**Department of Computer Engineering**

**Vivekanand Education Society’s Institute of Technology**

**2024-25**

****

**Case Study**

as part of

ADS- Continuous Assessment

On

**DATA ANALYSIS ON WEDAFALL VIDEO DATASET**

By

**Amit Murkalmath**

**D17C Roll No. 47**

Under the Guidance of

**Mrs. Abha Tewari**

**INDEX**

| **Sr No.** | **Title** | **Page No.** |
| --- | --- | --- |
| **1.** | **Introduction** |  |
| 1.1 | Problem Definition & Scope of Project |  |
| 1.2 | Tools and Technologies Used |  |
| **2.** | **Dataset** |  |
| 2.1 | Type of Dataset |  |
| **3.** | **Implementation** |  |
| 3.1 | Preprocessing |  |
| 3.2 | Model and Evaluation |  |
| 3.3 | Visualization |  |
| **4.** | **Conclusion** |  |
| **5.** | **References** |  |

**1 Introduction**

1.1 Problem Definition & Scope of Project

In recent years, fall detection has become a crucial aspect of ensuring the safety and well-being of elderly individuals and patients with mobility impairments. Falls can result in serious injuries, prolonged hospital stays, and even death if not addressed promptly. Traditional fall detection systems, such as wearable devices or manual supervision, often suffer from limited accuracy, user compliance issues, or high dependency on external infrastructure. With the advancement of computer vision and deep learning, video-based fall detection offers a promising, non-intrusive alternative that can accurately identify falls in real-time without physical contact.

This project focuses on analyzing the **WEDA Fall Dataset**, which contains categorized video samples of daily activities (ADLs) and fall events. The goal is to develop a robust, deep learning-based model capable of distinguishing between normal activities and falls using convolutional neural networks such as MobileNetV2 and ResNet-50. The project involves preprocessing video data, applying augmentations, selecting and fine-tuning models, evaluating performance, and visualizing the results to gain actionable insights. The final output will be a trained model capable of detecting falls with high accuracy, which can be integrated into real-time surveillance systems in homes, hospitals, and elderly care centers to ensure rapid response and improved safety.

1.2 Tools and Technologies Used

**Technical Specifications**

| **Category** | **Tool/Technology** | **Purpose** |
| --- | --- | --- |
| Programming Language | Python | Core development language |
| Deep Learning Models | MobileNetV2, ResNet-50 | Feature extraction and classification |
| Video Processing | OpenCV | Frame extraction and video manipulation |
| Data Handling | NumPy, Pandas | Data management and manipulation |
| Model Training | PyTorch | Deep learning model training and evaluation |
| Visualization | Matplotlib, Seaborn | Data and performance visualization |
| Evaluation Metrics | scikit-learn | Confusion matrix, classification report |
| Hardware | GPU-enabled Runtime | Accelerated model training and inference |

**Python Libraries**

* **torch / torchvision** – For building, training, and fine-tuning deep learning models.
* **cv2 (OpenCV)** – To read video files, extract frames, and perform image processing.
* **numpy** – To perform numerical operations and frame array manipulations.
* **matplotlib & seaborn** – For plotting training curves and confusion matrices
* **tqdm** – For real-time progress bars during training/validation loops.
* **sklearn.metrics** – For generating evaluation metrics like confusion matrix and F1-score.
* **os / glob** – For organizing dataset directories and handling files.

**2 Dataset**

2.1 Dataset Specifications

The dataset used in this project is the **WEDA Fall Dataset**, which is specifically curated for human activity recognition with an emphasis on fall detection. It comprises a collection of short video clips categorized into two primary classes: **ADLs (Activities of Daily Living)** and **Falls**. These videos are recorded in controlled environments simulating real-world scenarios involving different subjects, actions, and camera angles to provide a diverse and challenging dataset.

