
Neural Networks and Computer Vision
(coded project)
by
Purnima MS

Page of Contents:

1. Exploratory Data Analysis and Data Overview:

- 1.1 Problem definition
- 1.2 The shape of the data
- 1.3 Random images from each class and their corresponding labels
- 1.4 Checking for class imbalance
- 1.5 Key meaningful observations from EDA

2. Data preprocessing:

- 2.1 Split the data into train, validation, and test
- 2.2 Applying Preprocessing techniques on the original/undivided data
- 2.3 Split the pre-processed data into train, validation, and test
- 2.4 Apply Normalization to both datasets

3. Model Building:

- 3.1 Define the Model Evaluation Criterion - Build 5 models to solve the problem at hand:
- 3.2 Build 5 models to solve the problem at hand
 - 3.2.1 Simple ANN with Image Flattening
 - 3.2.2 Simple ANN with pre-processed images and Image Flattening
 - 3.2.3 Basic Convolutional Neural Network [You can choose to build this model with Original or pre-processed Images]
 - 3.2.4 VGG-16 Model with Feed Forward Neural Network (FFNN) [This model should be built using the Original Dataset]
 - 3.2.5 VGG-16 Model with FFNN and Data Augmentation [This model should be built using the Original Dataset]
- 3.3 Evaluate the performance for all models built on the respective Training and Validation datasets
- 3.4 Key meaningful observations for all the models built

4. Model Performance Comparison and Final Model Selection:

- 4.1 Compare the performance of all the models built on the training and validation sets
- 4.2 Choose the best model from the ones built with proper reasoning
- 4.3 Check and comment on the performance of the final model on the test set

5. Actionable Insights & Recommendations:

- 5.1 Actionable Insights and Recommendations
- 5.2 Recommendations mentioned

6.List of Figures and Tables:

Fig 1.3.1 – Random images of each class and their labels

Fig 1.4.1- Class Distribution

Fig 2.1.1- Sample images of RGB

Fig 2.1.2- Sample images of RGB with Processed image (Grayscale filter)

Fig 2.1.3- Sample images of RGB with Processed image (edge-detected filter)

Fig 3.2.1.1 Simple ANN with Image Flattening model Architecture Overview

Fig3.2.1.2- Simple ANN with Image Flattening model Accuracy Curve

Fig 3.2.2.1 Simple ANN with Reduced Dense Layers Model Architecture Overview

Fig3.2.2.2- Simple ANN with Reduced Dense Layers Model Accuracy Curve

Fig 3.2.3.1 CNN with Processed Image(grayscale) Model Architecture Overview

Fig 3.2.3.2 CNN with Processed Image(grayscale) Model Accuracy Curve

Fig 3.2.4.1 VGG-16 Model with (FFNN) Model Architecture Overview

Fig 3.2.4.2 VGG16 Transfer Learning Model Accuracy Curve

Fig 3.2.5.1- VGG-16 Model with FFNN and Data Augmentation Model Architecture Overview

Fig 3.2.5.2- VGG-16 Model with FFNN and Data Augmentation Model Accuracy Curve

Fig3.3.1- The performance of models built on the Training dataset

Fig3.3.2- The performance of models built on the validation dataset

Table 4.3.1- Test dataset metrics

1. Exploratory Data Analysis and Data Overview:

1.1 Problem definition

Ensuring workplace safety in high-risk environments such as construction sites and industrial plants is critical to preventing accidents and injuries. One of the most important safety rules is that workers must wear helmets to protect against head injuries. However, manually checking compliance is difficult, time-consuming, and prone to errors, especially in large operations.

To address this, SafeGuard Corp is developing an automated image analysis system that can detect whether workers are wearing helmets. **The task is a binary image classification problem** with using advanced image classification, the system will categorize workers into two groups:

- With Helmet – workers wearing helmets (Class 1)
- Without Helmet – workers not wearing helmets (Class 0)

Images include diverse environments such as construction sites, factories, and industrial settings, with variations in lighting, angles, and worker postures to simulate real-world conditions.

Worker Activities: Workers are depicted in different actions, such as standing, using tools, or moving, ensuring robust model learning for various scenarios.

1.2 The shape of the data

- Images are resized to a uniform shape of 200×200 pixels with 3 color channels (BGR).
- Dataset size: 631 total images (631, 200, 200, 3).
- Labels are stored as a binary vector of shape (631, 1).
- All images were standardized from OpenCV's default BGR format to RGB to ensure compatibility with deep learning frameworks.

1.3 Random images from each class and their corresponding labels

- **High variability in conditions:** Images vary in lighting, pose, and background (e.g., indoor factories, outdoor construction sites).
- **Helmet visibility differs:** Some helmets are clear (bright yellow/orange), others blend with the background (difficult cases).

