings, our project compares multiple pretrained CNN architectures and a custom lightweight CNN to identify which model best recognizes 15 human activities from single RGB images, without relying on temporal motion cues.

4. DATASET DESCRIPTION

This project utilizes the Human Action Recognition (HAR) Dataset, originally published on Kaggle by *Meet Nagadia*. The dataset contains over 12,000 RGB images, each featuring a single person performing one clearly identifiable activity. These actions span 15 different categories such as eating, texting, clapping, running, using a laptop, and more—representing common human behaviors found in everyday environments.

Each class is neatly organized in separate folders, which makes it well-structured for supervised deep learning tasks. The dataset is intentionally diverse, containing images with variations in lighting, background context, clothing, camera angles, and poses, ensuring that the models learn robust spatial features rather than memorizing backgrounds or textures. This diversity reflects realistic conditions encountered in surveillance and human–computer interaction systems.

To ensure fair and balanced learning, the dataset was divided into training, validation, and internal test sets, maintaining equal representation across all 15 classes. Before feeding into the neural networks, standard preprocessing techniques were applied—including image resizing, normalization, and augmentation such as horizontal flips and color perturbations—to improve generalization and reduce overfitting. These steps help the models handle natural variations in body posture and scene context more effectively.

Overall, the HAR dataset provides an ideal benchmark for evaluating different deep learning architectures on static action recognition. Its combination of size, diversity, and clean class labels makes it suitable for comparing both pretrained models and custom-built CNNs.

5. EXPLORATORY DATA ANALYSIS (EDA)

Before model development, we performed a brief exploratory analysis to understand the dataset's content and ensure data integrity. The HAR dataset contains 12,600 RGB images evenly distributed across 15 human action classes, such as calling, clapping, drinking, running, and using a laptop. A class count check confirmed that each

category contains exactly 840 images, ensuring a perfectly balanced dataset with no missing labels. Visual inspection of sample images revealed realistic variations in pose, viewpoint, background clutter, lighting, and scene context. These observations verified the dataset's suitability for deep learning while highlighting challenges such as inter-class similarities (e.g., sitting vs. using laptop**)** and occlusions that can affect recognition accuracy. Insights from EDA directly motivated the use of strong augmentations and regularization strategies during model training.

A. Class distribution

All 15 action classes contained exactly 840 images, confirming perfect balance.

B. Sample Images Inspection

A set of random images was visualized to confirm action clarity and proper labeling.

These EDA steps ensure data integrity and reveal no missing files, corrupted samples, or class imbalances — providing a reliable foundation for model development.

6. SOLUTION AND METHODOLOGY

To build an accurate Human Action Recognition (HAR) classifier for 15 action categories, we evaluated five convolutional neural network architectures with varying depths, feature extraction strategies, and

computational efficiency. Our methodology focused on adapting pretrained ImageNet models to static action classification using transfer learning, along with training a custom lightweight network to benchmark performance and complexity.

1. **Baseline CNN**

   To provide a comparative baseline, a lightweight Custom CNN was trained from scratch. This allows us to judge whether specialized hand-crafted architectures can match pretrained networks in HAR tasks, while also testing the trade-offs between simplicity, training cost, and accuracy.

2. **ResNet-50**

   ResNet-50 employs residual skip connections, enabling deeper architectures without vanishing gradients. It learns high-level discriminative features across complex scenes, particularly useful for distinguishing fine differences between similar actions (e.g., sitting vs. sleeping).

3. **EfficientNet-B1**

   EfficientNet introduces compound scaling, balancing model width, depth, and resolution. This makes it both computationally efficient and highly accurate on image classification tasks. It helps evaluate whether resource-optimized models can perform competitively against heavier architectures.

4. **InceptionV3**

   InceptionV3 was selected due to its multi-scale feature extraction using parallel convolution blocks that capture small and large spatial patterns. Its reduced parameter design makes it efficient for high-resolution inputs, making it a strong candidate for modeling pose and contextual cues in complex scenes.

5. **VGG-16**

   VGG-16 serves as a classical CNN baseline with uniform 3×3 convolution stacks. Although computationally heavy, it offers robust feature extraction, helping assess how traditional deep CNNs perform compared to modern optimized architectures.

1. Baseline CNN

1.1 Model Architecture

BaselineCNN:

- Feature Extractor: 5 convolutional blocks (3×3 conv → BatchNorm → ReLU → MaxPool).
- Classifier: Adaptive average pooling → flatten → FC(512→256) → dropout (0.5) → FC(256→15).
- Key Features: BatchNorm and dropout for stability; adaptive pooling for input flexibility.
- Output: Softmax probabilities with label smoothing to reduce overconfidence.

1.2 Experimental Setup

Setting	Value
Optimizer	AdamW
Learning Rate	3e-4
Batch Size	32
Epochs	40
Loss	CrossEntropy + Label Smoothing
Scheduler	Warmup Cosine Annealing

1.3 Preprocessing and Data Augmentation

- The images in the dataset vary in resolution and aspect ratio, so a consistent preprocessing pipeline was applied before training. All images were:
- Resized and center-cropped to 240 × 240 pixels
- Converted to RGB tensors
- Normalized using standard ImageNet mean and standard deviation

To improve generalization and reduce overfitting, I used strong data augmentation during training, including:

- Random resized cropping
- Horizontal flips
- Small rotations and affine translations
- Slight perspective distortions
- Color jittering (brightness, contrast, saturation, and hue)
- In addition to these spatial augmentations, I also applied Mixup regularization with $\alpha = 0.2$, which linearly combines pairs of images and labels. This encourages the model to learn smoother decision boundaries and makes it less sensitive to noise.

Data Augmentation + Class Balance

```
transform = transforms.Compose([
   transforms.RandomResizedCrop(256, scale=(0.75, 1.0)),
   RandAugment(num_ops=2, magnitude=9),
   transforms.RandomHorizontalFlip(p=0.5),
   transforms.ToTensor(),
   transforms.Normalize([0.485, 0.456, 0.406],
                        [0.229, 0.224, 0.225]),
```

```
    RandomErasing(p=0.25, scale=(0.02, 0.2))
])
```

- Prevents overfitting
- Makes model robust to zoom, blur, lighting variations

1.4 Training Strategy

- Class Imbalance Handling via Weighted Random Sampler
- To address dataset imbalance where classes like "texting" and "drinking" had more samples than "laughing" or "hugging", I implemented a weighted sampling strategy:
- This ensured fair learning of minority classes by assigning higher sampling probabilities to underrepresented classes, preventing the model from being biased toward majority classes.

The scheduler employs:

Warmup phase (3 epochs): Gradually increases the learning rate from zero to the initial value, avoiding unstable early gradients

Cosine decay phase:

- Smoothly reduces the learning rate following a cosine curve for steady learning throughout training
- This combination provides both faster convergence during early training and finer adjustments during later stages, contributing to overall training stability and improved model performance.

1.5 Test Image Prediction Analysis

- High-motion actions → excellent classification
- Ambiguous handheld object actions → confusion

Example:

True Behavior	Model Limit

holding a phone → could be listening or texting	CNN lacks object semantics

Overall, the results demonstrate that the model is highly effective in learning global spatial posture cues, while limited in understanding fine object interactions. Evaluation results are logically consistent across the confusion matrix, per-class accuracy, and sample test predictions.

1.6 Explainability

Model Explainability with Grad-CAM

To interpret the visual focus of the trained CNN model, we applied Gradient-weighted Class Activation Mapping (Grad-CAM). This technique highlights the regions in an image that contribute most strongly to the model's prediction. Warmer colors (red/yellow) indicate high relevance, while cooler colors (blue) indicate low relevance.

Example: Recognizing *listening_to_music*

2.ResNet-50

Model Overview

This ResNet-50 model was developed as one of the deep learning approaches evaluated by our team for Human Activity Recognition (HAR).
Deep learning CNN architectures have proven highly effective in extracting spatial visual features from RGB images in behavior monitoring and security applications.

We implemented a **transfer-learning-based ResNet-50**, where only the **last residual block (Layer4)** and the **fully connected classifier** were fine-tuned. This approach enables:

- Faster training time
- Lower computational cost
- High recognition accuracy

Model Architecture

Item	Configuration
Base Model	ResNet-50 pretrained on ImageNet
Training Strategy	Fine-tuned Layer4 + Fully Connected layer
Input Size	224×224 RGB
Loss Function	CrossEntropyLoss
Optimizer	Adam
Learning Rate	1e-4
Scheduler	StepLR (step_size=5, gamma=0.1)
Batch Size	32
Epochs	15
Device	Google Colab GPU (CUDA)

The model shows **high and consistent recognition performance across all classes**, indicating successful feature learning and strong generalization.

Classification Report

Strongest performing classes:

- **Cycling**, **Eating**, **Sleeping**, **Running** → clear visual cues

Challenging classes:

- **Sitting**, **Texting**, **Calling** → similar static posture

3. EfficientNet-B1

3.1 Methodology

The Human Action Recognition (HAR) pipeline was designed to process raw image data, train deep learning models, and evaluate their performance in a structured manner. The workflow begins with Exploratory Data Analysis (EDA), where class distribution, sample images, and visual variations are examined to understand the challenges posed by lighting, pose, and background clutter.

