(A PRESENTATION OF PROJECT ON TITLE)

FALL DETECTION

BY

NAIR JYOTHIKA REJI

REG NO: 230011020736

OVERVIEW

Introduction

Objectives

Technologies used

Algorithms used

Libraries used

Model building

Challenges & Limitations

Future Enhancements

INTRODUCTION

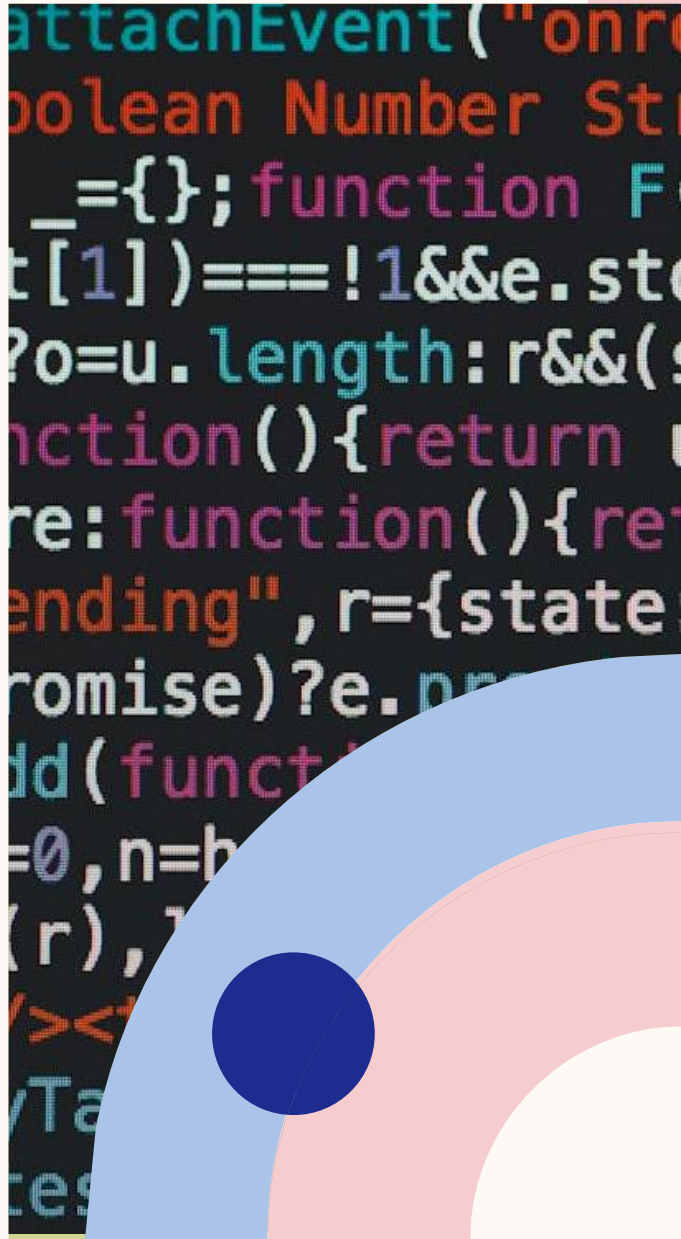
- Falls are a significant concern, particularly for the elderly and those with certain medical conditions, as they can lead to serious injuries or even death.
- The idea is to build a model that can accurately differentiate between images where a fall happens to other activities which will prove useful in healthcare surveillance systems.

OBJECTIVES

- To build a model that can identify images with falls and non-fall activities
- Usage of deep learning techniques
- Training the model using image datasets from Kaggle.

TECHNOLOGIES USED

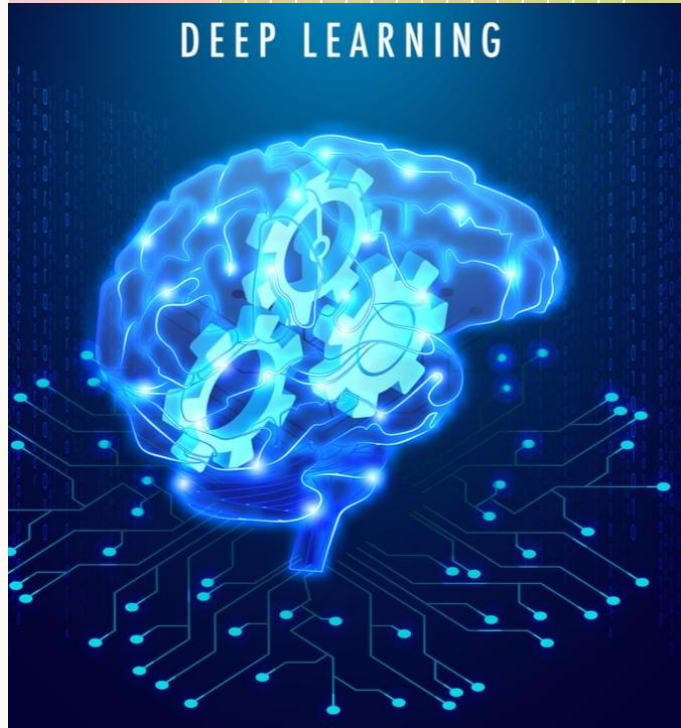
1. Python programming language
2. Jupyter Notebook



