Department of Computer Engineering
Bu-Ali Sina University

Computer Vision Course

Presented by Professor Hassan Khotanlou

Final Project

By:

Mahdi Jamebozorg

9912358013

Mohsen Hami

9912358016

Project Github Link

Fall 2023

Contents

ABSTRACT

Image enhancement is a fundamental task in the field of digital image processing, aiming to improve the visual quality of images for various applications, such as computer vision, medical imaging, and remote sensing.

In this project, we aim to design and implement a noise detection and deduction framework that leverages advanced image processing algorithms and deep learning techniques. By exploring different types of noise, including Gaussian noise, salt-and-pepper noise, and periodic, we seek to develop a comprehensive solution that can effectively identify and remove these artifacts from images. Through the integration of noise classification models and denoising algorithms, our framework aims to enhance the robustness and performance of Computer Vision systems in handling noisy image data.

Through the exploration of various noise types and the development of innovative solutions, we aim to contribute to the advancement of image denoising techniques and empower individuals to address noise-related challenges in real-world application

1. Phase 1: Binary Classification

Here, we want to design a simple but accurate classifier to apply to noisy datasets.

We have used different deep neural network architectures with transfer learning techniques to reach the best accuracy and generalization on the dataset.

As you see, in the first phase we have two Defected and Noisy classes for casting products and they have noise and this noise will affect the models's prediction, and the result of such a model is biased. In this phase we have different sub phase as below :

Preprocessing Phase:

after importing required packages from Tensorflow, Matplotlib, etc. we made 3 different directories in Colab to split the dataset into Train, Test, and Validation sets in relatively 70%, 15%, and 15% portions of the whole dataset.

Train, Test split is an inseparable part of ML projects because we need to train a model on the training set and the model's performance will be assessed on unseen test data before deploying the model in the real world.

So we randomly choose images to forge each set and finally, we have:

1063 Train images, 85 Test images, 85 Val images

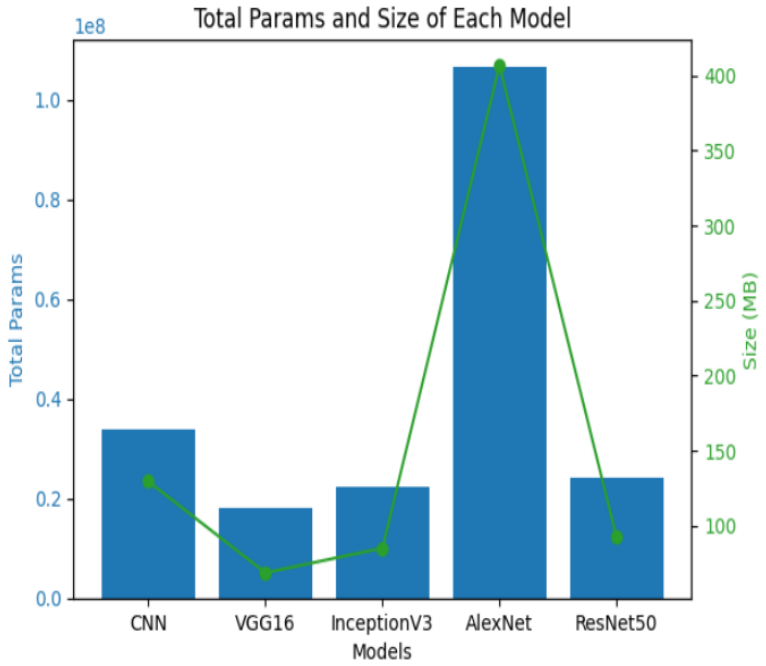
Also because the Deep model is generally considered as "data thirsty" we need to apply data augmentation techniques to increase dataset instances. In this process, we apply different operations on images like rescale image (which converts all pixels into 0-1),shear_range, zoom_range, and horizontal flipping.

We have applied these only on the Train dataset, but we need to rescale all images in the Validation and test dataset otherwise DNN models can't predict well!

In this case, because our DNN models require an image with size 204 * 204 we converted all images using the resize_images function which uses Pillow in Python to resize all images.

<u>Note:</u> we notified that Colab sometimes automatically generates some hidden checkpoint folder which leads the model to assume we have 3 different classes, we recognized this folder before defining the model and deleted that.