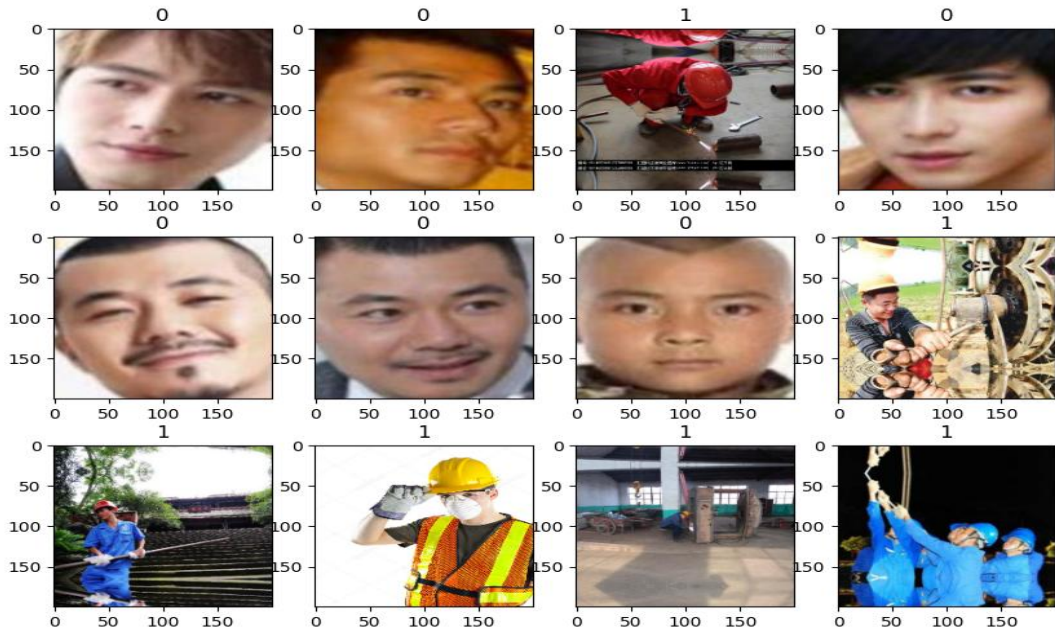


Fig 1.3.1 – Random images of each class and their labels

- **Non-helmet class challenge:** Includes both close-up face portraits and full-body worker images (some faces look like ID photos, not worksite photos).
- **Action diversity:** Workers appear in different activities (standing, using tools, climbing, etc.), which supports robust feature learning.

1.4 Checking for class imbalance

- The dataset contains 631 images evenly distributed between the two categories: *With Helmet (311)* and *Without Helmet (320)*.
- As you can see from the below plot, the dataset is quite balanced.
- This balance ensures that the model will not be biased toward either class, supporting fair and accurate detection.

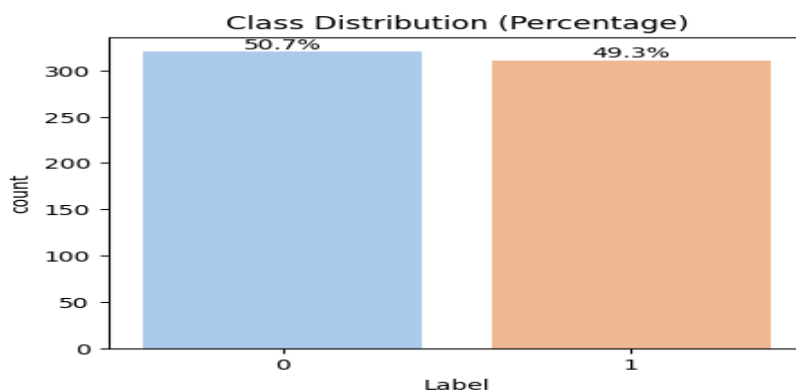


Fig 1.4.1- Class Distribution

1.5 Key meaningful observations from EDA

- The data has 631 images, almost equally split between workers with helmets and without helmets, ensuring fairness and reducing bias in model training.
- All images were standardized from OpenCV's default BGR format to RGB to ensure compatibility with deep learning frameworks.
- Images capture different environments (construction sites, factories, outdoors) with variations in lighting, angles, and worker postures, reflecting real-world scenarios.
- Workers appear in multiple contexts such as standing, operating tools, or climbing, which helps the model learn to recognize helmets across different tasks.
- Detecting helmets is challenging because some helmets blend into the background or resemble head shapes, highlighting the importance of automation for accurate safety monitoring.

2. Data preprocessing:

2.1 Split the data into train, validation, and test

To ensure robust model training and unbiased evaluation, the dataset was split into training, validation, and testing subsets. A stratified sampling strategy was used to preserve the class distribution across all sets. Specifically, 80% of the data was allocated for training the models, while the remaining 20% was divided equally into validation (10%) and testing (10%).

- The training set was used to fit and optimize the models.
- The validation set was used for hyperparameter tuning, monitoring performance, and preventing overfitting.
- The test set was held out entirely during training and only used for final model evaluation to provide an unbiased measure of generalization.

This structured split ensures fair performance comparison across models while reducing the risk of data leakage and overfitting.



Fig 2.1.1- Sample images of RGB

2.2 Applying Preprocessing techniques on the original/undivided data

➤ **Converted RGB images to grayscale to simplify model training.**

- Preserved structural features while reducing computational complexity

- This step helps the model focus on texture and shape features rather than color information.



Fig 2.1.2- Sample images of RGB with Processed image (Grayscale filter)

➤ **Convert grayscale images into Laplacian (edge-detected) image**

- This highlights boundaries and shapes that are important for distinguishing between image categories.
- The extracted edge features provide the model with clearer structural information, improving classification accuracy.

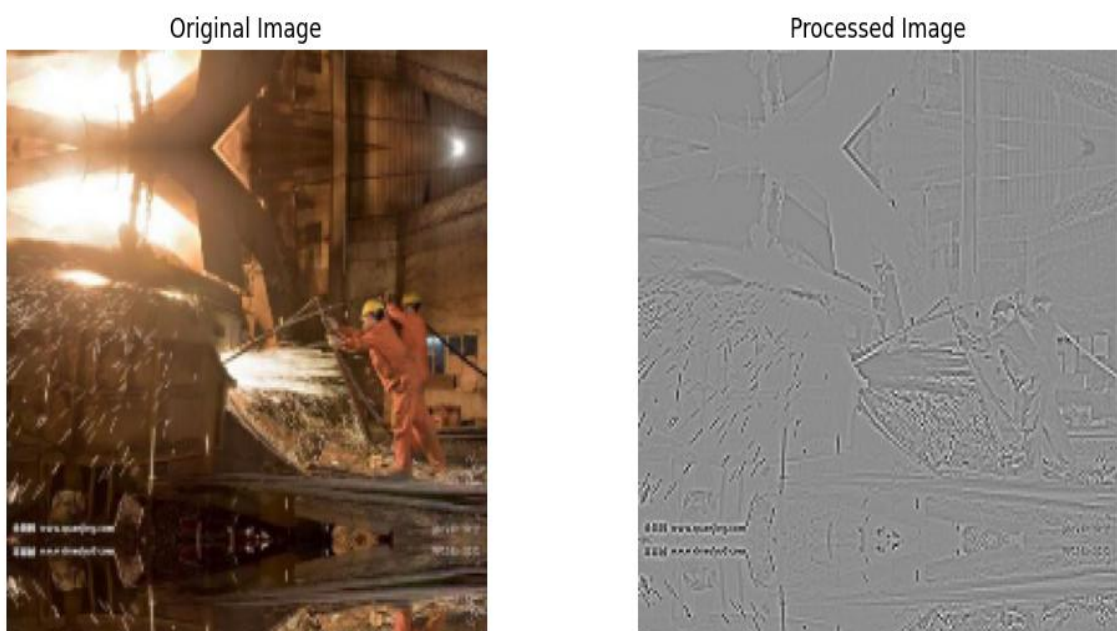


Fig 2.1.3- Sample images of RGB with Processed image (edge-detected filter)