ALGORITHMS USED

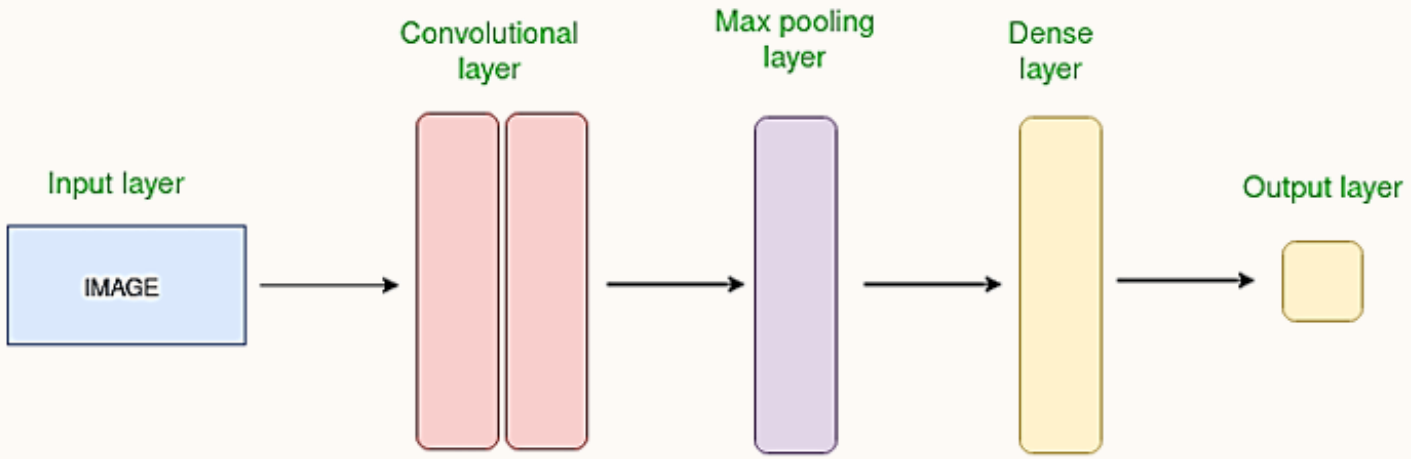
Deep learning :

- Deep learning is a specific subfield of machine learning: a new take on learning representations from data that puts an emphasis on learning successive layers of increasingly meaningful representations.
- In deep learning, these layered representations are learned via models called neural networks



Convolutional Neural Network :

Convolutional Neural Network (CNN) is the extended version of artificial neural networks (ANN) which is predominantly used to extract hierarchical features through convolutional layers.



VGG-16 :

- CNN architecture
- Proposed by the Visual Geometry Group (VGG) at the University of Oxford
- 16 layers
  - 13 convolutional layers
  - 3 fully connected layers

VGG-16 :

- Has ability to achieve strong performance on various computer vision tasks, including image classification and object recognition
- Also includes max pooling layers
- Activation function
  - ReLU
  - Softmax

LIBRARIES USED

- Tensorflow
- Keras
- NumPy
- Matplotlib
- Scikit-learn
- Keras Utilities
- OS
- Graphviz & Pydot

MODEL BUILDING

Data Preprocessing & Splitting :

- The first step in this project is encoding the labels associated with the images using OneHotEncoder() function

- Load images using os.walk()

- Resize both set of images (fall and non-fall) and convert to NumPy array.

- Labels are also reshaped into a NumPy array to align correctly with images.

- Split the dataset into training (80%) and testing (20%) subsets using train_test_split().

Building The Model:

1.Model Selection and Base Model Configuration : It involves choosing VGG16 as the base model. The parameter include_top=False excludes the top fully connected (FC) layers of VGG16 allowing us to add custom layers for our task. The pre trained weights from the ImageNet dataset are loaded enabling transfer learning.

2.Adding Custom Fully Connected Layers : Since VGG16 was originally designed for 1000-class classification, we need to add new layers tailored for binary classification (fall vs. no-fall).

Building The Model:

3. Fine-Tuning Specific Layers : The last few layers of VGG16 are made trainable, while earlier layers remain frozen to retain pre-trained features

4. Compiling the Model : Once the model architecture is defined, it needs to be compiled with Adam optimizer, binary crossentropy loss function, and evaluation metric.