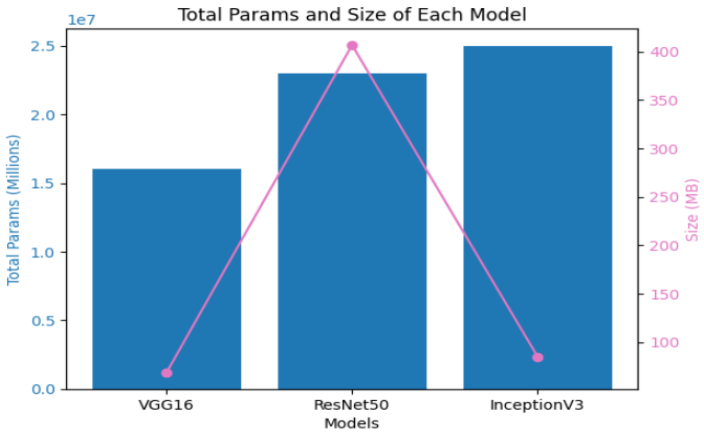
We have used the same assumption to compile and train the model but we want to depict some advantages and disadvantages of the model from different perspectives:



In a shallow comparison, you can see the lightest models are VGG16 and InceptionV3 you later will see the lightness of these two models is not real!

It's important to know that AlexNet and simple CNN don't have any pre-trained weights and don't use transfer learning but other models do.

Hence, for AlexNet and CNN, we need to define a more complex model with several hidden layers and neurons which leads to increased model size but we need an optimal complexity to detect underlying patterns of images.



We have to notice that pre-trained models have their own parameters and for better comparison, we need to regard them, in these two chart obviously you can see the magician of transfer learning!

Q1: How we can determine hyperparameters efficiently?
Since we had different projects before and based on inductive bias we can guess a range of different parameters for each model although we have used the Keras grid search tuner to find the best value of hyperparameters using a greedy approach.
For example in this code, we tested different optimizers and learning rates, and finding the best combination of these metrics is the final goal of this section.
optimizer = hp.Choice("model_name", values=["sgd", "adam"])
learning_rate = hp.Choice("learning_rate", values=[0.01, 0.1])

Q2: What's reason of using Keras callbacks in our solution?
Since a deep learning task is considered as a “universal approximator” which means we can achieve 100% accuracy but a poor generalization on test dataset, we have used different callbacks to prevent form model overfitting and increase the generalization of the model.
For example, “ ReduceLROnPlateau “ reduces the learning rate when the progress of a metric has stopped.
Some important callbacks are :
EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
You can find these callbacks in every fitting process and its precise definitions in the glossary part.

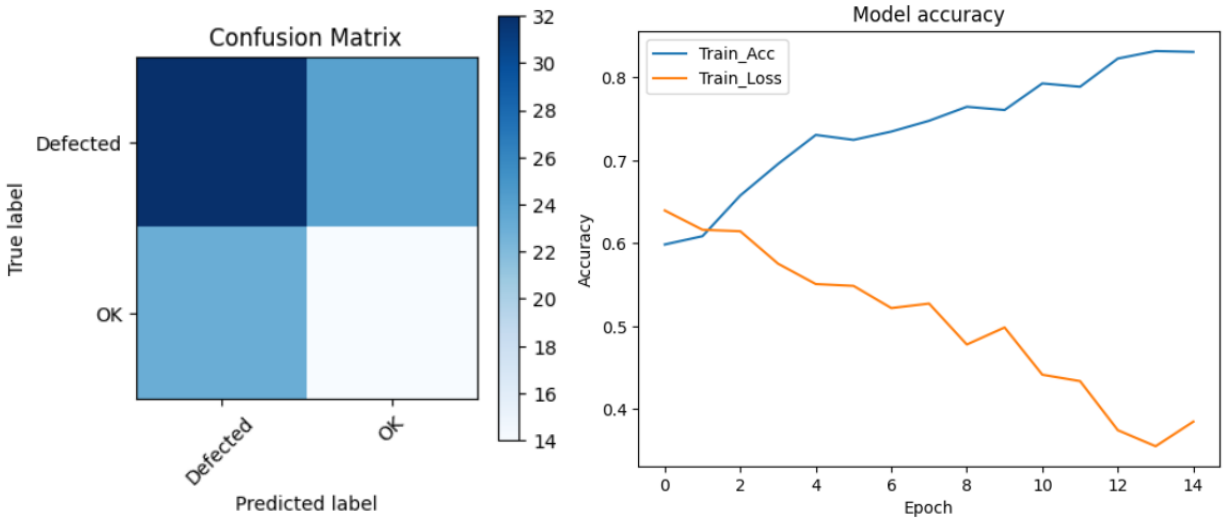
Results of first Phase:

We experimented different deep model with same assumption (15 epoch) to be comparable with each other based on different classification metrics:

Model	**Train_Acc**	**Test_Acc**	**Test_Loss**	**Precison**	**Recall**	**F1-score**	**Sen**	**Spe**	#rank
CNN	0.831	79.56	0.42	0.45	0.47	0.46	0.24	0.62	4
VGG16	0.945	94.62	0.098	0.50	0.49	0.50	0.37	0.57	1
ResNet50	0.64	68.23	0.61	0.54	0.56	0.54	0.27	0.75	5
AlexNet	0.82	81.17	0.40	0.47	0.44	0.44	0.45	0.42	3
InceptionV3	0.916	84.70	0.37	0.56	0.54	0.55	0.54	0.53	2

Dataset just trained in 15 epochs and in a noisy dataset! We will use from top two models for future prediction. (VGG16 and InceptionV3)

Charts just show for the best model (VGG16) although you can access other charts in the GitHub repository.

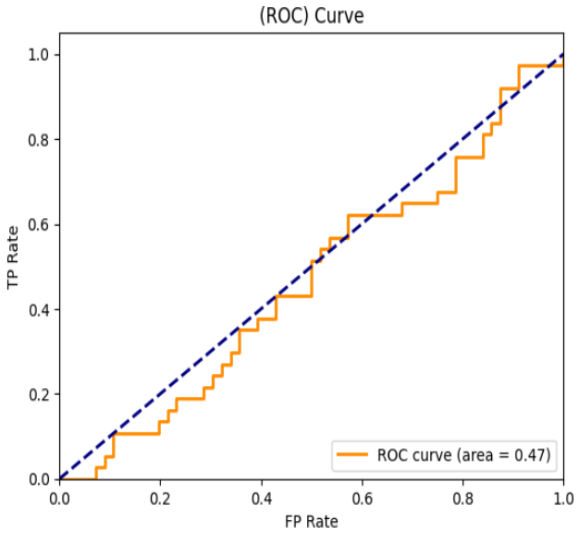


```
[INFO] classification report:
              precision    recall  f1-score   support

    Defected       0.58      0.57      0.58        56
          OK       0.37      0.38      0.37        37

    accuracy                           0.49        93
   macro avg       0.48      0.47      0.47        93
weighted avg       0.50      0.49      0.50        93

Sensitivity: 0.3783783783783784
Specificity: 0.5714285714285714
```



Based on shows confusion matrix we can deduct our model did not perform well in classifying noisy images well.

So in the third phase, we will apply denoise on this phase and will compare results with that.

2. Phase 2: Denoising Framework

At the sconce phase, we aim for noise detection which is a classification task then based on each type of noise we write a model for denoising in the spatial and frequency domain.

Q3 : Why frequency domain at all ?!

One of the key reasons for utilizing the frequency domain in image classification tasks is its ability to separate signal (desired information) from noise (unwanted artifacts) and enabling efficient image processing operations. In many real-world scenarios, images are corrupted by various types of noise, which can obscure important visual features and hinder accurate classification. By analyzing the frequency content of images, we can identify and isolate noise components that may be distinct in the frequency domain, enabling more effective noise reduction and image enhancement.

Preprocessing Phase for noise detection:

We have split the dataset into 3 different train, test, and validation parts to train an accurate classifier to detect types of noise which can be “salt and pepper”, “periodic” or “Gaussian”.

After that, we need to resize all images into 224*224 size to inject to the models! Also, we need to convert images into frequency domain so we have used “no.fft.fft2()” function for this work.