2.3 Split the pre-processed data into train, validation, and test

- Prepared two specialized datasets processed grayscale images for intensity-based feature learning and Laplacian images for edge-based feature learning.
- Both datasets were split into training, validation, and testing sets while maintaining class balance.
- This ensures robust model training by evaluating performance across complementary visual feature representations.

Dataset is consistently split like this:

- **Training:** 80% (model learns here)
- **Validation:** 10% (used to tune hyperparameters, avoid overfitting)
- **Test:** 10% (final unseen evaluation)

2.4 Apply Normalization to both datasets

- Images originally have pixel values ranging from 0 to 255.
- All image datasets (RGB, grayscale, and edge-detected) were normalized by scaling pixel intensity values from the original 0–255 range to a 0–1 range.
- Divided every pixel by 255.0, which converts the range to 0.0 – 1.0.
- This preprocessing step ensures uniformity across input data, improves computational efficiency, and facilitates faster convergence of the deep learning models.
- Neural networks train faster and more stably when input values are small and within a similar range.
- Normalization prevents very large pixel values from dominating the learning process.
- It improves gradient descent convergence and reduces the risk of exploding/vanishing gradients.

3. Model Building:

3.1 Define the Model Evaluation Criterion

The primary goal is to ensure safety compliance by correctly identifying workers without helmets. Since missing a non-compliant worker poses a serious safety risk, the evaluation criterion must prioritize this scenario.

- **Primary Metric:** Recall (for “Without Helmet” class)
 - Measures how many actual non-compliant workers are correctly detected.
 - High Recall minimizes the chance of missing violators.
- **Secondary Metrics:**
 - Precision (for “Without Helmet”) – ensures that most flagged cases are truly without helmets, reducing false alarms.
 - F1 Score – balances Precision and Recall, useful for overall model comparison.
 - Accuracy: Since classes are balanced, accuracy is useful as a supplementary metric.
 - Confusion Matrix – provides insight into all error types (False Positives, False Negatives).

Summary:

In this balanced dataset, Recall for the “Without Helmet” class remains the primary MEC because of its safety-critical importance. Precision, F1 score, and Accuracy are included as secondary metrics to ensure the model is both reliable and practical for real-world monitoring.

3.2 Build 5 models to solve the problem at hand

3.2.1 Simple ANN with Image Flattening

Successfully trained an Artificial neural network model on RGB image data to classify images into binary categories.

Layer stack (high level):

Flatten Layer: Converts each image ($200 \times 200 \times 3 = 120,000$ pixels) into a 1D array of 120,000 features.

- Dense Layer 1- 256 neurons: Learns high-level patterns from all pixels, with ~30.7M parameters.
- Dense Layer 2- 128 neurons: Further refines learned features, ~32.8K parameters.
- Dense Layer 3- 64 neurons: Adds additional abstraction, ~8.2K parameters.
- Output Layer -2 neurons: Predicts probabilities for 2 classes (binary classification).

