Weekly Report

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

Student: Yusif Mukhtarov

Date: 31/07/2023

Contents

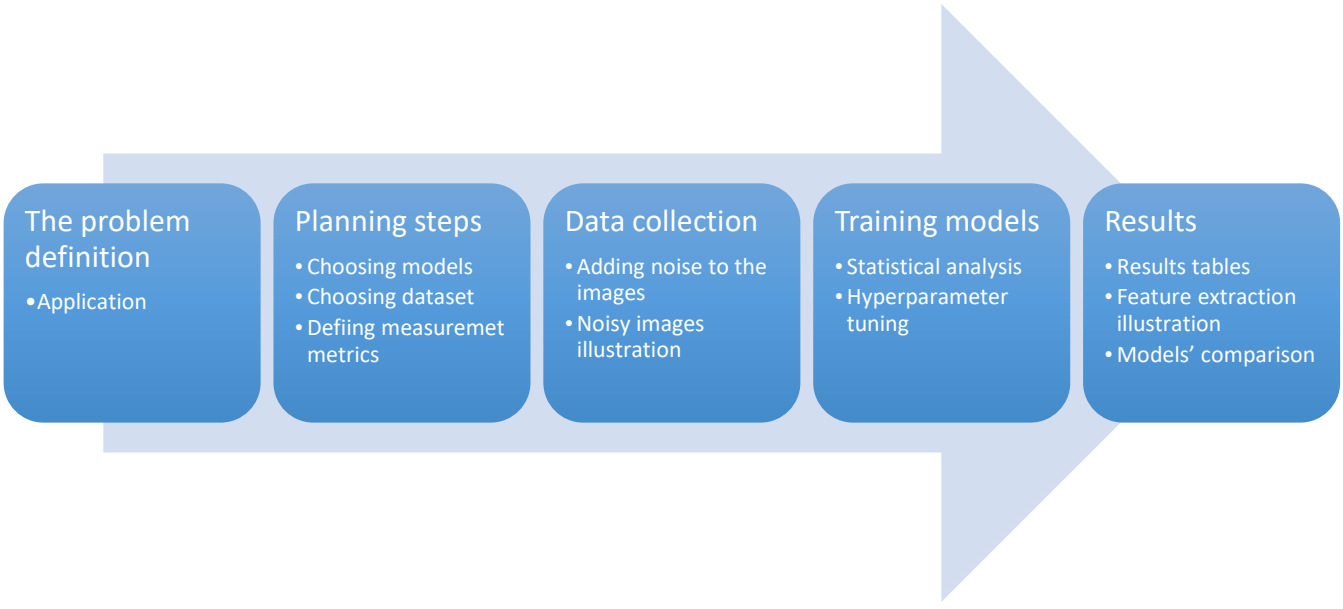
Abstract

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.

Weekly progress

As was highlighted in the last weekly progress report, the next crucial step in the research is hyperparameter tuning. I have chosen limited parameters to tune considering the computational power limit for Kaggle GPU, which is restricted to 30 hours per week. To make the most efficient use of this time, I decided to first test different parameter combinations to identify those that will not yield any significant results. I will then proceed to test the remaining combinations that show promise. Moreover, in my testing period I was using 15-20 epochs, but for hyperparameter tuning I have reduced it till 10 epochs to meet the restriction of Kaggle for GPU in 30 hours per a week. Here, I will describe the parameters that have been chosen to be tested:

1. Learning rate scheduler gamma, which is used to reduce the learning rate by a factor of gamma after a certain number of epochs. During my initial testing period, it appeared that a learning rate of 0.001 was too small, as it took too many epochs for the modules to reach good accuracies (>0.9). Consequently, I have decided to increase the learning rate to 0.005, and to test two different values for the learning rate scheduler gamma: 0.5 (reducing the learning rate twice) and 0.1 (reducing the learning rate ten times).
2. Learning rate scheduler step size, which indicated after how many steps the learning rate scheduler gamma will be reduced. During my initial testing period I noticed that models after 3rd -5th epoch cannot increase their accuracies, it seems that it is going around global minimum and cannot reach it. That is why I have decided to reduce the scheduler in the first case every 5 epoch, which means step size is 5, and in the next case every 3 epochs.
3. Momentum, which is a technique that helps accelerate the convergence of the optimization process and can help escape shallow local minima. During my initial testing period I noticed that tested 0.9 and 0.5, and those showed bad results. However, momentum 0.1 illustrated better results with the test accuracy (>0.9 on images with 10% noise), that is why I have decided to test 2 cases: without momentum, and with momentum (0, 0.1). 0.1 momentum means that previous velocity will have a relatively small influence, and most of the update will be guided by the current gradient. It is noteworthy to highlight that in most cases, the values for momentum are usually in the range between 0.8 and 0.99. However, for this specific research, considering the results obtained during the test period, I have decided to test a value of 0.1, and it yielded good accuracies for some models.