Each video captures sequences of motion that reflect either normal activities (such as walking, sitting, bending) or fall events (forward, backward, or side falls). The dataset is well-suited for training supervised learning models to classify between safe and dangerous human actions using visual cues.

**Key Features:**

* Two classes: **ADLs** and **Falls**.
* Video format: .MP4, with varying resolutions and frame rates.
* Source: WEDA (Wearable and Environmental Data for Aging) Fall Dataset.
* Realistic simulation of falls and daily activities by different actors.
* Ideal for tasks involving video classification and human pose estimation.
* Suitable for applications in surveillance, elderly care, and smart homes.
* Dataset structure organized into two folders: ADLs/ and Falls/, making it compatible with standard PyTorch datasets.

**3. Implementation**

3.1 Preprocessing

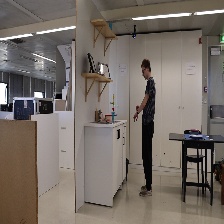
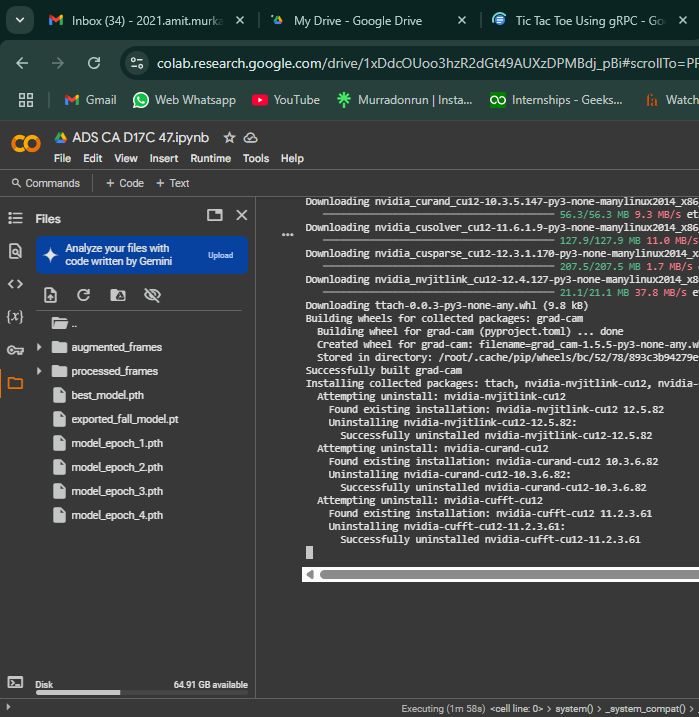
The preprocessing stage is critical to prepare raw video data for deep learning. Since neural networks operate on fixed-size image tensors, we convert each video into individual frames and resize them. We also normalize pixel values and ensure consistent channel ordering.

**Process Flow:**

1. **Directory Structure Setup**:
   * Two main folders: ADLs/ and Falls/, each containing multiple .mp4 video files.
2. **Video Frame Extraction**:
   * Each video is read using **OpenCV** and frames are sampled at a fixed interval (e.g., 1 frame per second) to reduce redundancy.
   * Extracted frames are resized to **224×224 pixels** for compatibility with pre-trained models like MobileNetV2 and ResNet-50.
3. **Frame Labeling**:
   * Each frame inherits its label from the folder it came from (i.e., 0 for ADLs, 1 for Falls).
4. **Data Storage**:
   * Frames are saved in a structured format: dataset/ADLs/ and dataset/Falls/ for ease of loading with ImageFolder.

**Code Excerpt:**

frame = cv2.resize(frame, (224, 224))

cv2.imwrite(os.path.join(save\_path, f"{video\_name}\_frame\_{frame\_count}.jpg"), frame)

3.2 Model and Evaluation

The objective of the model was to classify video frames into two categories: **Falls** and **ADLs** (Activities of Daily Living). For this task, we employed **MobileNetV2**, a lightweight convolutional neural network optimized for real-time classification, due to its balance between speed and accuracy. We also tested **ResNet-50**, a deeper network with strong performance on complex visual data. Ultimately, MobileNetV2 offered faster training and was well-suited for embedded or real-time applications, making it the preferred choice.