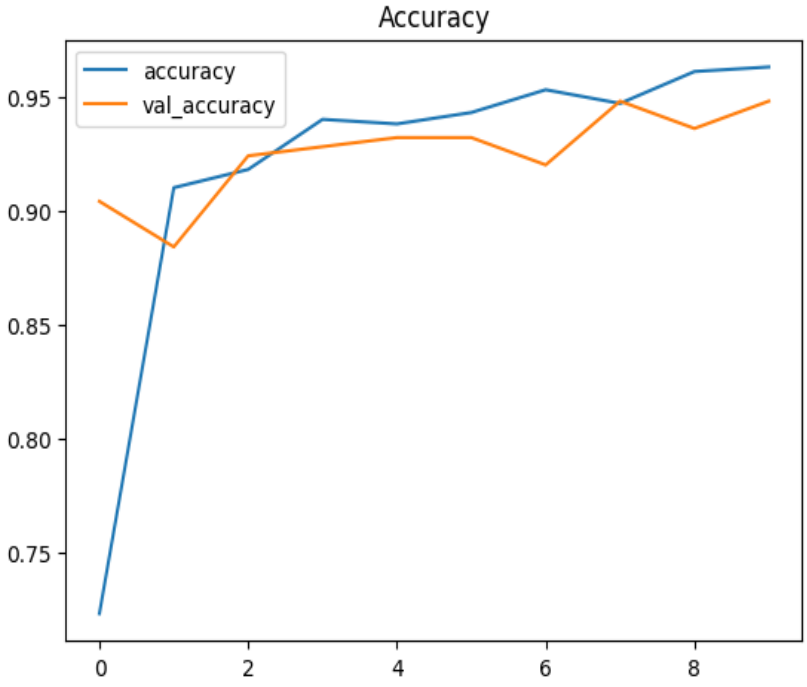
Notice: Because our algorithms need to shift images we have to apply fftshift() to converted images. (it’s also known as shift origin)

Here for the noise classification task, we made three different folders for each class and for the denoising task we want to detect the type of noise and define a special model for removing each noise.

A. Frequency domain noise classification :

After splitting the dataset into 3 different sets, we have applied the "Convert_to_MS()" function which is used to make the magnitude spectrum of the image so we can apply frequency domain operators on it easily.

We have used a pre-trained VGG16 model with "imagenet" weights to build a robust classifier, finally after six epochs the model reached Test Accuracy: %99.9, which is considered very good accuracy.



VGG16 model

We have tested the different model in the spatial domain but because of their poor performance we didn't continue then, although you can find the Github repository.

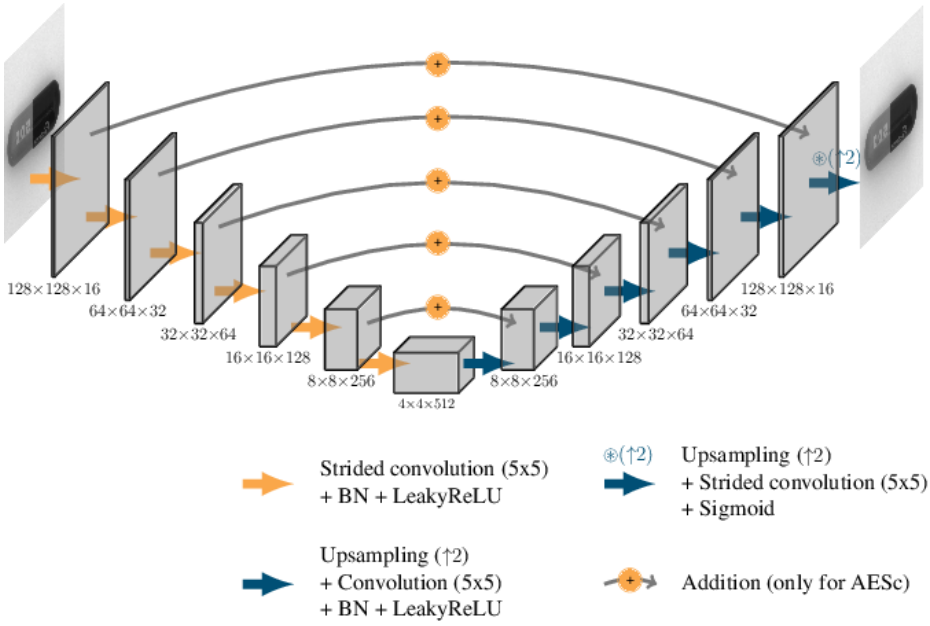
After detecting the type of noise we trained various models for removing the different types of noise effectively, the most difficult noise to remove was periodic and we used Autoencoder architecture in the spatial domain.