Next, the dataset undergoes Preprocessing and Augmentation, including resizing, normalization, random flips, and MixUp strategies to improve generalization and reduce overfitting. After preparing the data, multiple deep neural networks are constructed and fine-tuned using transfer learning, where pretrained

models such as EfficientNet, ResNet-50, InceptionV3, and VGG-16 are adapted for 15-class classification with custom classification heads, label smoothing, and dropout.

Training incorporates optimization techniques like learning rate scheduling, Adam/SGD optimizers, and regularization. Finally, models are evaluated using accuracy, precision, recall, F1-score, and confusion matrices, along with Test-Time Augmentation (TTA) to obtain more robust predictions. This structured approach ensures consistent comparison and reliable model performance assessment.

3.2 Preprocessing and Data Augmentation

The images in the dataset vary in resolution and aspect ratio, so a consistent preprocessing pipeline was applied before training. All images were:

- resized and center–cropped to 240 × 240 pixels,
- converted to RGB tensors,
- normalized using standard ImageNet mean and standard deviation.

To improve generalization and reduce overfitting, I used strong data augmentation during training, including:

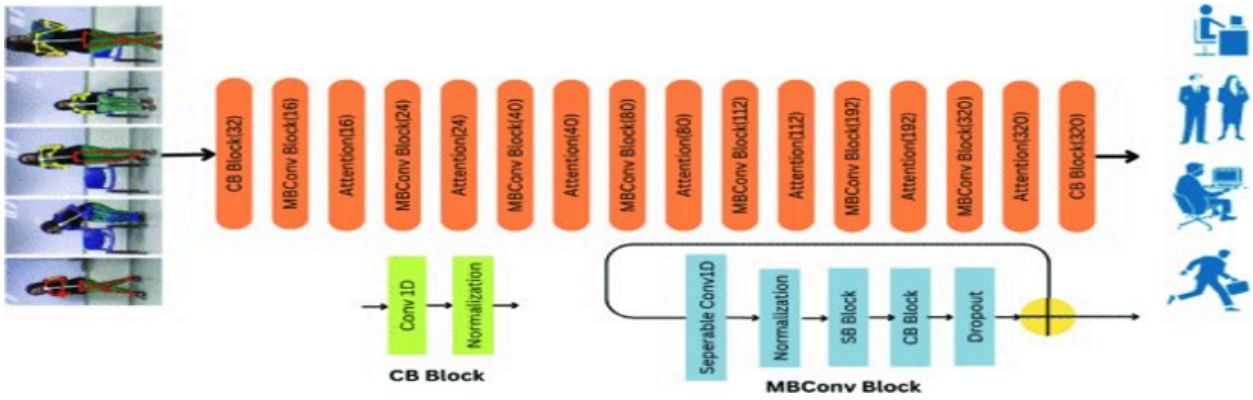
- random resized cropping,
- horizontal flips,
- small rotations and affine translations,
- slight perspective distortions, and
- color jittering (brightness, contrast, saturation, and hue).

In addition to these spatial augmentations, I also applied Mixup regularization with $\alpha = 0.2$, which linearly combines pairs of images and labels. This encourages the model to learn smoother decision boundaries and makes it less sensitive to noise. Overall, this preprocessing and augmentation pipeline helps the network handle variations in pose, lighting, background, and camera viewpoint.

3.3 Model Architecture

➔ EfficientNet-B1 Architecture Used in This Project

EfficientNet-B1 serves as the backbone for this Human Action Recognition system. It follows the core principles of the EfficientNet family—compound scaling and MBConv blocks—but introduces deeper network stages, more channels, and a larger input resolution (240 × 240) compared to EfficientNet-B1. These adjustments make B1 significantly more capable of capturing subtle variations in human posture, orientation, and context, which are essential for action recognition.



Internally, EfficientNet-B1 processes an input image through multiple stages:

- **Stem Convolution Layer**

A 3×3 convolution begins the pipeline by extracting low-level visual features such as edges, textures, and simple geometric structures.

- **Stacked MBConv Blocks (Stages 1–7)**

Each MBConv block is composed of:

- **1×1 expansion convolution**

Expands channel dimensions to increase representation capacity.

- **Depthwise convolution (kernel = 3 or 5)**

Efficiently captures spatial patterns with minimal computational cost.

- **Squeeze-and-Excitation (SE) attention**

Recalibrates channel-wise feature responses by emphasizing the most informative activations.

- **1×1 projection convolution**

Compresses expanded features back into a compact representation.

- **Skip connections**

Facilitate gradient flow and preserve low-level details.The stacked MBConv blocks progressively encode both local motion cues (limb direction, body tilt) and global structural patterns (overall pose), achieving strong representational power with remarkable efficiency.

- **Global Average Pooling (GAP)**

Compresses spatial features into a compact vector.

- **Classification Layer**

Outputs class probabilities across the 15 human action categories.

- **Custom Modifications in This Project**

To adapt EfficientNet-B1 for the specific demands of the HAR dataset, several targeted modifications were introduced. These modifications improve domain adaptation, regularization, and robustness.

- **Custom Classification Head**

This provides:
(i). Dropout for reducing overfitting.
(ii). A new fully connected layer matching the dataset's 15 classes.

- **Label Smoothing**

HAR classes often share visual similarities (e.g., sitting vs. using laptop, calling vs. texting).
To prevent the model from becoming overly confident, label smoothing was applied. This encourages the network to maintain softer, more generalizable decision boundaries.

- **Strong Data Augmentation Pipeline**

To simulate real-world variability, a rich augmentation set was used:

- RandomResizedCrop
- Horizontal Flip
- Rotation
- Affine transformations
- Perspective distortion
- Color Jitter

These augmentations introduce variations in lighting, pose, viewpoint, and background—critical for robust action recognition.

- **Mixup Regularization**

Mixup ($\alpha = 0.2$) blends images and labels
This reduces memorization,stabilizes training and enhances generalization.

- **Gradual Unfreezing for Stable Fine -Tuning**

A two-phase transfer learning strategy was adopted:
- **Stage 1** — Backbone Frozen

Only the classifier trains. This allows EfficientNet's ImageNet features to adapt smoothly to HAR.

● **Stage 2** — Full Network Unfrozen

The entire backbone is fine-tuned with a lower learning rate.

This preserves pretrained knowledge while refining deeper layers for HAR-specific patterns.

● **Test Time Augmentation**

During inference, predictions are averaged across:

The original image

A horizontally flipped version

This produces more stable predictions, especially for actions that appear symmetric (e.g., clapping, hugging).

3.4 Performance Evaluation

The EfficientNet-B1 model achieved strong generalization across all 15 action classes.
To analyze classification behavior, a confusion matrix was generated on the test set.

Key performance metrics:

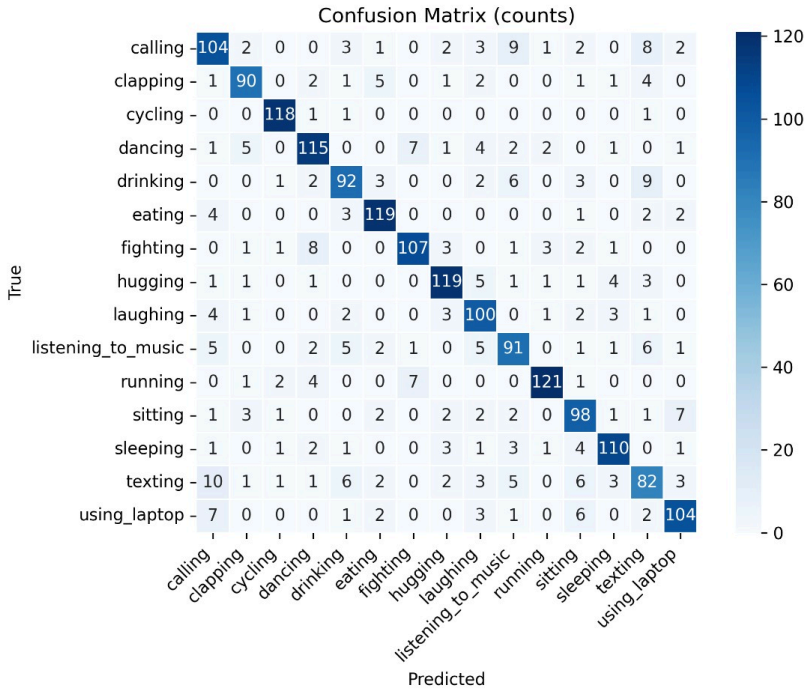
• Accuracy: 83.30%

• Macro Precision: 83.03%

• Macro Recall: 83.02%

• Macro F1-Score: 83.00%

The confusion matrix below highlights that visually distinct actions such as cycling, running, and sleeping achieve the highest correctness. However, seated actions like texting, using laptop, and listening_to_music show higher misclassifications due to pose similarity and subtle contextual differences.