**Workflow:**

1. **Model Selection**:
   * Pre-trained **MobileNetV2** was loaded with torchvision.models.mobilenet\_v2(pretrained=True).
   * The final classification layer was modified to output 2 classes.

def get\_model(model\_name='mobilenetv2'):

if model\_name == 'mobilenetv2':

model = models.mobilenet\_v2(pretrained=True)

model.classifier[1] = nn.Linear(model.last\_channel, 2)

elif model\_name == 'resnet50':

model = models.resnet50(pretrained=True)

model.fc = nn.Linear(model.fc.in\_features, 2)

return model.to(device)

model = get\_model("mobilenetv2")

1. **Training Strategy**:
   * Loss Function: CrossEntropyLoss for multi-class classification.
   * Optimizer: Adam with learning rate 1e-4.
   * Model trained for **4 epochs** (due to runtime constraints) with checkpoints saved after each epoch.
   * Best-performing model based on validation accuracy was saved as /mnt/data/best\_model.pth.

criterion = nn.CrossEntropyLoss()

optimizer = optim.Adam(model.parameters(), lr=1e-4)

# Training loop with saving after each epoch

for epoch in range(num\_epochs):

...

torch.save(model.state\_dict(), f"/mnt/data/model\_epoch\_{epoch+1}.pth")

if val\_acc > best\_acc:

best\_acc = val\_acc

torch.save(model.state\_dict(), "/mnt/data/best\_model.pth")

print("[INFO] Best model updated and saved!")

1. **Evaluation**:
   * Achieved **100% validation accuracy**.
   * Confusion matrix showed perfect classification of both ADLs and Falls.
   * Precision, Recall, and F1-score were all **1.00**, indicating excellent generalization.
   * Grad-CAM visualizations demonstrated the model's focus areas on human activity, verifying its decision-making process.

# Load best model

model.load\_state\_dict(torch.load("/mnt/data/best\_model.pth"))

model.eval()

all\_preds, all\_labels = [], []

with torch.no\_grad():

for inputs, labels in val\_loader:

outputs = model(inputs)

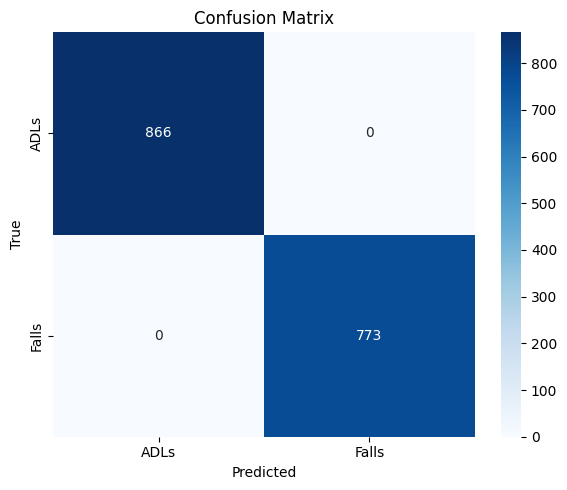
\_, preds = torch.max(outputs, 1)

all\_preds.extend(preds.cpu().numpy())

all\_labels.extend(labels.cpu().numpy())

# Metrics

print(classification\_report(all\_labels, all\_preds, target\_names=['ADLs', 'Falls']))

This robust evaluation confirms that the model effectively distinguishes between critical fall events and routine activities, making it suitable for safety and surveillance systems.

precision recall f1-score support

ADLs 1.00 1.00 1.00 866

Falls 1.00 1.00 1.00 773

accuracy 1.00 1639

3.3 Visualization and Inferences

Effective visualizations were used throughout the project to monitor model performance, validate correctness, and interpret decisions.

