



## **Weekly Report**

# **Comparative Analysis of Image Classification Models for Efficient and Accurate Classification across Noisy Vegetable Images**

Student: Yusif Mukhtarov

**Date: 08/06/2023**

## Contents

Abstract .....	3
Road map .....	3
The problem definition .....	4
Application areas .....	4
Planning steps .....	5
Results of the project .....	6
Hyperparameter tuning results .....	6
Analysis of results.....	7
Github .....	12
Feature work .....	13

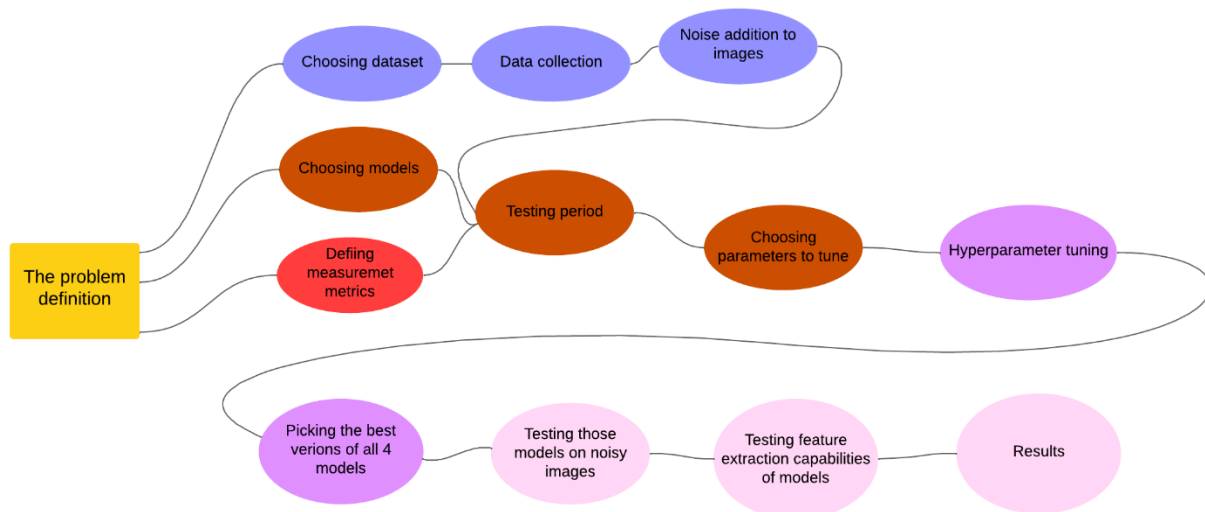
## Abstract

This research delves into the in-depth analysis of popular deep learning models, including AlexNet, VGG16, ResNet, and EfficientNet, applied to vegetable custom datasets. The study aims to explore and compare the performance of these models, considering a range of performance metrics to ensure an unbiased evaluation. The chosen vegetable datasets are carefully preprocessed, encompassing required image conditions and added noise to simulate real-world scenarios.

To evaluate the models thoroughly, classification accuracy serves as the primary performance metric. Additionally, precision, recall, and F1-Score are incorporated to provide a comprehensive understanding of the models' strengths and weaknesses in handling false positives, false negatives, and overall balanced performance. Furthermore, the research records the time taken for training and prediction phases, along with the number of parameters, to assess the models' efficiency.

The implications of this research span across various industries. In agriculture, these findings can lead to smarter farming practices, crop health monitoring, and pest detection. The food processing industry can benefit from improved quality control, as the models can assist in sorting vegetables based on visual attributes. Moreover, retail and supply chain management can optimize inventory processes using accurate vegetable identification.

## Road map



## The problem definition

In the field of deep learning for recognizing images, we usually train models using very good and clear datasets without any errors. However, in real-life situations, the pictures we get might not be perfect. They could have problems like bad weather, issues with the camera, or other factors, making the images noisy and of lower quality. This difference between the training data and real-world images can greatly affect how well the deep learning models work in real situations.