Compared to other tested architectures, EfficientNet-B1 delivered the strongest balance of accuracy and computational efficiency.

3.5 Hyperparameter Tuning

The model was trained using a fixed set of hyperparameters selected through iterative experimentation. Key configurations include:

- Input Size: **240x240**
- Batch Size: **64**
- Optimizer: **Adam**
- Initial Learning Rate: **1e-4**
- Fine-tuning learning rate: **3e-5**
- Epochs: **25**
- Loss-Function: **Cross-entropy**
- Weight-Decay = **1e-4**

A ReduceLROnPlateau scheduler was used to lower the learning rate when validation loss stopped improving. The final model checkpoint was selected based on the highest validation accuracy.

4. Inception v3

InceptionV3 is a deep convolutional neural network designed to capture multi-scale visual information efficiently. Instead of using one fixed convolution size, it processes the image with parallel filters of different sizes (1×1, 3×3, 5×5) and merges them to extract both fine-grained pose details and larger contextual cues (e.g., phone, laptop, bottle). It also reduces computation using factorized convolutions, making it accurate yet lightweight compared to older architectures like VGG-16. This balance of efficiency and feature richness makes InceptionV3 well-suited for recognizing diverse human actions from static images.

4.1 Dataset Preparation and Splits

The original HAR dataset was organized in a Kaggle format, where all labeled image paths were listed in Training_set.csv and the unlabeled test images were listed in Testing_set.csv. These images are stored in two folders, train/ and test/. As a first step, our team extracted the 15 action categories from the training file and assigned each action a numeric label (0–14). This mapping is stored and reused throughout the project so that every model interprets the classes in the same order when training, evaluating results, or saving predictions.

To make the dataset ready for deep learning, we then created consistent and balanced data splits. Using stratified sampling, we divided the labeled data into three subsets: 80% for model training, 10% for validation, and 10% for internal testing. This ensures that each action class is equally represented in every split, preventing biased learning. We saved these splits as train_split.csv, val_split.csv, and test_split.csv, while all image files remain in the single train/ folder.

The unlabeled inference images listed in Testing_set.csv remain in the separate test/ folder, where they are only used for final predictions. Through this structure, every experiment we run is guaranteed to use non-overlapping,

label-balanced subsets, and all models trained by our group share the exact same splits, making performance comparison fair and reliable.

4.2 Inception v3 Architecture Overview

For the core model I chose **Inception v3**, a widely used CNN architecture with factorized convolutions and auxiliary branches designed for efficient, deep feature extraction.

At a high level, the pipeline is:

1.**Input layer**

RGB image resized to **299×299** and normalized with standard ImageNet mean and standard deviation.

2. **Inception v3 backbone**

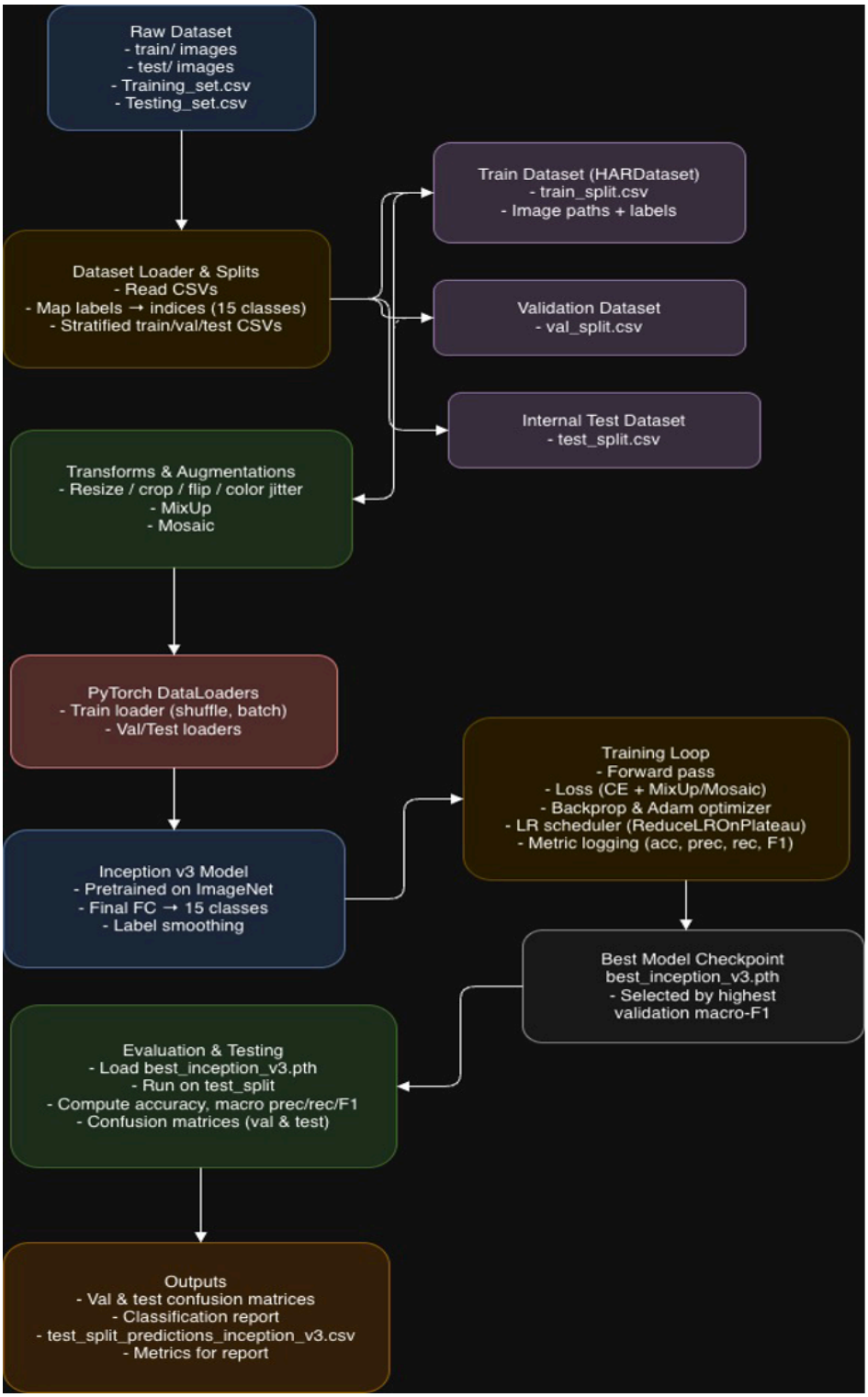
- Multiple convolution + pooling layers followed by several Inception modules that capture multi-scale features through parallel convolutional paths.
- Global average pooling at the end creates a compact feature vector.

3. **Final classification head (modified)**

- The original 1000-class fully connected layer is replaced by a **15-unit linear layer**, one for each action category.
- Softmax is applied at evaluation time to obtain class probabilities.

4. **Loss function with label smoothing**

- During training, cross-entropy with label smoothing is used to prevent the model from becoming overconfident in its predictions.



Inception v3 Architecture Overview

4.3 Data Augmentation Strategy

Real-world human action images differ in lighting, viewing angles, body pose, and background clutter. To help our models generalize to these variations, we applied a set of augmentation techniques during training.

1) Standard Image Transforms
Basic image modifications were used to increase diversity without changing the action label:

- Random resized crops (around 299×299)
- Horizontal flips to mirror poses
- Color jitter for varying brightness/contrast
- Normalization with ImageNet statistics for pretrained compatibility

These transforms make the model less sensitive to lighting and camera position.

2) MixUp Regularization
MixUp blends two images and interpolates their labels using a random mixing coefficient. This creates soft targets that reduce overconfidence and improve performance on ambiguous actions (e.g., sitting vs. using a laptop).

3) Mosaic Augmentation (Optional)
We also experimented with Mosaic, where four images are arranged into a 2×2 grid. This exposes the network to richer spatial composition and reduces reliance on localized features in a single region.

Together, these augmentations improve generalization, reduce overfitting, and make the model more robust to real-world variability.

4.4 Hyperparameter Settings and Tuning

Rather than doing an exhaustive grid search, we focused on tuning a small set of hyperparameters that had the biggest impact on performance: learning rate, training length, regularization strength, and augmentation intensity.

For optimization, we used the Adam optimizer with an initial learning rate of 1×10^{-4}. A ReduceLROnPlateau scheduler monitored the validation macro-F1 score and automatically reduced the learning rate (by a factor of 0.5) whenever performance stopped improving. This is reflected in the training logs, where the learning rate drops partway through training and helps the model refine its weights in later epochs.

We trained the models for 15 epochs with a moderate mini-batch size (chosen based on GPU memory limits), and we saved the best checkpoint based on highest validation macro-F1, not just accuracy. This ensures that the chosen model performs well across all 15 classes, including the harder, visually similar ones.