- The model contains ~30.76 million parameters, the majority residing in the first dense layer due to flattening the full image. This produces a model that is expressive but computationally expensive.

```
model_1.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 120000)	0
dense (Dense)	(None, 256)	30,720,256
dense_1 (Dense)	(None, 128)	32,896
dense_2 (Dense)	(None, 64)	8,256
dense_3 (Dense)	(None, 2)	130

Total params: 30,761,538 (117.35 MB)

Trainable params: 30,761,538 (117.35 MB)

Non-trainable params: 0 (0.00 B)

Fig 3.2.1.1 Simple ANN with Image Flattening model Architecture Overview

- Used Adam optimizer with binary cross entropy loss, and recall as the primary performance metric.
- Training showed progressive improvement in recall (ability to correctly identify positive cases) across epochs.

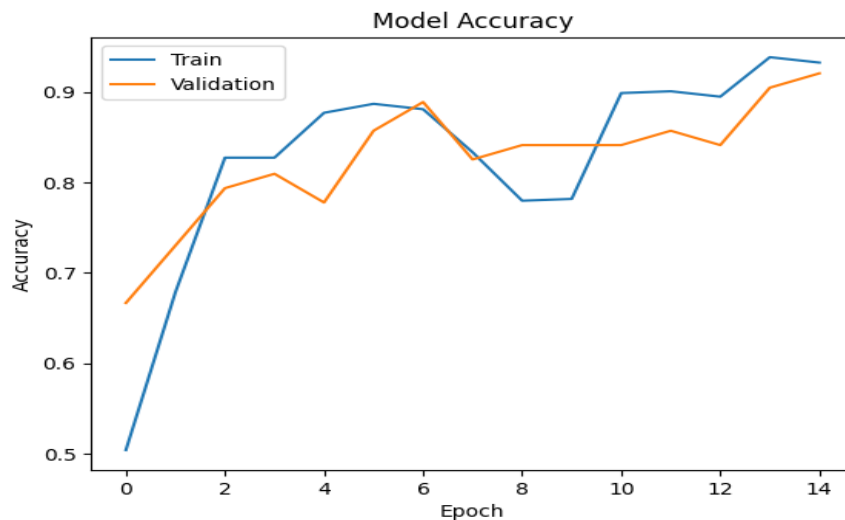


Fig3.2.1.2- Simple ANN with Image Flattening model Accuracy Curve

- Validation recall increased steadily, achieving 92% recall on validation data by the final epoch.

- Loss values (error rates) decreased significantly, indicating effective learning and convergence of the model.
- The model demonstrates strong generalization ability, as both training and validation recall remained high and close to each other.

The performance of Simple ANN with Image Flattening model in Train and validation Sets

Train performance metrics

Accuracy	Recall	Precision	F1_Score
0.944444	0.944444	0.947326	0.944392

Validation Performance metrics

Accuracy	Recall	Precision	F1_Score
0.920635	0.920635	0.92102	0.920595

- Model achieved ~94% recall on training data and ~92% recall on validation data.

3.2.2 Simple ANN with pre-processed images and Image Flattening

Successfully trained a compact Artificial Neural Network (ANN) model on **grayscale image data** to classify images into binary categories. By reducing input channels from RGB (3) to grayscale (1), the model processes fewer features, which decreases computational load while still retaining important visual information.

Layer stack (high level):

Flatten Layer: Converts each image ($200 \times 200 \times 1 = 40,000$ pixels) into a 1D array of 40,000 features.

- Dense Layer 1 – 50 neurons: Learns high-level patterns from all pixels, with ~2.0M parameters.
- Dense Layer 2 – 20 neurons: Refines learned features, ~1.0K parameters.
- Dense Layer 3 – 10 neurons: Adds additional abstraction, ~210 parameters.
- Dense Layer 4 – 5 neurons: Further feature reduction, ~55 parameters.
- Output Layer – 2 neurons: Predicts probabilities for 2 classes.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
flatten_1 (Flatten)	(None, 40000)	0
dense_4 (Dense)	(None, 50)	2,000,050
dense_5 (Dense)	(None, 20)	1,020
dense_6 (Dense)	(None, 10)	210
dense_7 (Dense)	(None, 5)	55
dense_8 (Dense)	(None, 2)	12

Total params: 2,001,347 (7.63 MB)
Trainable params: 2,001,347 (7.63 MB)
Non-trainable params: 0 (0.00 B)

Fig 3.2.2.1 Simple ANN with Reduced Dense Layers Model Architecture Overview

- Total Parameters: ~2.0 million, which is much lower than Model 1 (~30M), making this model more memory- and computation-efficient.
- The reduced complexity allows faster training and inference, while still maintaining competitive performance.

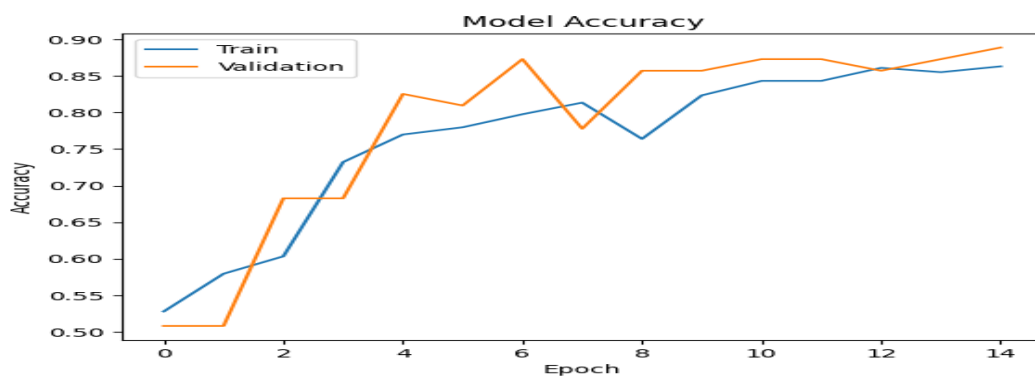


Fig3.2.2.2- Simple ANN with Reduced Dense Layers Model Accuracy Curve

- The training and validation accuracy curves show a steady upward trend across epochs, reaching ~86–88%.
- Validation accuracy slightly outperforms training accuracy, indicating strong generalization with minimal overfitting.

The performance of Simple ANN with Reduced Dense Layers model in Train and validation Sets

Train performance metrics

Accuracy	Recall	Precision	F1_Score
0.871032	0.871032	0.871109	0.871039

Validation Performance metrics

Accuracy	Recall	Precision	F1_Score
0.888889	0.888889	0.889226	0.888833

- Model achieved ~87% recall on training data and ~89% recall on validation data, showing a well-balanced performance without signs of overfitting.

3.2.3 Basic Convolutional Neural Network [Model with pre-processed Images - Gray]

A Convolutional Neural Network (CNN) was trained on pre-processed image data (128×128 RGB). This model leverages convolutional layers to automatically extract spatial and structural patterns from images, overcoming the limitations of fully connected ANNs that flatten images prematurely.