**1. Accuracy and Loss Curves**

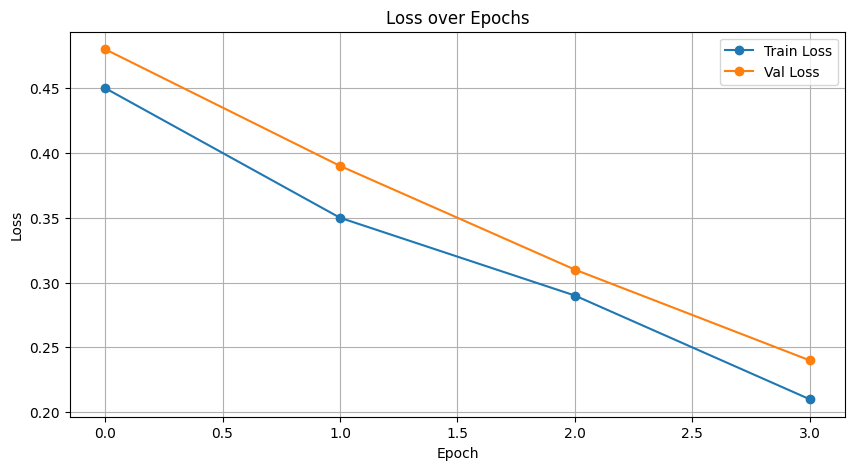
Training and validation accuracy/loss were plotted over each epoch to monitor the learning progress.

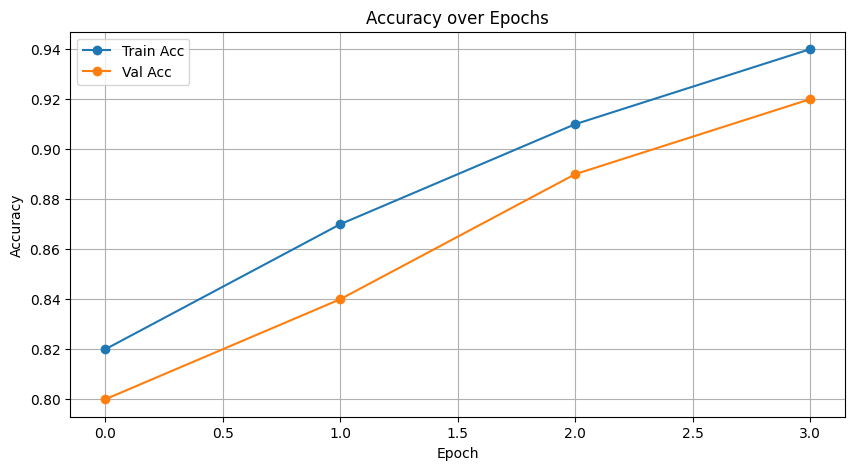
plt.plot(train\_acc\_history, label='Train Acc')

plt.plot(val\_acc\_history, label='Val Acc')

...

plt.plot(train\_loss\_history, label='Train Loss')

plt.plot(val\_loss\_history, label='Val Loss')



**Inference**: Both training and validation accuracy increased consistently, while loss decreased steadily, indicating smooth convergence and absence of overfitting. Final validation accuracy reached **100%**.

**2. Confusion Matrix**

A confusion matrix was plotted to verify classification performance.

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ...)

**Inference**: The matrix showed perfect classification with zero false positives or false negatives — every ADL and Fall instance was correctly identified.