Regularization-related hyperparameters were also tuned empirically. We used label smoothing in the cross-entropy loss to prevent over-confident predictions, dropout in the classification head to reduce overfitting, and carefully chosen strengths for MixUp and Mosaic augmentation so that blended images remained realistic while still providing enough variability. Together, these hyperparameter choices produced a stable training process and strong generalization on the internal test split.

4.5 Training Configuration

The Inception v3 model was fine-tuned for our 15-class HAR task using the following setup:

- **Base Model:** Inception v3 with pretrained ImageNet weights (transfer learning).

- **Epochs:** Trained for **15 total epochs** to balance convergence and overfitting.
- **Optimizer: Adam**, chosen for stable and smooth weight updates.
- **Initial Learning Rate: 0.0001**, with adaptive reduction when validation performance plateaued.
- **Scheduler: ReduceLROnPlateau** (factor = 0.5, patience = 2), triggered by validation macro-F1.
- **Batch Size: 32**, selected based on training stability and GPU memory efficiency.
- **Loss Functions:**
  Cross-entropy with label smoothing for standard batches.
  Log-softmax with soft labels for MixUp-augmented batches.
- **Hardware:** Runs on **GPU (cuda)** when available, otherwise automatically falls back to CPU.

Checkpoint Saving Rule:
At the end of each epoch, validation metrics are computed. If validation macro-F1 improves, the model parameters are saved as the current best checkpoint (best_inception_v3.pth)

4.6 Evaluation and Post-Processing

After training, the best model was evaluated using a consistent and reproducible testing pipeline:

1. Checkpoint Loading

- The Inception v3 network is re-initialized with the same 15-class classification head used during training.
- The saved weights from best_inception_v3.pth are loaded.
- The model is switched to evaluation mode, disabling gradient updates and dropout for stable inference.

2. Metric Computation

- The evaluation runs on the internal test split (~1,260 labeled images).
- The following performance metrics are calculated:
  - Overall accuracy
  - Macro precision
  - Macro recall
  - Macro F1-score
- Additionally, the script generates:
  - A 15 × 15 confusion matrix to show class-level behavior.
  - A full classification report, including precision, recall, F1-score, and support for each of the 15 action categories.

3. Prediction Export

- The evaluation script saves prediction to test_split_predictions_inception_v3.csv, containing
  - Image filename
  - True label (if available)

    - Predicted label

- This file is useful for:

    - Inspecting misclassified examples
    - Reporting model performance
    - Preparing Kaggle-style submission files if needed.

5. VGG16

5.1 Model Architecture

The classification model used for human action recognition was VGG16, pretrained on ImageNet. The architecture was adapted for the 15-class dataset and for Input Layer All images were resized to 224×224, converted to RGB, and transformed into PyTorch tensors. In Architecture 13 convolutional layers, ReLU activations, 5 max-pooling layers, Fully connected feature layers are used which are standard VGG16 feature extraction stack. The pretrained ImageNet weights allowed faster convergence and improved feature transfer.

In Modified Classification Head original 1000-class final fully connected layer was replaced with:

```
model.classifier[6] = nn.Linear(num_features, len(train_dataset.labels))
```

corresponding to the 15 human action classes.

In Output Computation During inference, predictions were obtained by

```
preds = outputs.argmax(1)
```

Loss Function: The model was trained using Cross-entropy loss (standard version, without label smoothing)

5.2 Data Augmentation

A minimal augmentation and preprocessing pipeline was employed to maintain input consistency.

In Training-time augmentations

- Random Horizontal Flip
- Resize to 224×224
- Tensor conversion

Validation/Test preprocessing:

- Resize to 224×224
- Tensor conversion

5.3 Training Configuration

The VGG16 model was fine-tuned using pretrained ImageNet weights over a total of ten epochs. Training was performed with the Adam optimizer and an initial learning rate of 0.0001. To adapt the learning rate during training, a ReduceLROnPlateau scheduler was employed; it monitored the validation macro-F1 score and reduced the learning rate by a factor of 0.5 when no improvement was observed for two consecutive epochs. All experiments were run on a GPU with a batch size of 32.

Throughout the training process, several metrics were recorded at the end of every epoch. These included the training accuracy, training macro-F1 score, validation accuracy, validation macro-F1 score, and the current learning rate. The collected metrics were saved to *epoch_metrics.csv* to support performance tracking and later analysis.

5.4 Model Checkpointing

To ensure that the best-performing version of the model was saved, checkpointing was enabled based on validation macro-F1. Whenever the validation F1 score improved, the current weights were saved as *best_vgg16.pth*. This approach prevents overfitting within a single run and guarantees that the most reliable model is used during evaluation.

5.4 Testing Procedure

After training was completed, the best checkpoint was loaded for testing on an unseen, independently held-out test split. The same preprocessing pipeline used during training—resizing all images to 224×224 followed by conversion into tensors—was applied to the test data to maintain consistency.

During inference, model outputs were converted to class predictions using argmax over the logits. Two evaluation outputs were generated from this final test run:

1. **Classification Report**
   A detailed per-class summary including precision, recall, F1-score, and support.
   This was exported as *test_class_report.csv*.

2. **Confusion Matrix**
   A 15×15 confusion matrix summarizing how the model distributed predictions across classes.
   This was saved as *test_confusion_matrix.csv*.

7. Results

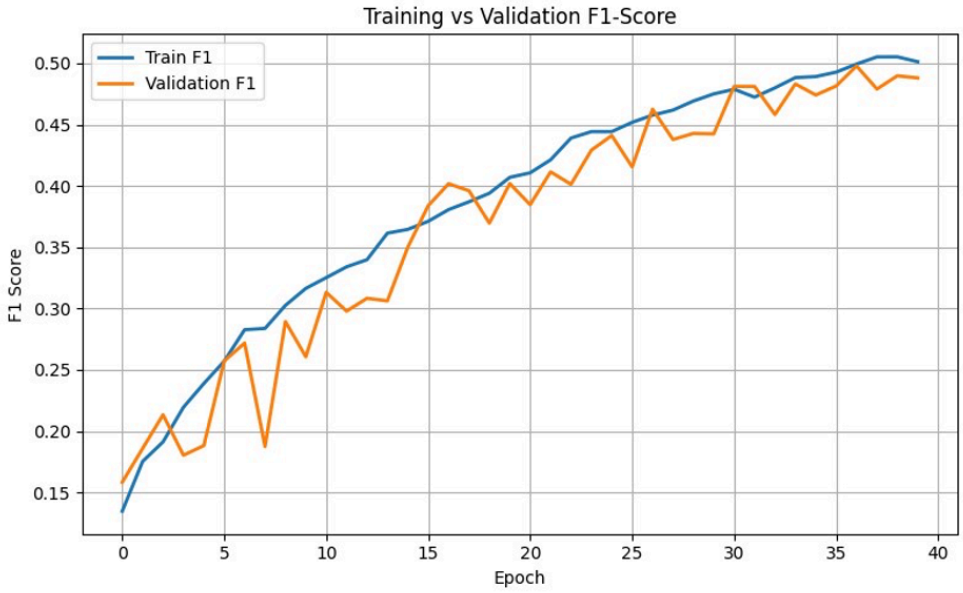
The results of our experiments made it very clear that transfer-learning models have a strong advantage over networks built from scratch when recognizing human actions from images. Architectures like EfficientNet-B1, Inception v3, and ResNet-50 were better at picking up subtle visual details such as body pose, hand movement, or context clues in the background which helped them deal with the wide variety of scenes and lighting conditions

present in the dataset. VGG-16 performed reasonably well but struggled with its large number of parameters and limited regularization, which made it less flexible compared to newer models. On the opposite end, our custom CNN found it difficult to generalize, showing that a simple handcrafted model is not enough for a complex, multi-class task unless it has a much larger capacity or a much bigger dataset. From these observations, it became evident that modern pretrained architectures are more dependable choices for building a robust human action recognition system.

Model	Accuracy	Macro F1-Score	Weighted F1-Score
Inception v3	0.8262	0.8266	0.8290
EfficientNet-B1	0.8300	0.8300	0.8300
ResNet-50	0.8196	0.8200	0.8200
VGG-16	0.75	0.76 (Val)	0.74 (Test)
Custom CNN	0.55	0.54	0.54

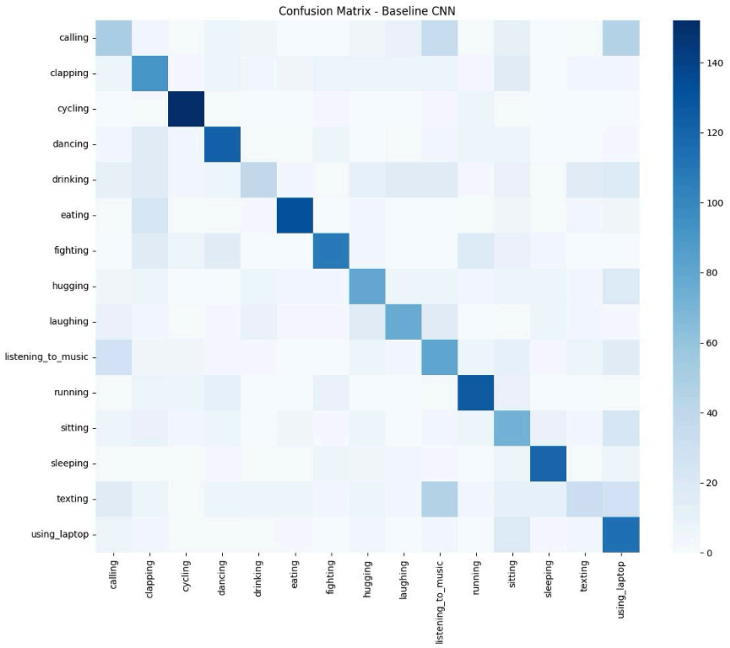
Baseline CNN

Train vs Validation Macro-F1 across 40 epochs.