finally, after much research, we understand that an adaptive Gaussian notch filter in the frequency domain is a better approach for removing periodic noise but we didn't have enough time to implement it. (article link)

After many experiments, we found an optimal and fast method to remove noise for each type of noise.
For salt and pepper noise we suggested a median filter with kernel size=3, we experimented with different combinations of kernel size in different numbers and we used 3 sequential **median filters with kernel size=** 3, using another kernel size **(like 5 or 7)** couldn't reconstruct the image very well!

Autoencoder network Architecture :

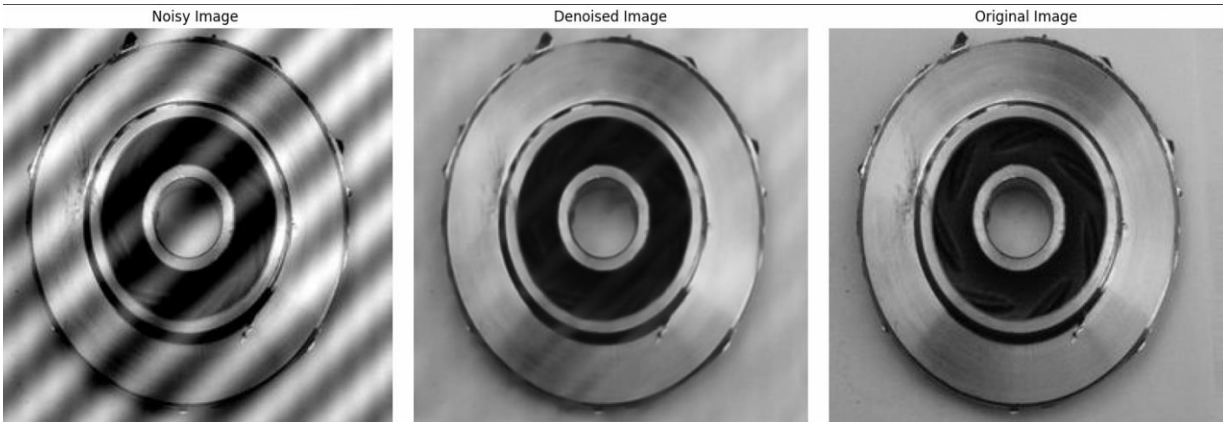
An Autoencoder architecture that incorporates convolutional layers in both the encoder and decoder parts of the network. The main advantage of autoencoders over the aforementioned algorithms is its ability to map more complex, nonlinear relationships between data.



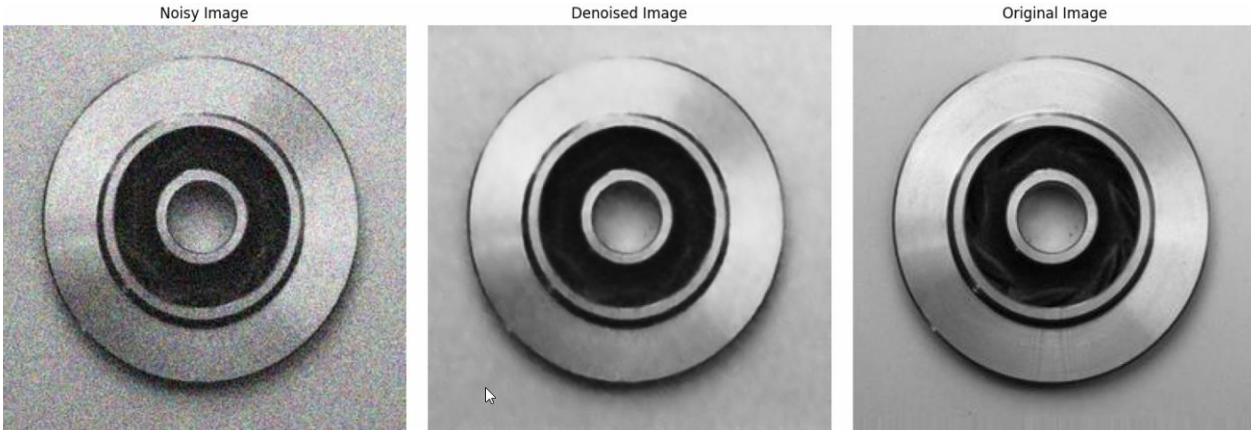
Q3: what is benefits of skip connections is Autoencoder ?