The problem we want to solve is that there isn't enough research and study on how these popular deep learning models, such as AlexNet, VGG16, ResNet, and EfficientNet, perform when they have to deal with noisy and imperfect images. We need to look into how these models behave and adapt in these more challenging situations. By doing this research and studying how well these models work with realistic images, we can better understand how they can be used in practical applications and real-life scenarios.

## Application areas

The research has wide-ranging impacts across various industries. In agriculture, it can improve farming practices, monitor crop health, and detect pests. Food processing can benefit from better quality control by sorting vegetables accurately. Retail and supply chain management can optimize inventory using precise vegetable identification.

Apart from agriculture and food, this research has potential applications in environmental monitoring, healthcare, nutrition, and smart kitchen appliances. It can aid in preserving biodiversity and promoting healthier eating habits. Smart kitchen appliances can use these models to improve user experiences and cooking processes.

Moreover, the research goes beyond vegetable recognition and can be applied to other image types, like noisy faces and cars. By using similar methods, it can guide deep learning models for various tasks like facial recognition, object detection, autonomous vehicles, surveillance, and medical imaging. The insights gained here serve as a helpful resource for researchers and practitioners dealing with different image challenges in AI-driven solutions.

## Planning steps

At the beginning of the project, a plan was laid out to test six different models for image classification: LesNet, MobileNet, AlexNet, VGG16, EfficientNet, and ResNet. The dataset selection was intended to be from popular sources, and various augmentation techniques like flipping, rotation, occlusion, and noise addition were planned. However, after careful discussions with supervisors, it was realized that attempting all this within the 11-month time frame would be too ambitious.

Consequently, the project scope was narrowed down, and it was decided to focus on testing only four models: AlexNet, VGG16, EfficientNet, and ResNet. Moreover, the decision was made to evaluate these models using only noisy images.

Initially, the intention was to use well-known datasets such as ImageNet, Cifar-10, or Cifar-100. However, following discussions with supervisors, it was deemed better to opt for a random dataset. The reason behind this was that some of the previously listed models might have been trained on the mentioned popular datasets and using them could lead to ambiguity in the results. Therefore, the Vegetable dataset, considered neutral, was chosen for the image classification tasks. Consequently, the project's name was modified to "Comparative Analysis of Image Classification Models for Efficient and Accurate Classification across Noisy Vegetable Images ". To degrade the image quality, it was chosen to add Gaussian noise at different levels: 10%, 30%, 50%, 70%, and 100%. After introducing this noise, it was intended to evaluate the performance of the models on each of these percentage levels to understand how well they can handle images with varying degrees of noise.

In the early stages of the project, there was a consideration of training all the models from scratch. However, due to computational limitations, it was advised by the supervisor to use pretrained models instead. This approach would save computational resources and still allow for meaningful comparisons. For hyperparameter tuning, the focus is planned on optimizing learning rates, schedulers, and momentum values for the models. These parameters play a crucial role in the performance of deep learning models and tuning them would help achieve better results in our image classification tasks.

# Results of the project

## Hyperparameter tuning results

As was discussed in the previous weekly report, I conducted hyperparameter tuning last week to figure out the best versions of all models. I have tested 3 parameters and 2 values for each parameter. For each models the results for 9 combinations were obtained, however, just for simplicity only best 3 for each will be shown:

model	n	lr	m	s	g	a
alexnet	10	0.005	0	3	0.5	0.990
alexnet	10	0.005	0.1	3	0.1	0.989
alexnet	10	0.005	0	3	0.1	0.988

model	n	lr	m	s	g	a
vgg16	10	0.005	0.1	3	0.1	0.986
vgg16	10	0.005	0	3	0.5	0.984
vgg16	10	0.005	0	3	0.1	0.982

model	n	lr	m	s	g	a
effNet	10	0.005	0.1	3	0.1	0.988
effNet	10	0.005	0	3	0.1	0.987
effNet	10	0.005	0.1	4	0.1	0.987

model	n	lr	m	s	g	a
resnet	10	0.005	0.1	4	0.1	0.995
resnet	10	0.005	0.1	3	0.1	0.994
resnet	10	0.005	0.1	4	0.5	0.994