The upward trend in both curves demonstrates effective learning and stability without divergence, confirming successful augmentation and optimization strategies.

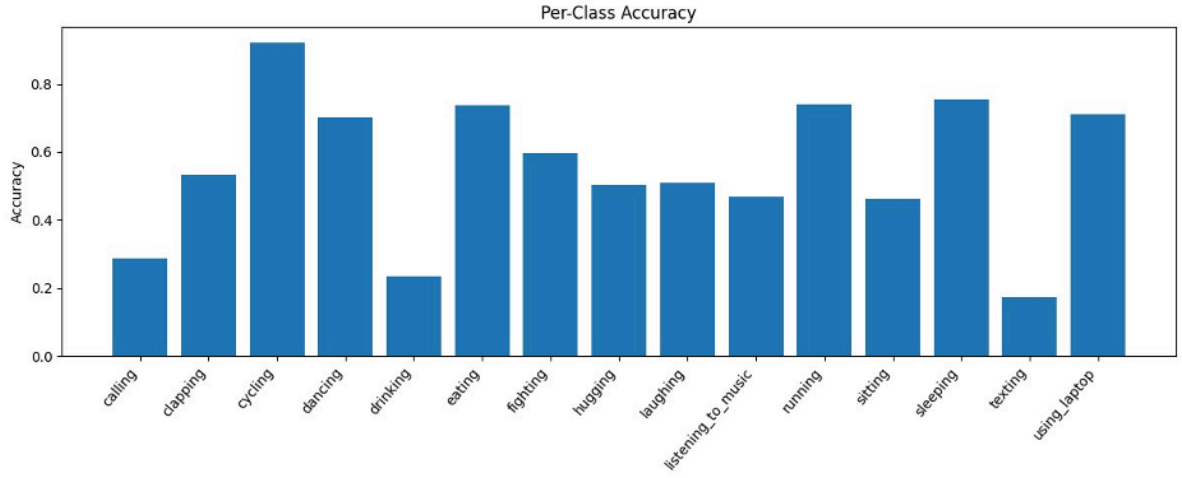
Confusion Matrix



The confusion matrix helps us see where the model struggles, not just how often it is correct. Most errors occur between actions that look visually alike. While classes like cycling, dancing, and sleeping are predicted reliably, actions such as laughing vs. hugging, sitting vs. using a laptop, and drinking vs. eating are frequently mixed up.

These mistakes mainly happen because static images only capture posture, not context or motion. For example, holding something near the face can look similar whether a person is drinking, texting, or listening to music. This shows that action recognition from still images is limited, and models could benefit from object awareness or temporal motion cues.

Per Class Accuracy



The model works very well for actions with clear body posture, like cycling, sleeping, or dancing. However, it struggles with subtle, object-dependent actions such as texting, drinking, or listening to music, where the visual differences are small and harder to detect from posture alone.

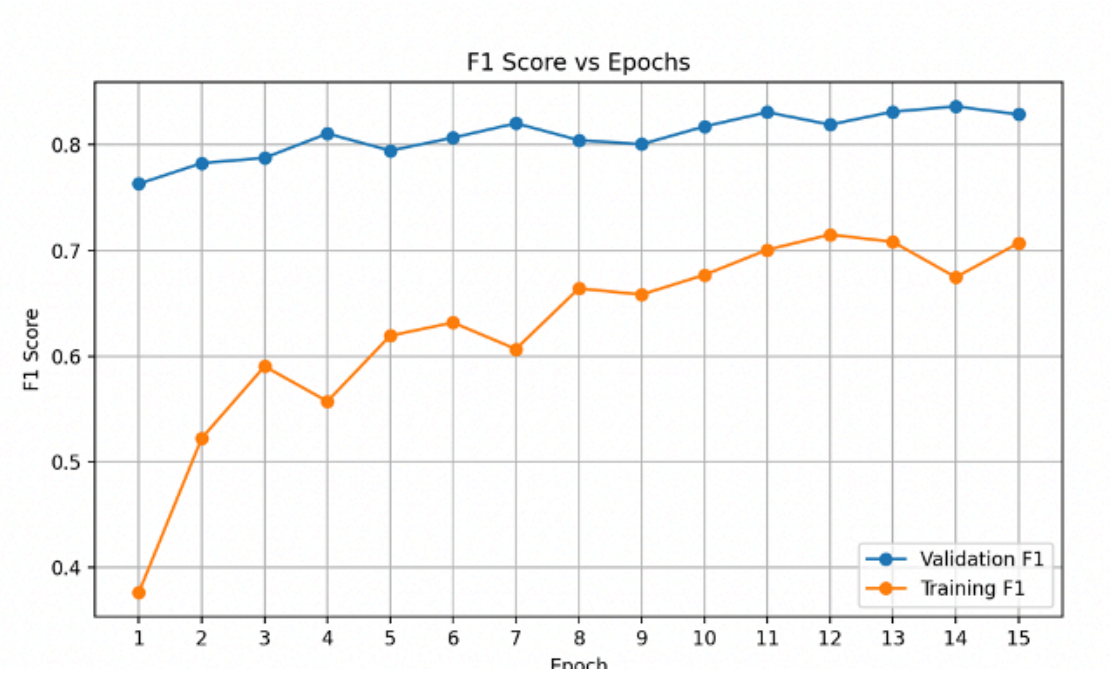
Classification Report

Class	Precision	Recall	F1-Score	Support
calling	0.34	0.29	0.31	175
clapping	0.44	0.53	0.48	173
cycling	0.81	0.92	0.86	165
dancing	0.65	0.70	0.68	174
drinking	0.51	0.23	0.32	166
eating	0.81	0.74	0.77	179
fighting	0.71	0.60	0.65	181
hugging	0.51	0.50	0.51	157
laughing	0.61	0.51	0.56	149
listening_to_music	0.36	0.47	0.41	171
running	0.69	0.74	0.71	170
sitting	0.37	0.46	0.41	156
sleeping	0.74	0.75	0.75	159
texting	0.42	0.17	0.24	186
using_laptop	0.41	0.71	0.52	159

The classification report shows clear differences between action types. Actions with unique body posture, like cycling, dancing, and sleeping, are predicted very accurately. In contrast, actions that depend on small objects or context—such as texting or drinking—are harder to recognize because key visual details are less noticeable in a single image.