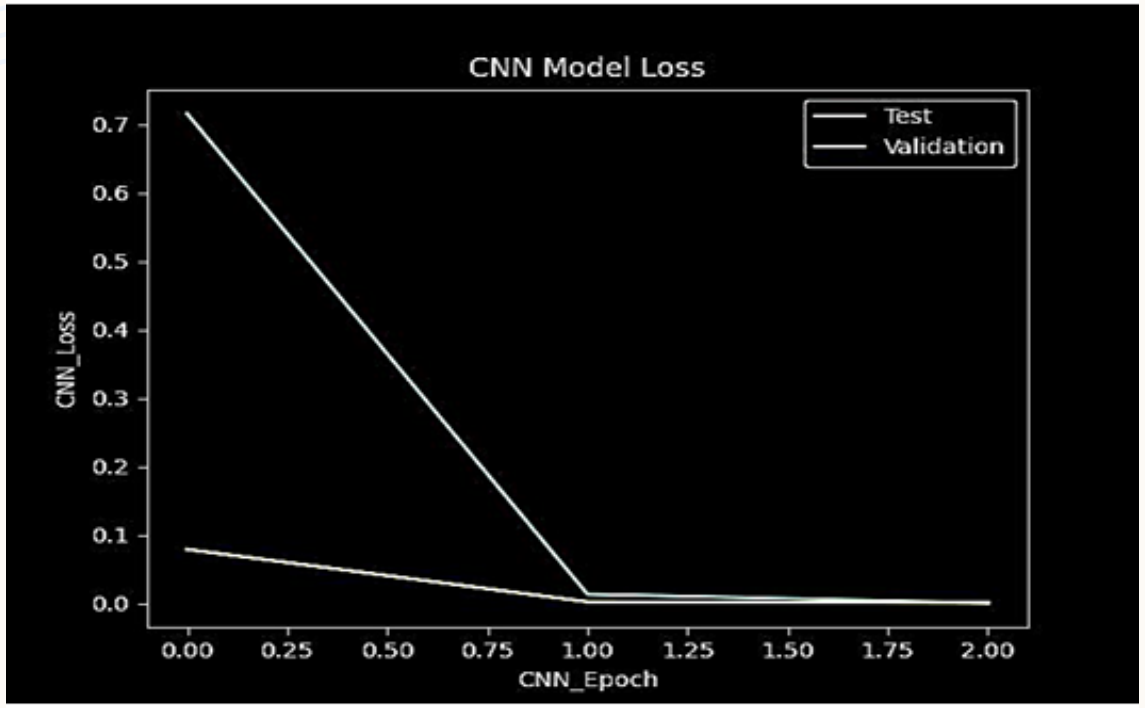
Building The Model:

```
Epoch 1/3
25/25 ━━━━━━━━━━━━━━━━━━━━ 315s 12s/step - accuracy: 0.7528 - loss: 1.9049 - val_accuracy: 0.9800 - val_loss: 0.0788
Epoch 2/3
25/25 ━━━━━━━━━━━━━━━━━━━━ 417s 17s/step - accuracy: 0.9962 - loss: 0.0237 - val_accuracy: 1.0000 - val_loss: 0.0022
Epoch 3/3
25/25 ━━━━━━━━━━━━━━━━━━━━ 418s 17s/step - accuracy: 1.0000 - loss: 6.7487e-04 - val_accuracy: 1.0000 - val_loss: 8.0710e-04
```