It is noteworthy to highlight that the only reason why the values of accuracies don't differ much lies in the percentage of noise, and but the ranking of models remained unchanged, ensuring that the best versions of the models were still selected. The main reason for opting for 10% noise for the validation dataset is to showcase higher accuracies in presentation. But now I believe I should pick 20% of noise to demonstrate a more pronounced variation in accuracy values.

## Analysis of results

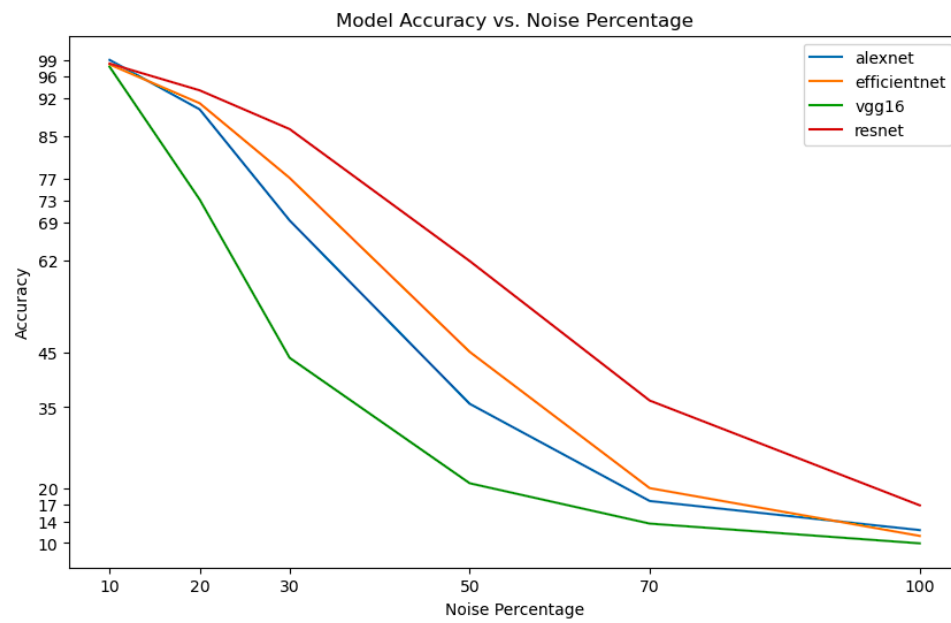
After picking the results of the project it was finally time to test them on the datasets with 6 different noise levels: 10, 20, 30, 50, 70 and 100%. In tables below the results for all 7 datasets are represented:

model	noise_percentage	accuracy	precision	f	time	number_parameters
alexnet	10	98.97	0.99	0.99	0.065	57065295
resnet	10	98.3	0.98	0.98	0.721	42530895
efficientnet	10	98.13	0.98	0.98	0.583	4026763
vgg16	10	97.7	0.98	0.97	12.194	134321999
model	noise_percentage	accuracy	precision	f	time	number_parameters
resnet	20	93.37	0.94	0.93	0.667	42530895
efficientnet	20	91	0.93	0.91	0.497	4026763
alexnet	20	89.87	0.92	0.89	0.067	57065295
vgg16	20	73.27	0.82	0.71	12.227	134321999

model	noise_percentage	accuracy	precision	f	time	number_parameters
resnet	30	86.23	0.91	0.86	0.689	42530895
efficientnet	30	77.23	0.85	0.77	0.489	4026763
alexnet	30	69.4	0.83	0.67	0.069	57065295
vgg16	30	44.1	0.65	0.39	12.235	134321999
model	noise_percentage	accuracy	precision	f	time	number_parameters
resnet	50	61.93	0.8	0.62	0.681	42530895
efficientnet	50	45.2	0.73	0.46	0.5	4026763
alexnet	50	35.63	0.69	0.32	0.065	57065295
vgg16	50	21	0.37	0.12	12.226	134321999