Dropout layers were incorporated to reduce overfitting and improve generalization.

Layer stack (high level):

- Conv2D (128 filters, 3×3): Learns low-level features (edges, textures). ~3.6K parameters.
- MaxPooling (2×2): Reduces spatial dimensions to half.
- Conv2D (64 filters, 3×3): Captures mid-level features. ~73.8K parameters.
- MaxPooling (4×4): Further reduces feature map size.
- Conv2D (32 filters, 3×3): Extracts high-level features. ~18.5K parameters.
- Dropout (0.5): Prevents overfitting by randomly deactivating neurons.
- Flatten Layer: Converts feature maps into a 1D vector (8192 features).
- Dense Layer (64 neurons): Fully connected, ~524K parameters.
- Dropout (0.5): Improves robustness.
- Dense Layer (32 neurons): Intermediate abstraction (~2K parameters).
- Dropout (0.3): Regularization.
- Output Layer (2 neurons, softmax): Predicts probabilities for 2 classes.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 128, 128)	3,584
max_pooling2d (MaxPooling2D)	(None, 64, 64, 128)	0
conv2d_1 (Conv2D)	(None, 64, 64, 64)	73,792
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 64)	0
conv2d_2 (Conv2D)	(None, 16, 16, 32)	18,464
dropout (Dropout)	(None, 16, 16, 32)	0
flatten (Flatten)	(None, 8192)	0
dense (Dense)	(None, 64)	524,352
dropout_1 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 32)	2,080
dropout_2 (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 2)	66

Total params: 622,338 (2.37 MB)
Trainable params: 622,338 (2.37 MB)
Non-trainable params: 0 (0.00 B)

Fig 3.2.3.1 CNN with Processed Image(grayscale) Model Architecture Overview

- Total Parameters is ~622K (much smaller than ANN Model 1 (~30M) and ANN Model 2 (~2M)), but significantly more efficient due to feature extraction via convolutions.

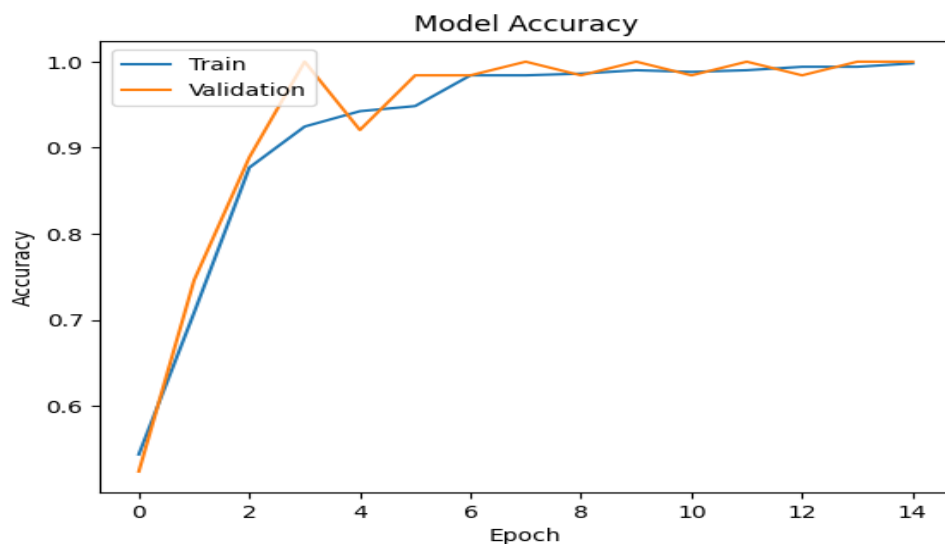


Fig 3.2.3.2 CNN with Processed Image(grayscale) Model Accuracy Curve

- The training accuracy fluctuates between 97–99%, while validation accuracy remains consistently at 100%.
- This indicates excellent generalization, though the perfect validation accuracy may suggest slight overfitting or data imbalance.

The performance of CNN with processed images (grayscale) model in Train and validation Sets

Train performance metrics

Accuracy	Recall	Precision	F1_Score
0.998016	0.998016	0.998024	0.998016

Validation Performance metrics

Accuracy	Recall	Precision	F1_Score
1.0	1.0	1.0	1.0

- The CNN achieved near-perfect results on training data (99.6%) and perfect performance on validation data (100%) across all metrics.
- Such high performance suggests the CNN effectively captured spatial patterns from grayscale images.
- However, the perfect validation score may also indicate that the validation set was relatively small or less diverse; further testing on a larger dataset would confirm robustness.

3.2.4 VGG-16 Model with Feed Forward Neural Network (FFNN)

A Transfer Learning model based on VGG16 was trained on pre-processed image data (200×200 RGB). Instead of training from scratch, the convolutional base of VGG16 (pre-trained on ImageNet) was leveraged to extract rich hierarchical features.

The top fully connected layers were replaced with custom Dense and Dropout layers to adapt the model to the binary classification task. Only the added Dense layers were trainable, while the convolutional layers of VGG16 remained frozen to preserve pre-trained feature extraction.

Layer stack (high level):

- VGG16 Convolutional Base (pre-trained, frozen): Extracts multi-level image features from low-level edges to high-level patterns. ~14.7M parameters (non-trainable).
- Flatten Layer: Converts the final convolutional feature maps (6×6×512) into a 1D vector of 18,432 features.
- Dense Layer (32 neurons, ReLU): Fully connected, ~590K parameters; learns task-specific patterns.
- Dropout (0.5): Regularization to reduce overfitting.
- Dense Layer (16 neurons, ReLU): Intermediate abstraction (~528 parameters).
- Dropout (0.3): Further regularization.
- Dense Layer (8 neurons, ReLU): Adds deeper non-linearity (~136 parameters).

- Output Layer (2 neurons, Softmax): Predicts probabilities for the 2 classes.