The skip connections improve the performance of the Autoencoder, the positions and number of these connections can be experimented with. The model can also be trained with different levels of noise factor for better generalization of results.

Based on our investigation of the dataset, periodic noise is the most difficult noise to remove, because the domain of image corruption is very wide we needed to use from Deep learning method for denoising so we have used Autoencoder architecture to remove periodic noise, but strangely we noticed it can remove Gaussian noise well!



Periodic noise reduction using Deep Autoencoder : PSNR = 22.57



Gaussian noise reduction using Deep Autoencoder : PSNR = 24.52

So we have used from lightweight Autoencoder denoiser and we trained it on 80% of dataset2 Images we have used 10% for test and 10% for validation.

We have used different callbacks to achieve a robust model with high generalization on test data.

After training all of the models for noise type detection and noise reduction we have built a pipeline to perform on 25% of the whole noisy images from dataset2, and we have calculated all of the metrics to compare different results:

```
---> Medain filter result  <----
-> Avg SSIM for Salt noise  : 87.969 ± 1.507 and MAX is : 91.389
-> Avg PSNR for Salt noise : 34.658 ± 0.485 and MAX is : 36.06
-> Avg LPIPS for Salt noise  : 5.84 ± 1.269 and MAX is : 10.853
-> These results are valid for 100 instances ! for random  25% percent of Dataset2!
```

```
---> Autoencoder result for gaussian noise  <----
-> Avg SSIM for gaussian noise  : 64.047 ± 12.885 and MAX is : 88.32
-> Avg PSNR for gaussian noise : 23.73 ± 1.734 and MAX is : 27.756
-> Avg LPIPS for gaussian noise  : 18.984 ± 10.682 and MAX is : 46.763
-> These results are valid for 107 instances ! for random  25% percent of Dataset2!
```

```
---> Autoencoder result for Periodic noise  <----
-> Avg SSIM for Periodic noise  : 81.696 ± 4.753 and MAX is : 87.542
-> Avg PSNR for Periodic noise : 22.831 ± 3.047 and MAX is : 26.824
-> Avg LPIPS for Periodic noise  : 5.124 ± 2.143 and MAX is : 13.031
-> These results are valid for 96 instances ! for random  25% percent of Dataset2!
```

As you can see,the results have been verified by visual improvement metric and it means we don't pay attention just to some numbers!

Different challenges with this project:

The noise classifier model performed well on my colleague's system but I wanted to merge its code to my denoiser code to build an end-to-end pipeline, it did not work well and it couldn't predict Salt and pepper noise at all, so as I previously mentioned for image resizing in Phase 1 I have used Pillow package with “Near” filter to resize image and this was the failure point of the model, so I changed that code with Keras built-in function for image resizing and our model’s performance got well.

3. Phase 3: Final Lap

After training all models based on ML model development principles, we have saved all of the models and their weights for google colab to Google Drive. Our pipeline now is ready to get a noisy dataset as input and report improvement of classification task after noise removal!

We have improved the accuracy of the VGG16 model **from 94.6% to 97%** on test data .

```
3/3 [==============================] - 1s 132ms/step - loss: 0.0985
Test Accuracy: 94.62365508079529
3/3 [==============================] - 1s 125ms/step

[INFO] Confusion Matrix:
[[32 24]
 [23 14]]

[INFO] classification report:
              precision    recall  f1-score   support

    Defected       0.58      0.57      0.58        56
          OK       0.37      0.38      0.37        37

    accuracy                           0.49        93
   macro avg       0.48      0.47      0.47        93
weighted avg       0.50      0.49      0.50        93

Sensitivity: 0.3783783783783784
Specificity: 0.5714285714285714


4/4 [==============================] - 3s 724ms/step - loss: 0.0969
Test Accuracy: 97.02970385551453
4/4 [==============================] - 1s 126ms/step

[INFO] Confusion Matrix:
[[40 23]
 [26 12]]

[INFO] classification report:
              precision    recall  f1-score   support

    Defected       0.61      0.63      0.62        63
          OK       0.34      0.32      0.33        38

    accuracy                           0.51       101
   macro avg       0.47      0.48      0.47       101
weighted avg       0.51      0.51      0.51       101

Sensitivity: 0.3157894736842105
Specificity: 0.6349206349206349
```