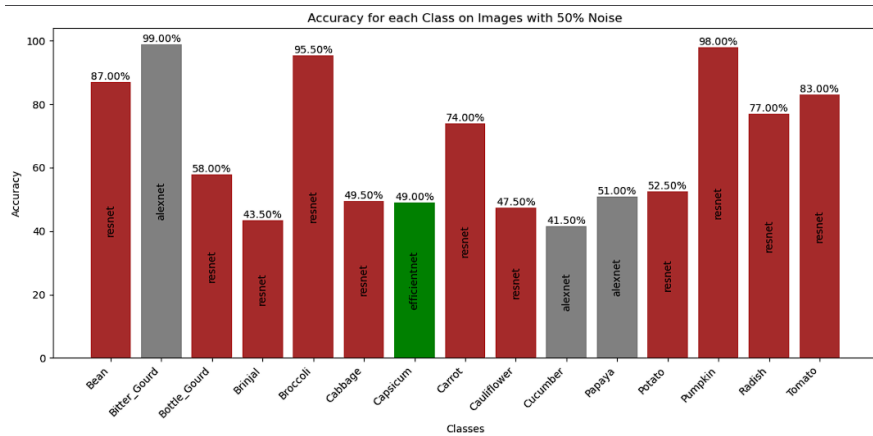
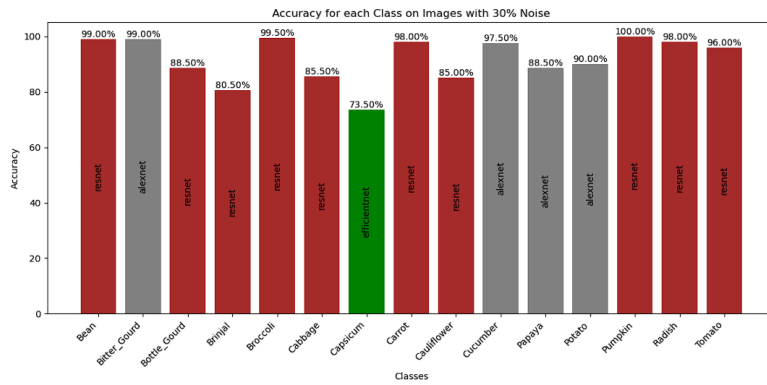
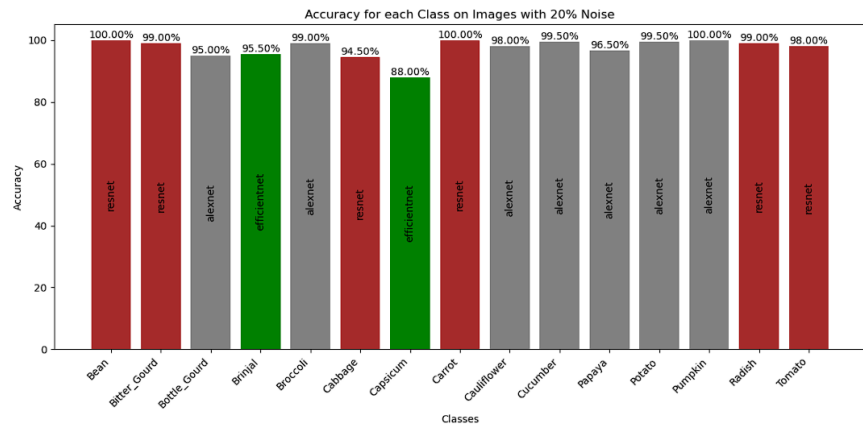
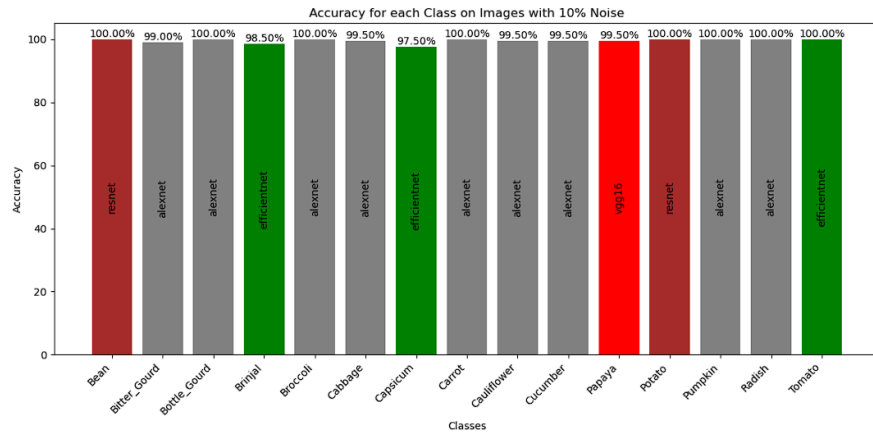
model	noise_percentage	accuracy	precision	f	time	number_parameters
resnet	70	36.23	0.6	0.31	0.703	42530895
efficientnet	70	20.1	0.61	0.18	0.489	4026763
alexnet	70	17.73	0.5	0.13	0.067	57065295
vgg16	70	13.57	0.13	0.06	12.186	134321999
model	noise_percentage	accuracy	precision	f	time	number_parameters
resnet	100	16.93	0.34	0.12	0.677	42530895
alexnet	100	12.37	0.27	0.07	0.068	57065295
efficientnet	100	11.3	0.23	0.05	0.508	4026763
vgg16	100	9.93	0.09	0.04	12.197	134321999