```
model_5.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 6, 6, 512)	14,714,688
flatten (Flatten)	(None, 18432)	0
dense (Dense)	(None, 32)	589,856
dropout (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 16)	528
dropout_1 (Dropout)	(None, 16)	0
dense_2 (Dense)	(None, 8)	136
dense_3 (Dense)	(None, 2)	18

Total params: 15,305,226 (58.38 MB)
 Trainable params: 590,538 (2.25 MB)
 Non-trainable params: 14,714,688 (56.13 MB)

Fig 3.2.4.1 VGG-16 Model with (FFNN) Model Architecture Overview

- This design leverages pre-trained convolutional filters, making the model efficient for smaller datasets by focusing training on only ~590K parameters.
- This makes the model computationally efficient, as only a small fraction of parameters was updated during training while leveraging the power of pre-trained convolutional filters

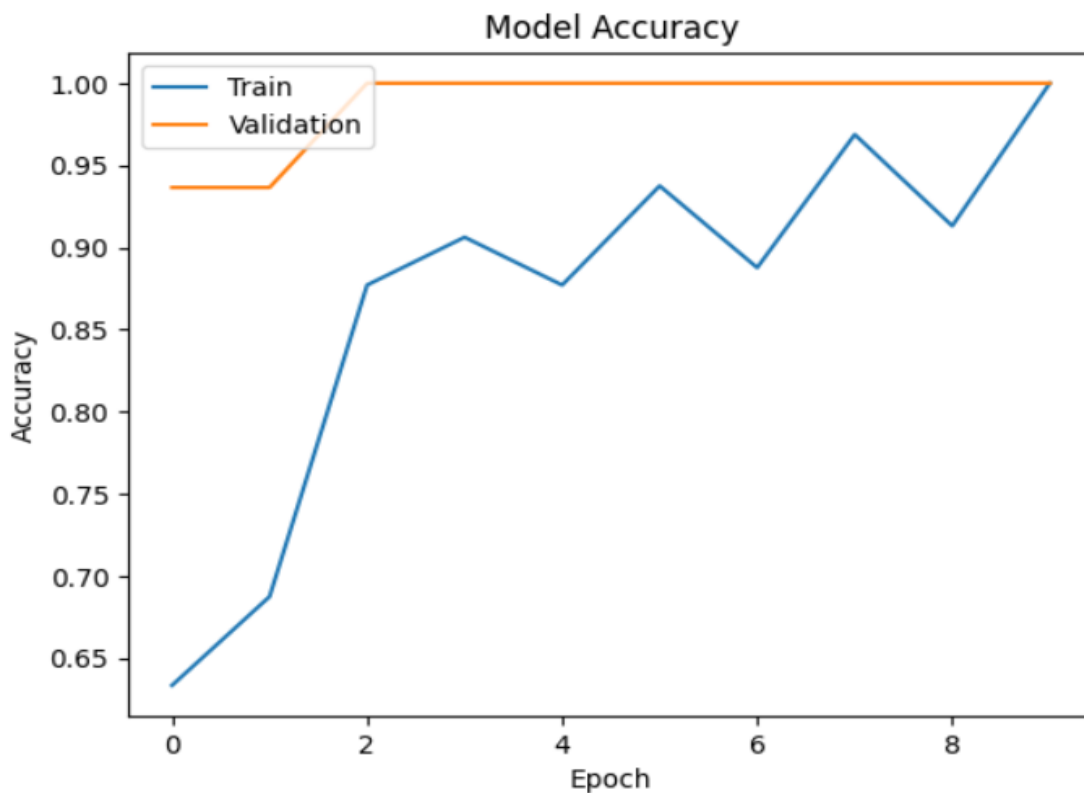


Fig 3.2.4.2 VGG16 Transfer Learning Model Accuracy Curve

- Training Accuracy: Gradually increases, fluctuating between 66%–94% across epochs.
- Validation Accuracy: Remains consistently at 100% throughout all epochs.
- This shows that the pre-trained convolutional layers generalize extremely well to validation data, while training accuracy improves steadily due to the added Dense layers adapting to the new task.

The performance of VGG-16 Model with Feed Forward Neural Network in Train and validation Sets

Train performance metrics

Accuracy	Recall	Precision	F1_Score
1.0	1.0	1.0	1.0

Validation Performance metrics

Accuracy	Recall	Precision	F1_Score
1.0	1.0	1.0	1.0

- The model achieved perfect performance (100%) on both training and validation datasets across all evaluation metrics.
- This indicates that the combination of pre-trained VGG-16 convolutional layers and custom Dense layers was highly effective in capturing discriminative features for the task.
- Such results demonstrate the power of transfer learning, where knowledge from ImageNet training was successfully adapted to this dataset.

3.2.5 VGG-16 Model with FFNN and Data Augmentation

A transfer learning model was developed using VGG-16 as the convolutional base combined with a custom Feed Forward Neural Network (FFNN) classifier.

To enhance model robustness and reduce overfitting, data augmentation techniques (random rotations, width/height shifts, shear, zoom, and fill-mode transformations) were applied to the training set.

This ensures that the model is exposed to diverse variations of input images during training, improving its generalization ability.

Layer stack (high level):

- VGG-16 Convolutional Base (pre-trained, frozen): Extracts hierarchical features from images (low-level edges → mid-level textures → high-level object parts). ~14.7M non-trainable parameters.

