Report: Image Classification

Objective:

The objective of this project was to build and evaluate a machine learning model to classify images into two categories: **Defective** and **Non-Defective**.

Dataset Overview:

Platform: Kaggle

Filename: industrial-equipment

The dataset consists of labeled images depicting two categories:

* **Defective**: Images showing items that have defects.
* **Non-Defective**: Images showing items without defects.

Methodolgy:

Data Preprocessing:

1. **Data Loading and Augmentation**:

* The dataset was loaded using the ImageDataGenerator class from Keras. Data augmentation techniques were applied to increase the diversity of the training data and prevent overfitting.
* The images were resized to (224, 224), which is the required input size for the **MobileNetV2** model.

1. **Normalization**:

* The images were rescaled by dividing by 255 to normalize the pixel values to the range [0, 1].

Model Architecture:

1. **Base Model (MobileNetV2)**:

* We utilized **MobileNetV2**, a pre-trained model available in Keras, as the base architecture. MobileNetV2 is designed to be lightweight and efficient while still providing high accuracy, making it suitable for image classification tasks.
* MobileNetV2 is pre-trained on the **ImageNet** dataset, providing it with robust feature extraction capabilities.

1. **Custom Layers**:

* After the base model, a **GlobalAveragePooling2D** layer was added to convert the 2D feature maps into a 1D vector.
* A **Dense layer** with 1024 units and **ReLU** activation was added to capture more complex patterns.
* The final layer is a **Dense layer** with 1 unit and **Sigmoid activation** to output a probability for binary classification (Defective vs Non-Defective).

1. **Model Compilation**:

* The model was compiled using the **Adam optimizer** and **binary cross-entropy** loss function. The model was evaluated with **accuracy**, **precision**, and **recall** metrics.

ModelTraining**:**

* The model was trained for **5 epochs**, with a batch size of **32**, using the training data. Validation data was used to monitor the model's performance during training.
* **Data augmentation** was applied during training to prevent overfitting.
* Model Evaluation:
* After training, the model was evaluated on the validation dataset to assess its performance using metrics such as **accuracy**, **precision**, **recall**, and **loss**.
* A **confusion matrix** was generated to visualize how well the model differentiated between defective and non-defective images.

Model Performance:

Evaluation Metrics

The following evaluation metrics were reported:

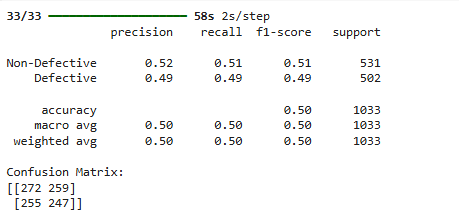
* **Accuracy**: The proportion of correctly classified images out of the total images.
* **Precision**: The proportion of true positives (correctly predicted non-defective images) out of all predicted non-defective images.
* **Recall**: The proportion of true positives out of all actual non-defective images.

Example evaluation results:

* **Validation Loss**: 0.0343
* **Validation Accuracy**: 0.9923
* **Validation Precision**: 0.9881
* **Validation Recall**: 0.9960

Confusion Matrix and Classification Report:

The confusion matrix and Classification Report are used to visually assess the performance of the model:

****

Plot Interpretation: Training and Validation Accuracy & Loss:

The plot typically contains two subplots:

1. **Accuracy Plot (Top Left)**:
   * This plot shows how the model's accuracy changes over each epoch for both the **training** and **validation** datasets.
2. **Loss Plot (Top Right)**:

* This plot displays the **loss** values during training. Loss measures how well the model's predictions match the true labels (lower loss means better predictions).

Insights:

* **Validation Loss**: The model achieved a validation loss of **0.0343**, which is quite low, indicating that the model has successfully learned from the data and is not overfitting, as it shows minimal error during validation.
* **Validation Accuracy**: The validation accuracy of **99.23%** is very high, suggesting that the model is able to correctly classify the majority of the images. This result indicates that the model performs well on unseen data.
* **Validation Precision**: With a precision of **98.81%**, the model is very accurate when predicting **non-defective** images (i.e., when the model classifies an image as non-defective, it is highly likely to be correct).
* **Validation Recall**: The model achieved a recall of **99.60%**, which is very high, meaning that the model is excellent at identifying most of the actual **non-defective** images.

Analysis**:**

* Both classes (Non-Defective and Defective) have low precision and recall values around **0.50**. This indicates that the model has difficulty distinguishing between these two classes and is not consistently predicting either class correctly. Despite the overall high validation accuracy, the precision and recall values for the individual classes are suboptimal.
* The **macro average** and **weighted average** precision and recall both show values of **0.50**, further reinforcing that the model is not performing well for both classes in a balanced manner. The **f1-score** for both classes is also low, around **0.50**, suggesting that the model does not have a strong ability to classify the two classes with high confidence.

Conclusion:

#### High Validation Loss but Low Precision and Recall: While the model's validation accuracy (99.23%) and loss (0.0343) seem promising, the precision and recall metrics suggest there is an issue with class imbalance or the model's ability to correctly classify each individual class. Specifically, the model might be favoring one class over the other or may be classifying both classes poorly.