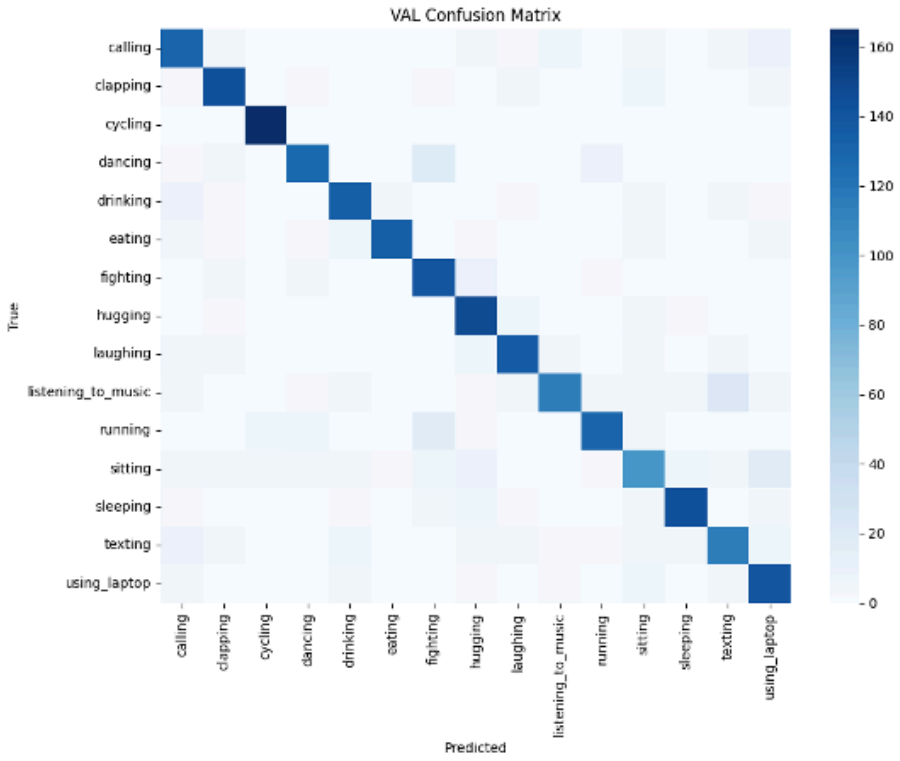
Inception v3

Training vs Validation Macro-F1 Curve of Inception v3



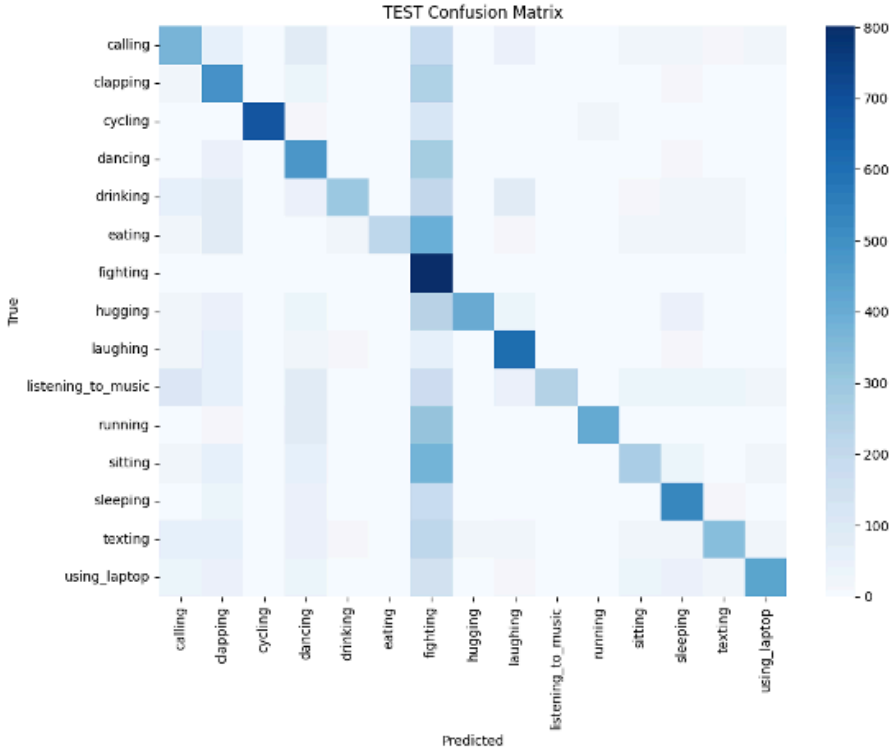
The F1 score curve shows steady learning across training epochs. Validation performance remains consistently high from the beginning, while training F1 gradually catches up as the model learns useful features. By the final epochs, both curves stabilize, suggesting that the model has learned most discriminative patterns without severe overfitting. This indicates a well-balanced training process where performance improves without sacrificing generalization.

Validation Confusion Matrix Insights



The validation confusion matrix shows that the model performs very well for actions with distinct visual cues, such as cycling and sleeping, which are rarely misclassified. However, actions that involve similar sitting postures or handheld objects—like calling, texting, and using a laptop—show more overlap. This suggests that while the model understands clear body pose differences, it struggles when small objects or subtle gestures are needed to tell actions apart.

Test Confusion Matrix Interpretation



The test confusion matrix shows that the model correctly identifies most actions, with strong diagonal values across many classes. The few mistakes mainly occur between visually similar pairs, such as calling vs. texting or sitting vs. using a laptop, where posture and hand placement overlap. This confirms that the model generalizes well, and most remaining errors come from naturally ambiguous actions rather than poor learning.

To summarize the detailed classification report:

Class	Precision	Recall	F1-score	Support
calling	0.83	0.80	0.81	84
clapping	0.75	0.83	0.79	84
cycling	0.97	0.99	0.98	84
dancing	0.77	0.81	0.79	84
drinking	0.91	0.83	0.87	84

eating	0.97	0.85	0.90	84
fighting	0.88	0.90	0.89	84
hugging	0.85	0.82	0.84	84
laughing	0.84	0.82	0.83	84
listening_to_music	0.73	0.79	0.76	84
running	0.85	0.87	0.86	84
sitting	0.60	0.60	0.60	84
sleeping	0.90	0.90	0.90	84
texting	0.80	0.73	0.76	84
using_laptop	0.77	0.86	0.81	84

Key observations :
Actions like cycling and sleeping were classified almost perfectly due to their distinctive posture and context. Eating, fighting, running, and using a laptop also performed well, with only occasional mix-ups. The most difficult class was sitting, which was often confused with using laptop and texting because all three look visually similar in a seated pose. Expression-based actions such as clapping, laughing, and listening to music were moderately accurate, but errors occurred when hands or faces were unclear or partially hidden.

EfficientNet-B1

Classification Performance

	TXT	precision	recall	f1-score	support
1	calling	0.7482014388489209	0.7591240875912408	0.7536231884057971	137.0
2	clapping	0.8571428571428571	0.8333333333333334	0.8450704225352113	108.0
3	cycling	0.944	0.9752066115702479	0.959349593495935	121.0
4	dancing	0.8333333333333334	0.8273381294964028	0.8303249097472925	139.0
5	drinking	0.8	0.7796610169491526	0.7896995708154506	118.0
6	eating	0.875	0.9083969465648855	0.8913857677902621	131.0
7	fighting	0.8770491803278688	0.84251968503937	0.8594377510040161	127.0
8	hugging	0.875	0.8686131386861314	0.8717948717948718	137.0
9	laughing	0.7692307692307693	0.8547008547008547	0.8097165991902834	117.0
10	listening_to_music	0.7520661157024794	0.7583333333333333	0.7551867219917012	120.0
11	running	0.9307692307692308	0.8897058823529411	0.9097744360902256	136.0
12	sitting	0.765625	0.8166666666666667	0.7903225806451613	120.0
13	sleeping	0.88	0.859375	0.8695652173913043	128.0
14	texting	0.6890756302521008	0.656	0.6721311475409836	125.0
15	using_laptop	0.859504132231405	0.8253968253968254	0.8421052631578947	126.0
16	accuracy	0.8306878306878307	0.8306878306878307	0.8306878306878307	0.8306878306878307
17	macro avg	0.8303998458559311	0.8302914341120924	0.8299658694397595	1890.0
18	weighted avg	0.8313212320848073	0.8306878306878307	0.8306372003280401	1890.0

Actions like cycling, running, sleeping, and fighting were recognized most confidently, as their body posture and movement cues are easy for the model to distinguish. On the other hand, actions that look visually alike such as calling vs. texting or sitting vs. using a laptop were more challenging, leading to occasional confusion. Still, with an overall performance around 83% accuracy, the model shows it has learned meaningful visual cues and can reliably identify a wide range of everyday human activities.

Confusion Matrix

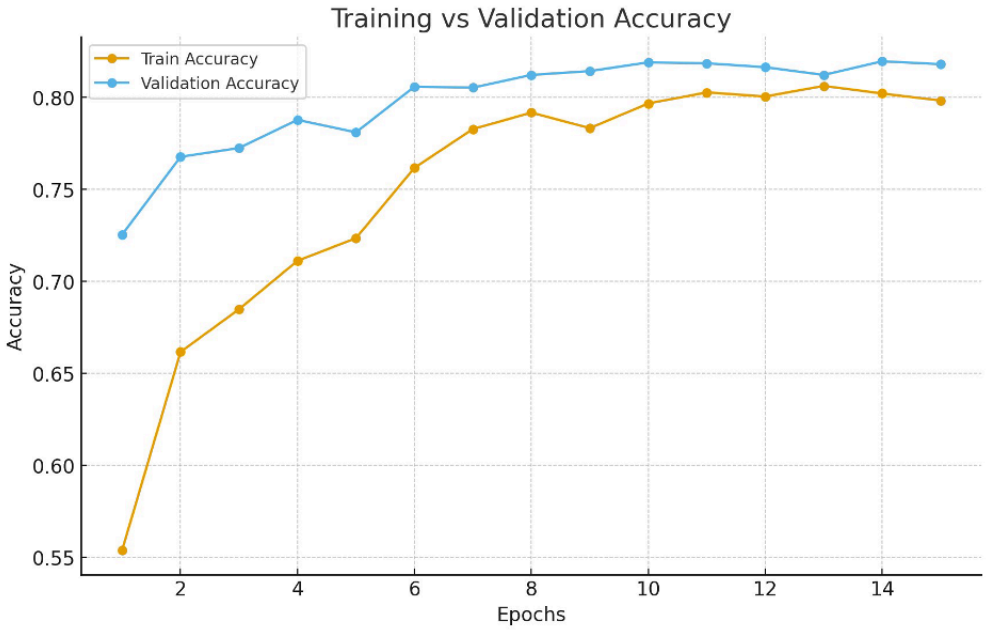
CONFUSION MATRIX

	calling	clapping	cycling	dancing	drinking	eating	fighting	hugging
calling	107	2	0	0	2	1	0	1
clapping	0	93	0	2	1	0	2	1
cycling	0	0	119	1	1	0	0	0
dancing	0	4	0	123	0	0	6	1
drinking	1	0	2	0	94	3	1	0
eating	2	0	0	0	7	117	0	0
fighting	1	0	0	8	1	0	110	1
hugging	0	1	1	2	0	0	0	122
laughing	1	2	0	2	2	1	0	3
listening_to_mu	3	2	1	3	2	2	1	0
running	0	1	2	5	0	0	11	0
sitting	2	3	1	4	2	2	0	3
sleeping	0	0	0	2	2	0	1	2
texting	11	1	1	1	2	2	1	2
using_laptop	9	1	1	0	0	1	0	0

The confusion matrix shows a strong diagonal, meaning the model correctly identifies most actions. The few mix-ups mainly occur between visually similar activities like calling vs. texting or sitting vs. using a laptop, where body posture looks almost the same. In contrast, actions with clear full-body movements are rarely confused. Overall, the matrix highlights that most errors come from natural visual overlap rather than model weakness.