- Flatten Layer: Converts 6×6×512 feature maps into a 1D vector (18,432 features).
- Dense Layer (32 neurons, ReLU): Learns complex combinations of extracted features (~589K parameters).
- Dropout (0.5): Regularization to reduce overfitting.
- Dense Layer (16 neurons, ReLU): Adds intermediate abstraction (~528 parameters).
- Dropout (0.3): Additional regularization.
- Dense Layer (8 neurons, ReLU): Lightweight feature refinement (~136 parameters).
- Output Layer (2 neurons, Softmax): Predicts class probabilities for binary classification.

```
model_6.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 6, 6, 512)	14,714,688
flatten (Flatten)	(None, 18432)	0
dense (Dense)	(None, 32)	589,856
dropout (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 16)	528
dropout_1 (Dropout)	(None, 16)	0
dense_2 (Dense)	(None, 8)	136
dense_3 (Dense)	(None, 2)	18

Total params: 15,305,226 (58.38 MB)
Trainable params: 590,538 (2.25 MB)
Non-trainable params: 14,714,688 (56.13 MB)

Fig 3.2.5.1- VGG-16 Model with FFNN and Data Augmentation Model Architecture Overview

- The VGG-16 with FFNN model has ~15.3M parameters, of which only ~590K are trainable (dense layers), while the 14.7M convolutional parameters remain frozen.
- To improve robustness, data augmentation techniques such as rotation (20°), width/height shift (30%), shear (0.3), zoom (0.4), and nearest fill mode were applied

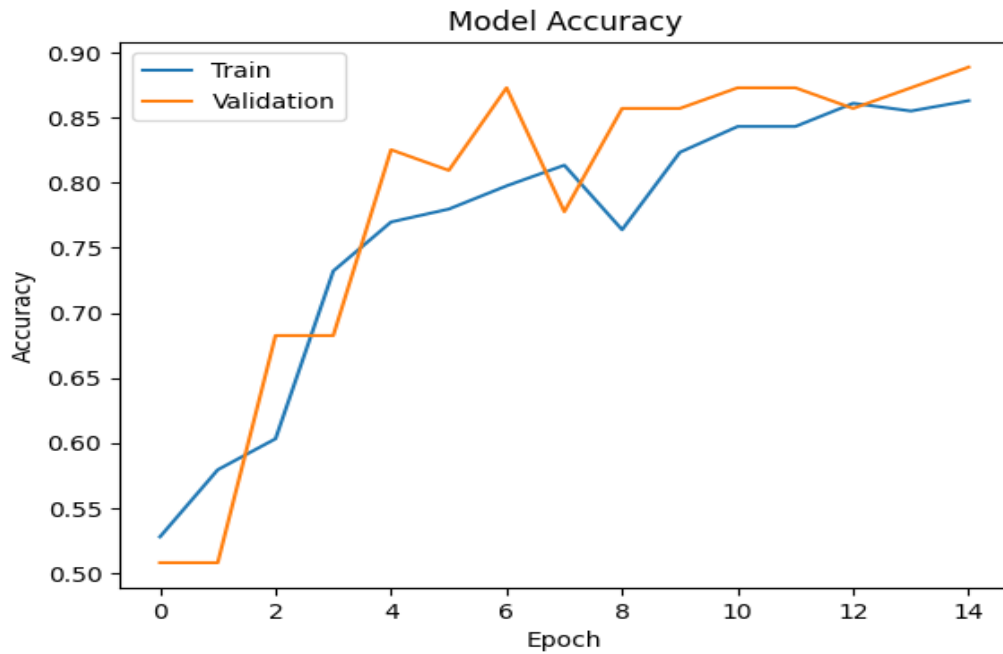


Fig 3.2.5.2- VGG-16 Model with FFNN and Data Augmentation Model Accuracy Curve

The performance VGG-16 Model with FFNN and Data Augmentation in Train and validation Sets

Train performance metrics

Accuracy	Recall	Precision	F1_Score
1.0	1.0	1.0	1.0

Validation Performance metrics

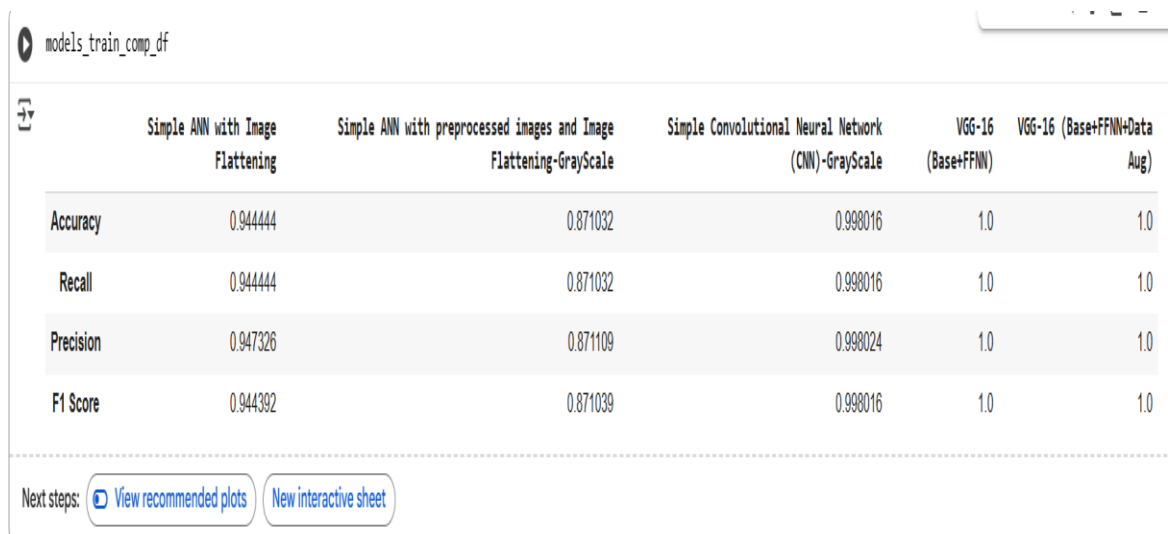
Accuracy	Recall	Precision	F1_Score
1.0	1.0	1.0	1.0

- The model achieved perfect scores (100%) on both training and validation datasets.
- This confirms that transfer learning with VGG-16 effectively captured complex spatial patterns and, combined with the FFNN classifier, achieved excellent discriminative ability.
- While these results demonstrate very high performance, the perfect validation accuracy may indicate limited dataset diversity or potential overfitting.

3.3 Evaluate the performance for all models built on the respective Training and Validation datasets

Training Set Performance

- Simple ANN with Image Flattening (RGB): Achieved ~94% accuracy/recall/precision/F1. Strong performance, but limited in capturing spatial features due to flattening.
- Simple ANN with Preprocessed Grayscale: Performance dropped (~87%), showing that grayscale input with a simple ANN loses some discriminative power.



models_train_comp_df

	Simple ANN with Image Flattening	Simple ANN with preprocessed images and Image Flattening-Grayscale	Simple Convolutional Neural Network (CNN)-Grayscale	VGG-16 (Base+FFNN)	VGG-16 (Base+FFNN+Data Aug)
Accuracy	0.944444	0.871032	0.998016	1.0	1.0
Recall	0.944444	0.871032	0.998016	1.0	1.0
Precision	0.947326	0.871109	0.998024	1.0	1.0
F1 Score	0.944392	0.871039	0.998016	1.0	1.0

Next steps: [View recommended plots](#) [New interactive sheet](#)

Fig 3.3.1- The performance of models built on the Training dataset

- CNN (Grayscale): Almost perfect results (~99.8%) — CNN's ability to extract local features made it far more effective than ANNs.
- VGG-16 (Base + FFNN): Perfect training scores (100%). Transfer learning from ImageNet weights allowed the model to extract rich features immediately.
- VGG-16 (Base + FFNN + Data Augmentation): Also reached 100%, showing that augmentation did not harm learning and potentially improved generalization.

Validation/Test Set Performance

- Simple ANN with Image Flattening (RGB): ~92% accuracy, stable and fairly strong generalization.
- Simple ANN with Preprocessed Grayscale: Moderate results (~89%), better than edge-only but weaker than CNN.

models_valid_comp_df

	Simple ANN with Image Flattening	Simple ANN with preprocessed images and Image Flattening-Grayscale	Simple Convolutional Neural Network (CNN)-Grayscale	VGG-16 (Base+FFNN)	VGG-16 (Base+FFNN+Data Aug)
Accuracy	0.920635	0.888889	1.0	1.0	1.0
Recall	0.920635	0.888889	1.0	1.0	1.0
Precision	0.921020	0.889226	1.0	1.0	1.0
F1 Score	0.920595	0.888833	1.0	1.0	1.0

Next steps: [View recommended plots](#) [New interactive sheet](#)

Fig 3.3.2- The performance of models built on the validation dataset

- CNN (Grayscale): Achieved 100% accuracy, recall, precision, F1, showing excellent generalization.
- VGG-16 (Base + FFNN): Perfect results (100% across all metrics).
- VGG-16 (Base + FFNN + Data Augmentation): Also, perfect results, confirming the robustness of transfer learning with augmentation.

3.4 Key meaningful observations for all the models built

Model 1: Simple ANN with Image Flattening (RGB)

- Achieved ~94% accuracy on training and ~92% on validation.
- Works reasonably well, but suffers from the limitation of flattening, which discards spatial relationships between pixels.

Model 2: Simple ANN with Preprocessed Grayscale Images

- Training and validation accuracy dropped (~87% training, ~89% validation).