We have improved the accuracy of the InceptionV3 model **from 84.7% to 90%** on test data.

```
4/4 [==============================] - 4s 652ms/step - loss: 0.2251
Test Accuracy: 90.09901285171509
4/4 [==============================] - 3s 67ms/step

[INFO] Confusion Matrix:
[[36 27]
 [19 19]]

[INFO] classification report:
              precision    recall  f1-score   support

    Defected       0.65      0.57      0.61        63
          OK       0.41      0.50      0.45        38

    accuracy                           0.54       101
   macro avg       0.53      0.54      0.53       101
weighted avg       0.56      0.54      0.55       101

Sensitivity: 0.5
Specificity: 0.5714285714285714
3/3 [==============================] - 4s 1s/step - loss: 0.3794
Test Accuracy: 84.70588326454163
3/3 [==============================] - 2s 77ms/step

[INFO] Confusion Matrix:
[[28 24]
 [15 18]]

[INFO] classification report:
              precision    recall  f1-score   support

    Defected       0.65      0.54      0.59        52
          OK       0.43      0.55      0.48        33

    accuracy                           0.54        85
   macro avg       0.54      0.54      0.53        85
weighted avg       0.56      0.54      0.55        85

Sensitivity: 0.5454545454545454
Specificity: 0.5384615384615384
```

Conclusion :

Noise in images can significantly impact the performance of image classification tasks by introducing unwanted artifacts that can obscure important visual features and hinder accurate classification. The presence of noise can distort image content, leading to misinterpretation of key patterns and features by machine learning models.

By effectively removing noise from images through techniques such as filtering in the frequency domain, the accuracy and other metrics of classification tasks can be significantly improved. For instance, after noise removal, the accuracy of the InceptionV3 model increased from 84% to 90%, while the VGG16 model saw an improvement from 94% to 97%.

The reduction of noise in images allows machine learning models to focus on relevant visual information, leading to more accurate feature extraction and classification. This improvement in accuracy demonstrates the importance of noise removal in enhancing the performance of image classification models and underscores the impact of noise on the overall effectiveness of Computer Vision systems.

Glossary of terms :

Term	Acronym	Definition
classifier		Is an algorithm that automatically orders or categorizes data into one or more of a set of “classes.
deep neural network	DNN	A class of machine learning algorithms similar to the artificial neural network and aims to mimic the information processing of the brain.
transfer learning		Reuse of a pre-trained model on a new problem which a machine exploits the knowledge gained from a previous task to improve generalization about another.
data augmentation		The process of artificially generating new data from existing data, primarily to train new Deep learning (DL) models.
Autoencoder		Autoencoder is an unsupervised artificial neural network that learns how to efficiently compress and encode data then learns how to reconstruct the data back from the reduced encoded representation to a representation that is as close to the original input as possible.
inductive bias		The **inductive bias** (also known as **learning bias**) of a learning algorithm is the set of assumptions that the learner uses to predict outputs of given inputs that it has not encountered.

References :

1. **Periodic Noise Reduction Methods for High-Resolution Infrared Line Scanning Images[link]**
2. **What are Autoencoders? Applications and Use Cases [link]**
3. **Adaptive Periodic Noise Reduction in Digital Images Using Fuzzy Transform [link]**
4. **Gaussian Noise Removal in an Image using Fast Guided Filter and its Method Noise Thresholding in Medical Healthcare Application [link]**
5. **Salt and Pepper Noise Removal Method Based on the Edge-Adaptive Total Variation Model [link]**
6. **Different article form google scholar [link]**
7. **And Lesson slides …**