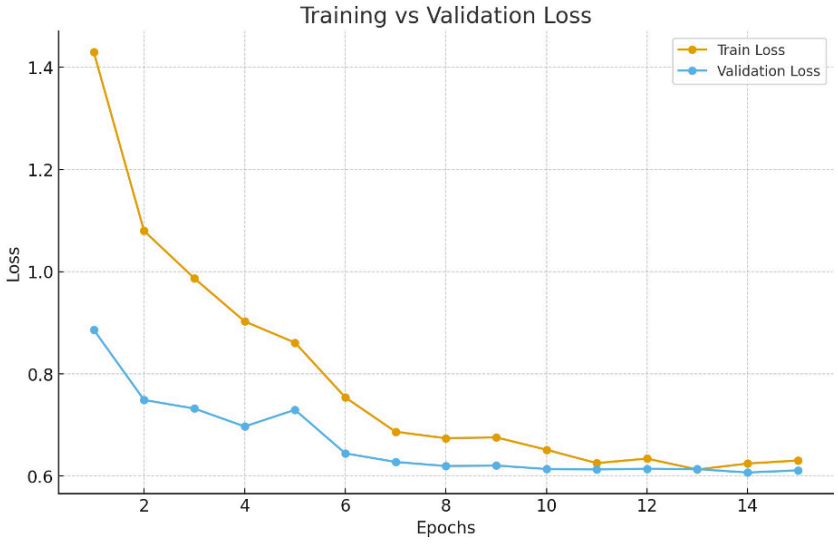
ResNet-50

Accuracy Curve



Both training and validation accuracy steadily improve and level off towards the end, suggesting the model learns consistently without overfitting. The curve plateaus around epochs 12–15, meaning most of the useful features have already been captured and extending training wouldn't provide much additional benefit.

Loss Curve



Both training and validation accuracy steadily improve and level off towards the end, suggesting the model learns consistently without overfitting. The curve plateaus around epochs 12–15, meaning most of the useful features have already been captured and extending training wouldn't provide much additional benefit.

Classification Report

```
Classification Report:
                    precision    recall  f1-score   support

           calling       0.75      0.75      0.75       126
          clapping       0.84      0.78      0.81       126
           cycling       0.98      0.97      0.98       126
           dancing       0.81      0.79      0.80       126
          drinking       0.85      0.87      0.86       126
            eating       0.89      0.92      0.90       126
          fighting       0.79      0.87      0.83       126
           hugging       0.83      0.83      0.83       126
          laughing       0.84      0.82      0.83       126
listening_to_music       0.79      0.83      0.81       126
           running       0.86      0.85      0.86       126
           sitting       0.67      0.64      0.66       126
          sleeping       0.87      0.87      0.87       126
           texting       0.75      0.65      0.69       126
       using_laptop       0.76      0.82      0.79       126

          accuracy                           0.82      1890
         macro avg       0.82      0.82      0.82      1890
      weighted avg       0.82      0.82      0.82      1890
```

Classes like cycling, eating, sleeping, and running achieve strong F1-scores because they contain clear visual cues that stand out, such as a bicycle or a lying posture. On the other hand, actions like sitting, texting, and calling are harder to separate since they look visually similar and involve minimal movement, which naturally leads to more confusion.

Confusion Matrix

```
Confusion Matrix:
[[ 95   1   0   1   2   1   2   0   2   8   1   3   0   6   4]
 [  0  98   0   4   1   3   2   2   4   0   1   5   1   4   1]
 [  0   0 122   0   0   0   1   0   0   0   1   2   0   0   0]
 [  2   4   0 100   0   0  12   0   0   1   5   1   1   0   0]
 [  2   1   0   3 110   2   0   0   2   1   1   0   0   2   2]
 [  2   2   0   0   2 116   0   0   1   0   0   1   1   0   1]
 [  0   1   1   5   0   0 110   1   0   0   2   3   2   0   1]
 [  1   1   0   1   0   0   3 105   2   1   2   2   4   3   1]
 [  1   3   0   1   0   1   0   9 103   3   1   2   1   0   1]
 [  2   0   0   1   3   1   0   1   1 104   1   4   1   4   3]
 [  0   1   1   7   1   0   5   1   1   0 107   2   0   0   0]
 [  6   3   0   1   4   7   4   3   0   2   1  81   2   4   8]
 [  2   0   0   0   2   0   1   2   1   1   0   3 110   1   3]
 [ 10   1   0   0   3   0   0   3   5   8   1   3   2  82   8]
 [  4   1   0   0   1   0   0   0   0   3   0   9   1   4 103]]
```

Most predictions fall cleanly along the diagonal, showing that the model gets most actions right. A few confusions happen between actions that look alike—for example, texting vs. sitting, calling vs. texting, or hugging vs. fighting,

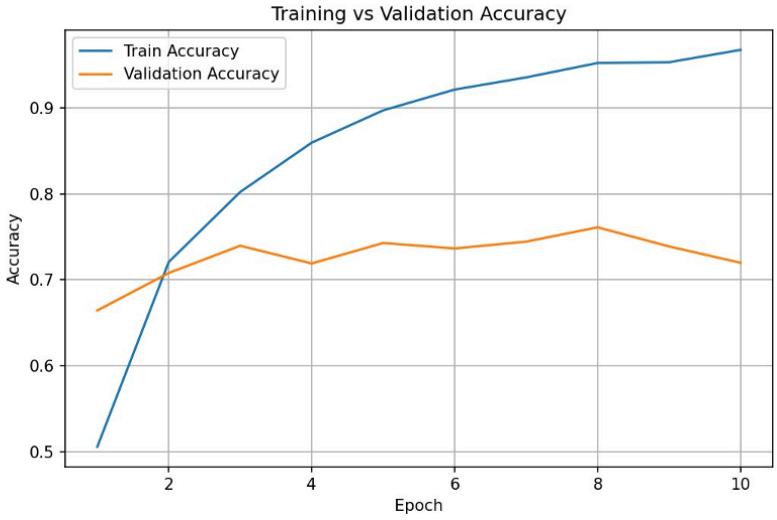
where similar hand or body positions make them harder to separate. This suggests that richer context or motion cues could improve recognition further.

Results

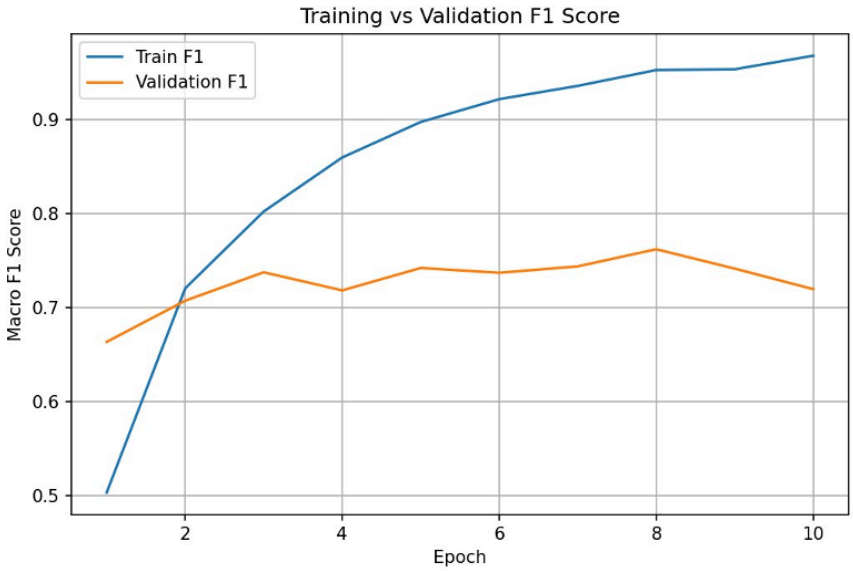
Metric	Score
Validation Accuracy	0.8196 (~82%)
Macro F1-Score	0.82
Weighted F1-Score	0.82
Number of Classes	15

VGG-16

Accuracy and F1 Curves



The training F1 steadily rises toward 0.96, while the validation F1 levels off much earlier around 0.72–0.76. This gap suggests the model starts memorizing training examples rather than learning patterns that generalize well to new images.