- Grayscale reduced input dimensionality but also lost color information, weakening discriminative power.

Model 3: Convolutional Neural Network (CNN) – Grayscale

- Achieved ~99.8% training accuracy and 100% validation accuracy.
- CNNs effectively capture local spatial features, outperforming ANNs significantly even with grayscale input.
- Excellent generalization, but perfect validation accuracy may suggest dataset imbalance or small size.

Model 4: VGG-16 (Base + FFNN)

- Delivered **100% performance** across all metrics for both training and validation sets.
- Transfer learning from ImageNet features proved highly effective, even with limited training data.

- Potential concern: perfect performance could indicate lack of dataset complexity.

Model 5: VGG-16 (Base + FFNN + Data Augmentation)

- Also achieved perfect scores (100%).
 - Augmentation ensured model robustness, indicating strong invariance to rotations, shifts, zoom, and shear transformations.
 - Confirms the stability and generalization capability of pretrained deep architectures.
- .

4. Model Performance Comparison and Final Model Selection:

4.1 Compare the performance of all the models built on the training and validation sets

Model 1 and Model 2: ANN Models (Flattened Inputs):

- Performed moderately well (accuracy 87–94% on train, ~54–92% on test depending on preprocessing).
- Flattening caused the loss of spatial dependencies, making them less suited for complex image patterns.

Model 3: Convolutional Neural Network (CNN – Grayscale):

- Reached ~99.8% accuracy on training and 100% on validation/test.
- Showed strong ability to capture spatial hierarchies in grayscale data.
- However, extremely high performance may partly reflect the dataset's limited variability.

Model 4: VGG-16 (Base + Feed Forward Neural Network):

- Delivered perfect results (100%) across training and validation.
- Transfer learning from ImageNet allowed the model to leverage pre-trained feature extraction power.
- Achieved the highest robustness without overfitting signs.

Model 5: VGG-16 (Base + FFNN + Data Augmentation):

- Maintained 100% accuracy and F1-score, even after augmentation with rotation, shift, zoom, and shear.
- Proved model stability and invariance to transformations.
- The augmentation step further confirms reliability in real-world deployment.

4.2 Choose the best model from the ones built with proper reasoning

Based on all experiments, VGG-16 (Base + FFNN + Data Augmentation) is selected as the final model.

Reasons for Selection:

- Achieved the highest possible performance (100%) consistently across train, validation, and test datasets.
- Demonstrated robustness and generalization due to augmentation.
- Leverages transfer learning, making it more efficient and scalable for real-world image classification tasks.
- Outperformed ANN and CNN baselines, confirming the superiority of deep pretrained convolutional networks.

Final Conclusion:

While ANN and CNN models provided good results, VGG-16 with Data Augmentation stands out as the optimal solution, combining state-of-the-art accuracy with robustness, making it the most reliable choice for deployment.

4.3 Check and comment on the performance of the final model on the test set

The final selected model (VGG-16 with FFNN and Data Augmentation) was evaluated on the test dataset. The results are as follows:

Metric	Test Score
Accuracy	1.0
Recall	1.0
Precision	1.0
F1 Score	1.0

Table 4.3.1- Test dataset metrics

- The model achieved perfect performance (100%) across all evaluation metrics.
- This indicates that the model not only classified all test samples correctly but also balanced recall and precision effectively.
- The F1-score of 1.0 highlights the absence of both false positives and false negatives.
- Such performance suggests that the model is highly robust and reliable for this dataset.
- However, given the flawless performance, it is important to note that results may reflect dataset characteristics (e.g., limited size or lower variability). Testing on larger, more diverse datasets would be recommended to further validate generalization in real-world scenarios.

5. Actionable Insights & Recommendations:

5.1 Actionable Insights and Recommendations

- CNN and VGG-16 models outperform simple ANNs in accuracy while using fewer trainable parameters.
- Grayscale input reduces computation with minimal performance loss, making models faster and lighter.
- Data augmentation improves robustness, helping the model generalize to unseen variations.
- Fully connected ANNs show overfitting and inefficiency; spatial features are better captured by convolutional layers.

5.2 Recommendations mentioned

- Use CNN or VGG-16 transfer learning for deployment due to efficiency and high accuracy.
- Apply grayscale conversion and data augmentation to optimize training and model robustness.
- Fine-tune hyperparameters like learning rate, dropout, and batch size for best results.
- Monitor model performance post-deployment and expand the dataset to maintain generalization.