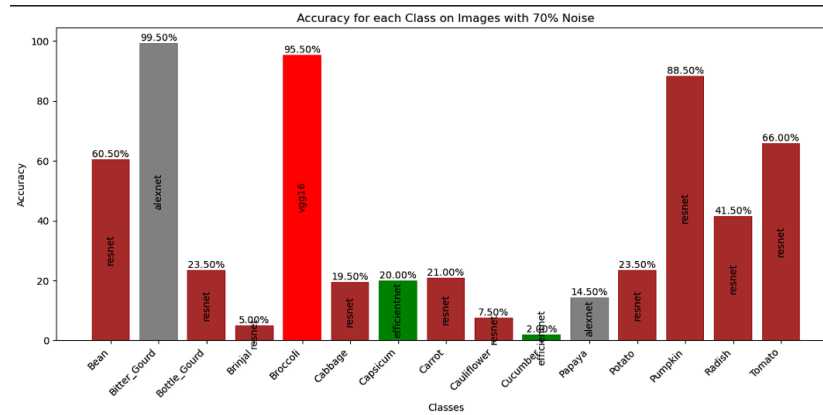
The superiority of ResNet over other models becomes evident from the tables. Furthermore, the graph below provides solid confirmation of this observation, as it clearly shows that as the noise percentage increases, the gap between ResNet and other models becomes increasingly pronounced:



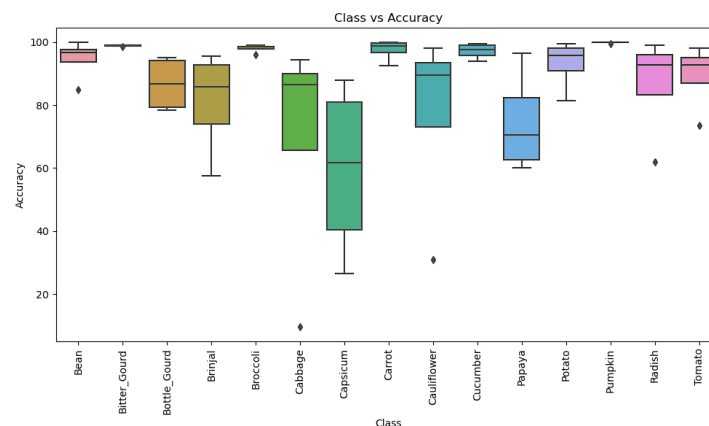
After finding out the best model it was decided to make deeper analysis to study the performance of models on different classes.





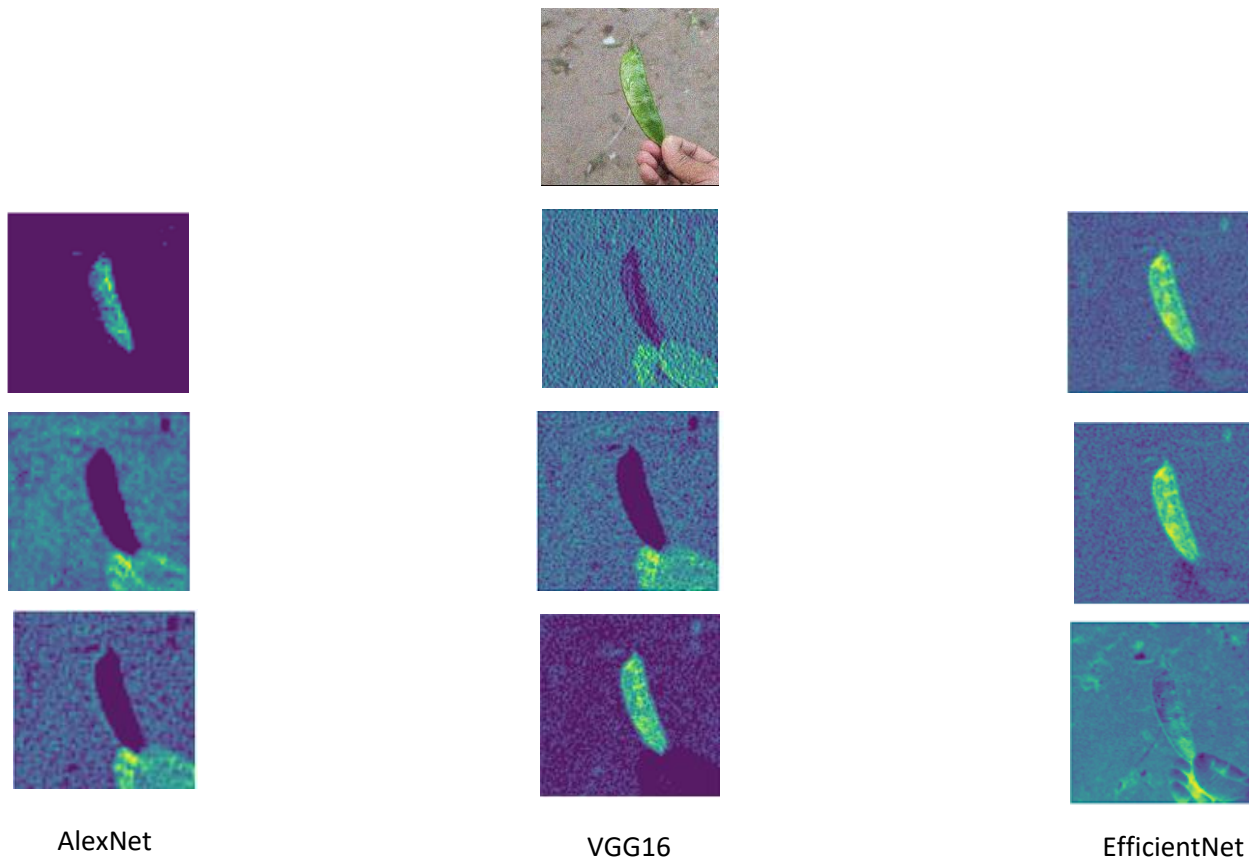


In this analysis, ResNet's clear superiority over other models is once again evident. As the percentage of noise increases, ResNet consistently outperforms the competition in classifying a greater number of classes more accurately than other models. However, there are some specific classes that specific models are classifying better than other models. For example, Alexnet is the best models for Bitter Gourd classification almost in all datasets, same works for efficientNet classifying Capsicum, resnet classifying Bean, Carrot, Radish and Tomato. To assess which classes were easily classified and which brought difficulties to models to recognize them the figure below has been illustrated:

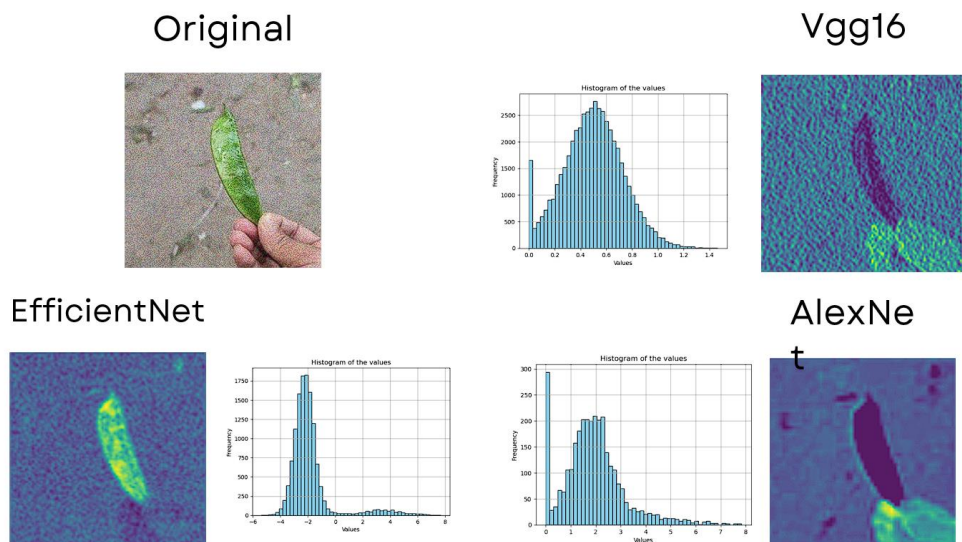


It is clear from the figure that Bitter Gourd, Broccoli and Pumpkin are classes that were easily classified, however, models experienced serious challenges to classify Capsicum, Papaya, and Cabbage.

After analyzing the results of models, it was decided to analyze feature extraction capabilities of models, and to demonstrate them following picture was illustrated:



More deep analysis showed that some models with a narrow distribution of feature extracted values showed better results:









## Github

1. adding\_noise.ipynb – I have preprocessed, added noise to images and saved them in data folder.
2. dataloader.py – a main loader function for loading images
3. feature\_extraction.py, statistical\_analyses.py – for statistical and feature extraction analyses
4. initial\_preprocessing.py – image augmentation tests
5. results\_in\_kaggle.ipynb – in this file I have tested the models, plotted charts, illustrated tables and examined feature extraction capabilities of models
6. hyperparameter\_tuning – a folder where I have placed 4 files for each of 4 models' hyperparameter tuning.
7. Drafts – a folder with test codes, drafts, examined features, etc.
8. Data – a folder for input data and **output results**. Since my data is image dataset, I will put a link to my dataset in google drive ([https://drive.google.com/drive/folders/1j67w7Q9tnrejNltWP92Sg\\_722pyU0fMG?usp=sharing](https://drive.google.com/drive/folders/1j67w7Q9tnrejNltWP92Sg_722pyU0fMG?usp=sharing) ), moreover, I will upload there the results of the research as well:

8.1 data/results/modelName/**data**/modelName\_data\_noisePercentage.csv in this file I have saved the results in the table form with columns – Target, Predicted, TF (shows 1 of Target == Predicted, else 0), percentage of noise that image has.

t	p	tf	poise_percentage
3	3	1	10







data > results > alexnet > data

Name	Date modified	Type	Size
 alexnet_data_10.csv	8/1/2023 4:28 PM	Microsoft Excel Co...	29 KB
 alexnet_data_20.csv	8/1/2023 4:28 PM	Microsoft Excel Co...	29 KB
 alexnet_data_30.csv	8/1/2023 4:28 PM	Microsoft Excel Co...	29 KB
 alexnet_data_50.csv	8/1/2023 4:28 PM	Microsoft Excel Co...	29 KB
 alexnet_data_70.csv	8/1/2023 4:28 PM	Microsoft Excel Co...	28 KB
 alexnet_data_100.csv	8/1/2023 4:28 PM	Microsoft Excel Co...	31 KB

8.2 Date/results/modelName/**res**/modelName\_res\_noisePercentage.csv in this file I have saved the results of models tested on 6 test datasets (dataset with 10, 20 ,30, 50, 70 and 100% noise percentage). This file is also represented in a table format with following columns: Class, Accuracy, Recall, Precision, and F score for each class as well as for the whole dataset.

Class	Accuracy	Recall	Precision	F score
Bean	99	0.99	0.99	0.99

data > results > alexnet > res

Name	Date modified	Type	Size
 alexnet_res_10.csv	8/1/2023 4:28 PM	Microsoft Excel Co...	1 KB
 alexnet_res_20.csv	8/1/2023 4:28 PM	Microsoft Excel Co...	1 KB
 alexnet_res_30.csv	8/1/2023 4:28 PM	Microsoft Excel Co...	1 KB
 alexnet_res_50.csv	8/1/2023 4:28 PM	Microsoft Excel Co...	1 KB
 alexnet_res_70.csv	8/1/2023 4:28 PM	Microsoft Excel Co...	1 KB
 alexnet_res_100.csv	8/1/2023 4:28 PM	Microsoft Excel Co...	1 KB

## Feature work

If I was asked how I would do this project from the beginning given that I had enough computational power and human resources, I would describe it in 3 main steps:

1. Finding perfect naturally generated noise images from different domains to simulate real case scenarios.
2. Building a customized neural network by hyperparameter tuning the number of layers, neurons in those layers, and other ml parameters.
3. Finding a threshold by using the system where we will denoise those images and then classify them by applying different machine learning models. From the results produced by those ml models we are picking the best accuracy and time required for prediction. Those 2 values for the best accuracy and time will be our threshold that we will try to pass it with our neural network that will classify noise images without being trained on noise images and without being denoised.

The whole idea behind the project is not just simply classifying noise images, the idea is the creation of a perfect model that will be able to classify new types of image datasets that it was not being trained on. The whole idea is creating ML model that is able to generalize and classify correctly new types of image datasets – in our case noisy images. I didn't have enough time, resources, and computational power to accomplish it, however, it can be achieved in future work. The main problem of the research will not be computational power or creating ML model, indeed the biggest problem of this research is finding naturally produced labeled large noisy image dataset from different domain. Different domains are key since it is very crucial to test the generalization abilities of the model, and its applicability in different domains.

Current small research project problem can be very helpful for future work by:

1. Using the structure of resnet as an example for new customized neural network, since this CNN showed excellent results in the testing stage.

2. In the provided material from the result part there are a lot of interesting patterns full of insights that can be useful in future work. For example, some of models showed extremely good results in classifying some classes even in dataset with 100% of noise.
3. I have tested several hyperparameters and its results can give clues in future work in hyperparameter tuning section.