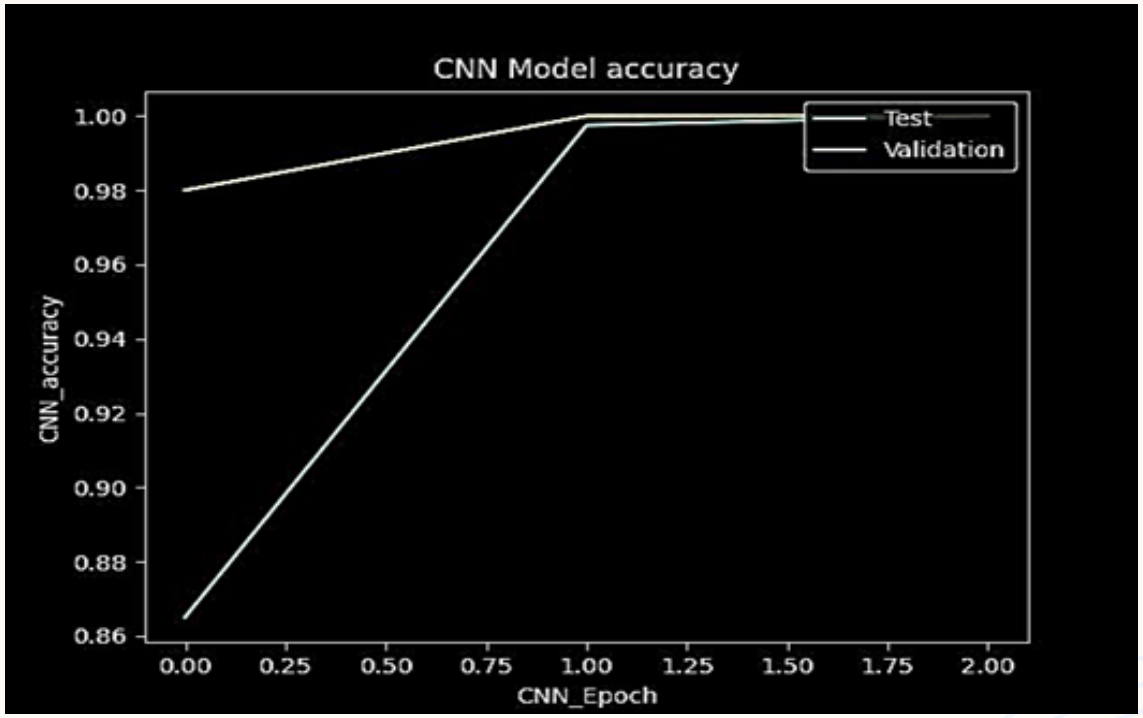
Model Performance :

	precision	recall	f1-score	support
Fall	1.00	1.00	1.00	208
NonFall	1.00	1.00	1.00	192
accuracy			1.00	400
macro avg	1.00	1.00	1.00	400
weighted avg	1.00	1.00	1.00	400

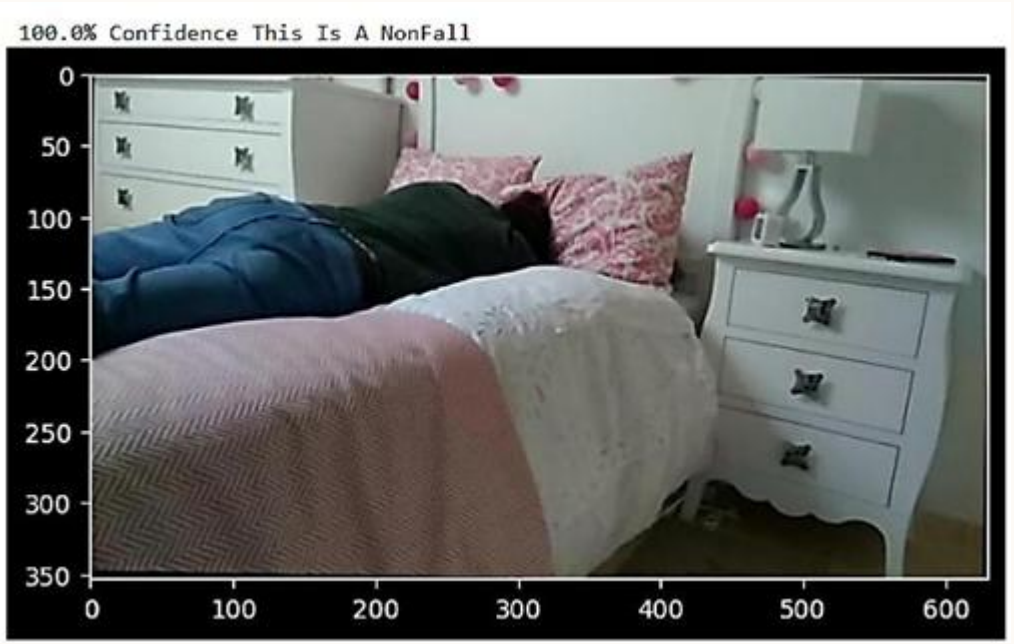
Loss value Vs Epoch plot



Accuracy value Vs Epoch plot



Model Predictions



CHALLENGES

1. Ensuring accuracy

2. Network issues while handling large datasets

3. Requirement of appropriate computational resources and hardware setup for running epochs and processing large datasets

LIMITATIONS

- Accuracy of the model based on images
- Misclassification of other activities as falls
- Appropriate dataset for detecting falls, especially those about to happen
- Making the system suitable for real-world scenarios

FUTURE ENHANCEMENTS

- Integrating a hybrid model
- Improving data quality and diversity
- Real-time deployment
- Also integrating pose estimation models to understand body posture and estimation

CONCLUSION

- Fall detection project utilizing VGG-16 model demonstrates potential of deep learning in enhancing safety and monitoring systems.
- The model successfully differentiated between fall and non fall activities.
- Incorporating several enhancements into the system can ensure robustness of the system across different environments and improve its real world applicability.
- Overall, it reinforces potential of deep learning in building smart, responsive and impactful solutions.

THANK YOU !!