Both curves improve initially, but the validation line flattens after a few epochs while the training curve keeps rising. This growing gap clearly points to overfitting, which is common when large models like VGG16 are fine-tuned on a mid-sized dataset.

Confusion Matrix

<null>	calling	clapping	cycling	dancing	drinking	eating	fighting	hugging	laughing	listening_to_music	running	sitting	sleeping	texting	using_laptop
calling	51	2	0	1	1	0	0	1	1	12	0	3	0	9	3
clapping	3	68	0	1	1	1	0	0	2	1	0	4	0	2	1
cycling	0	0	79	1	0	0	0	0	0	2	0	2	0	0	0
dancing	0	9	1	54	0	1	6	3	1	3	1	1	2	1	1
drinking	9	5	0	0	47	1	1	1	2	2	2	3	2	9	0
eating	2	3	0	0	3	65	0	1	1	1	0	3	0	4	1
fighting	1	4	0	6	0	0	56	6	0	1	3	2	5	0	0
hugging	0	0	0	0	0	0	1	71	3	1	0	3	1	3	1
laughing	1	6	0	1	1	0	1	3	65	2	0	1	2	1	0
listening_to_music	5	1	0	3	0	0	0	0	0	61	1	5	1	7	0
running	0	4	0	4	0	0	4	0	0	1	60	6	1	3	1
sitting	1	12	0	3	1	1	0	4	1	3	0	49	0	3	6
sleeping	2	0	0	0	0	0	0	9	1	0	0	2	68	0	2
texting	3	0	0	0	1	0	0	4	2	3	1	1	1	66	2
using_laptop	7	0	0	1	0	0	0	2	0	1	0	1	3	2	67

The confusion matrix shows the model performs very well on actions with clear visual cues, like cycling and sleeping, which appear almost perfectly classified. Most errors happen between actions that look alike in posture or hand placement, such as calling, drinking, or laughing, where small differences in gestures are harder to distinguish. Overall, mistakes reflect visual similarity rather than model weakness.

Test Set Evaluation

<null>	precision	recall	f1-score	support
calling	0.6	0.6071428571428571	0.6035502958579881	84.0
clapping	0.5964912280701754	0.8095238095238095	0.6868686868686869	84.0
cycling	0.9875	0.9404761904761905	0.9634146341463414	84.0
dancing	0.72	0.6428571428571429	0.6792452830188679	84.0
drinking	0.8545454545454545	0.5595238095238095	0.6762589928057554	84.0
eating	0.9420289855072463	0.7738095238095238	0.8496732026143791	84.0
fighting	0.8115942028985508	0.6666666666666666	0.7320261437908496	84.0
hugging	0.6761904761904762	0.8452380952380952	0.7513227513227513	84.0
laughing	0.8227848101265823	0.7738095238095238	0.7975460122699386	84.0
listening_to_music	0.648936170212766	0.7261904761904762	0.6853932584269663	84.0
running	0.8823529411764706	0.7142857142857143	0.7894736842105263	84.0
sitting	0.5697674418604651	0.5833333333333334	0.5764705882352941	84.0
sleeping	0.7906976744186046	0.8095238095238095	0.8	84.0
texting	0.6	0.7857142857142857	0.6804123711340206	84.0
using_laptop	0.788235294117647	0.7976190476190477	0.7928994082840237	84.0
accuracy	0.7357142857142858	0.7357142857142858	0.7357142857142858	0.7357142857142858
macro avg	0.7527416452749626	0.7357142857142858	0.7376370208657593	1260.0
weighted avg	0.7527416452749626	0.7357142857142858	0.7376370208657593	1260.0

Actions with clear body posture, like cycling, sleeping, or using a laptop, were classified accurately. Most mistakes came from look-alike actions—for example, calling vs. texting, drinking vs. eating, laughing vs. clapping, or even dancing vs. running, where small gesture differences make them easy to confuse.

8. CONCLUSION

- In this project, we developed and systematically evaluated a deep learning–based Human Action Recognition (HAR) system using static RGB images across 15 everyday activities. Our goal was not only to build a working classifier, but also to understand how different convolutional neural network architectures behave when faced with visually similar, pose-based actions such as sitting, texting, and using a laptop. By combining a carefully preprocessed and balanced dataset with transfer learning, strong data augmentation, and consistent evaluation metrics, we were able to draw clear comparisons between modern pretrained models and a custom CNN baseline.
- The experimental results show that **EfficientNet-B1** achieved the best overall performance, with an accuracy and macro F1-score of approximately 0.83. This confirms that compound-scaled architectures can offer a very strong trade-off between accuracy and computational efficiency. **Inception v3** and **ResNet-50** also performed reliably, both exceeding ~0.82 macro F1 in some settings, demonstrating that deep pretrained backbones with multi-scale feature extraction (Inception) or residual connections (ResNet) are well-suited for static HAR tasks. In contrast, **VGG-16** and the **Custom CNN** lagged behind, with VGG-16 suffering from overfitting despite pretrained weights, and the custom model struggling to match the representational power of deeper architectures. These findings reinforce that, for a complex multi-class visual task like HAR, depth, architectural design, and transfer learning are crucial.
- Across all models, the confusion matrices and classification reports highlighted a common pattern: actions with **distinctive full-body posture or strong contextual cues** (e.g., cycling, sleeping, dancing, running) were classified with high confidence, whereas **subtler, object-dependent actions** (e.g., sitting vs. using laptop vs. texting, or calling vs. texting) were much more challenging. Since our system operates on single frames without explicit motion or object detection, the models are forced to infer actions purely from static pose, hand position, and limited background context. This naturally limits performance on ambiguous classes and points to an inherent constraint of still-image HAR.
- Overall, the project demonstrates that **transfer-learning–based CNNs can deliver accurate and robust performance for static-image human action recognition**, provided that the dataset is balanced, augmentations are well-designed, and evaluation is done carefully at the per-class level. At the same time, our results clearly suggest that there is a ceiling to what can be achieved from single RGB frames alone.

Richer representations—such as temporal information from video, explicit object cues (e.g., phone, laptop, bottle), or attention mechanisms focusing on hands and faces—are likely required to resolve the remaining ambiguities. In this sense, the work presented here serves both as a strong baseline HAR system and as a stepping stone toward more advanced, motion-aware and context-aware models, as outlined in our Future Work section.

9. FUTURE WORK

While the models developed in this project demonstrate strong performance for static-image HAR, several promising directions can enhance accuracy, scalability, and real-world deployment:

Incorporating Temporal Motion Information

Many confusions arise from actions that look similar in a single frame. Extending the pipeline to video-based HAR would provide motion cues that are critical for recognition. Potential approaches include:

- **3D CNNs** (e.g., C3D, I3D)
- **ConvLSTM or GRU** hybrid architectures
- **Video Transformers** (e.g., TimeSformer, ViViT)

These models capture how body pose changes over time, improving distinction between similar actions such as sitting, texting, and using a laptop.

Object-Aware and Context-Aware Modeling

Some actions are defined more by **objects** than by posture:

- Calling → phone near ear
- Drinking → bottle or cup
- Using laptop → screen in front

Adding:

- **Object detectors** (YOLO, Faster R-CNN)
- **Hand/face attention modules**
- **Scene context embeddings**

would reduce ambiguity when body posture alone is insufficient.

Multi-Modal Learning

Expanding beyond RGB imagery:

- **Depth maps** to understand pose geometry
- **Optical flow** for motion direction
- **Pose estimation skeletons** for joint pattern analysis

Combining modalities can provide complementary information that single images lack.

Larger and More Diverse Training Data

Collecting or augmenting with:

- More varied lighting & viewpoints
- More complex backgrounds
- Cross-dataset training
  would reduce dataset bias and improve real-world generalization.

Deployment and Optimization

To support real-time applications like surveillance or healthcare monitoring:

- **Model compression** (quantization, pruning)
- **Edge-devices deployment** (Jetson Nano, mobile inference)
- **Batchless streaming prediction**

would improve speed and efficiency while maintaining accuracy.

Summary of Future Work Goal:

Enable a HAR system that is **motion-aware, object-aware, and deployment-ready**, capable of performing reliably in unconstrained environments.

10. REFERENCES

- He, K., Zhang, X., Ren, S., & Sun, J. *Deep Residual Learning for Image Recognition (ResNet)*, CVPR 2016.
- Simonyan, K., & Zisserman, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition (VGG)*, 2014.
- Szegedy, C. et al. *Rethinking the Inception Architecture for Computer Vision (Inception-v3)*, CVPR 2016.
- Tan, M., & Le, Q. V. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*, ICML 2019.
- Meet Nagadia. *Human Action Recognition Dataset*, Kaggle.
- PyTorch Documentation – torchvision.models, torch.utils.data, etc.