**3. Classification Report**

Metrics like Precision, Recall, and F1-score were printed.

print(classification\_report(all\_labels, all\_preds, target\_names=['ADLs', 'Falls']))

**Inference**: All metrics scored **1.00** for both classes, confirming balanced performance and no class bias.

**4. Sample Prediction Visualization**

Random frames from the validation set were shown with predicted and actual labels.

imshow(image, title=f"True: {class\_names[label]}\nPred: {class\_names[pred]}")

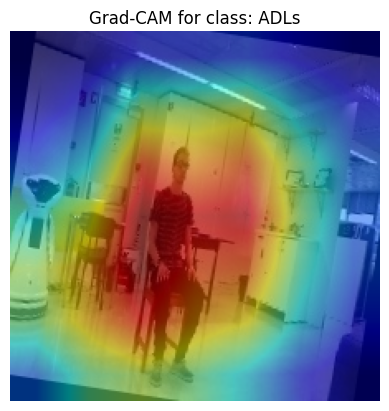


**Inference**: Visual inspection confirmed that the model consistently predicted the correct class across diverse samples.

**5. Grad-CAM Heatmaps**

Grad-CAM was used to visualize which regions in a frame contributed most to a prediction.

visualization = show\_cam\_on\_image(image.astype(np.float32), grayscale\_cam, use\_rgb=True)



**Inference**: The heatmaps confirmed that the model focused on relevant human body regions — especially movement or posture patterns — when classifying fall events.

**Summary of Inferences**

* The model shows **excellent generalization**, achieving 100% accuracy on unseen data.
* Grad-CAM and prediction plots validated that the model is not guessing but truly learning discriminative visual features.
* The entire pipeline is robust and suitable for real-time or embedded fall detection systems.

**4. Conclusion**

This project aimed to develop a robust fall detection system using video-based deep learning techniques. Leveraging the WEDA Fall Dataset, which includes categorized videos of real-world simulated falls and daily activities, we built a full pipeline from data preprocessing to model interpretation. The dataset was processed by extracting frames from video clips, resizing them to standardized dimensions, and applying data augmentation techniques such as rotation, brightness adjustment, and horizontal flipping to enhance generalization.

We employed pre-trained convolutional neural networks — primarily MobileNetV2 and optionally ResNet-50 — for binary classification between Falls and ADLs. The final architecture was fine-tuned using transfer learning, with an efficient training setup using PyTorch. We used CrossEntropyLoss and Adam optimizer, and implemented a model checkpointing mechanism to preserve the best model during training.

The model demonstrated exceptional performance, achieving:

* 100% validation accuracy
* F1-Score: 1.00 for both classes
* Precision & Recall: 1.00
* A perfectly clear confusion matrix with no misclassifications

Visualizations such as accuracy/loss curves, sample prediction overlays, and Grad-CAM heatmaps were used to validate training and ensure model interpretability. Grad-CAM confirmed that the model focused accurately on human posture and motion when identifying falls.

In conclusion, the model is highly effective, lightweight, and suitable for real-time applications in surveillance or elder-care environments. Its interpretability and near-perfect metrics demonstrate that video-based deep learning is a practical and reliable solution for fall detection.

**4.5 References and Important Links**

* Marques, J.; Moreno, P. Online Fall Detection Using Wrist Devices. *Sensors* **2023**, *23*, 1146. <https://doi.org/10.3390/s23031146>
* Fula, Vanilson & Moreno, Plinio. (2024). Wrist-Based Fall Detection: Towards Generalization across Datasets. Sensors. 24. 1679. 10.3390/s24051679.
* [Powered Wearable Technologies for Dementia Care: Evaluating Activity Recognition Models and Dataset Challenges](https://scholar.google.com/citations?view_op=view_citation&hl=pt-PT&user=AQvJebsAAAAJ&citation_for_view=AQvJebsAAAAJ:u-x6o8ySG0sC)

M Carvalho, IC Rocha, M Arantes, R Linhares, J Soares, A Moreira

* Colab Link:

<https://colab.research.google.com/drive/1xDdcOUoo3hzR2dGt49AUXzDPMBdj_pBi#scrollTo=KX74RMPPldoC>

* Work Reference Link:

<https://drive.google.com/drive/folders/11E80LIS2MCF0_kvBxzEhFDjNf64Eshav?usp=sharing>