The hyperparameter tuning process illustrated in Figure 1. As was mentioned previously, 4 models are being tested in this research: Alexnet, EfficientNet, ResNet, and VGG16. Results of all models will be shown separately since there are a lot of information and charts obtained from it.

```
Epoch 10/10
----------

train Loss: 0.0019 Acc: 0.9993


val Loss: 0.0786 Acc: 0.9807

Training complete in 52m 7s
Best val Acc: 0.982333
model                                                      vgg16
num_epochs                                                    10
learning_rate                                              0.005
momentum                                                       0
step                                                           3
gamma                                                        0.1
accuracy                                                0.982333
history          [[0.9320666666666667, 0.9941333333333334, 0.99...
Name: 0, dtype: object
STARTING 2 ITERATION
Starting to test lr = 0.005, m = 0, s = 3, g = 0.5
Epoch 1/10
```

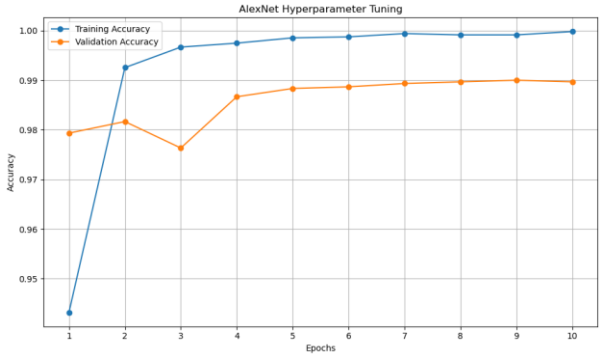
Figure 1. The hyperparameter tuning process, lr – learning rate, m – momentum, s – step size, g – scheduler gamma

Alexnet

Results of hyperparameter tuning:

	model	num_epochs	learning_rate	momentum	step	gamma	accuracy
1	alexnet	10	0.005	0.0	3	0.5	0.990000
4	alexnet	10	0.005	0.1	3	0.1	0.988667
0	alexnet	10	0.005	0.0	3	0.1	0.988333
2	alexnet	10	0.005	0.0	4	0.1	0.988000
3	alexnet	10	0.005	0.0	4	0.5	0.988000
5	alexnet	10	0.005	0.1	3	0.5	0.985667
7	alexnet	10	0.005	0.1	4	0.5	0.985667
6	alexnet	10	0.005	0.1	4	0.1	0.985333

The model that was trained with momentum, step size, and gamma equal to 0, 3, 0.5 respectfully was found to be the best. In the graph below the training procedure was illustrated:



After hyperparameter tuning it was tested on data set with 0, 10, 20, and 30% noise. In the table below results for testing data with 10% noise is illustrated:

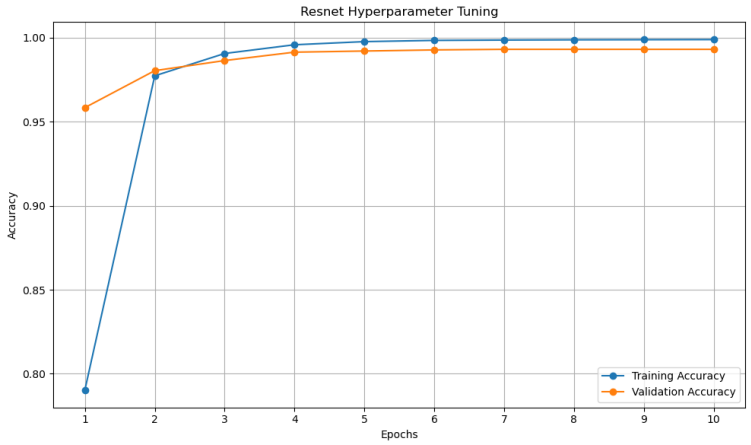
	Class	Accuracy	Recall	Precision	F_score
0	Bean	99.00	0.99	0.99	0.99
1	Bitter_Gourd	99.00	0.99	1.00	0.99
2	Bottle_Gourd	100.00	1.00	0.99	0.99
3	Brinjal	97.50	0.97	0.99	0.98
4	Broccoli	100.00	1.00	1.00	1.00
5	Cabbage	99.50	0.99	0.99	0.99
6	Capsicum	93.50	0.94	1.00	0.97
7	Carrot	100.00	1.00	1.00	1.00
8	Cauliflower	99.50	0.99	0.99	0.99
9	Cucumber	99.50	0.99	0.98	0.98
10	Papaya	99.00	0.99	0.95	0.97
11	Potato	100.00	1.00	0.98	0.99
12	Pumpkin	100.00	1.00	0.98	0.99
13	Radish	100.00	1.00	1.00	1.00
14	Tomato	98.00	0.98	1.00	0.99
15	Total	98.97	0.99	0.99	0.99

Resnet

Results of hyperparameter tuning:

	model	num_epochs	learning_rate	momentum	step	gamma	accuracy
6	resnet	10	0.005	0.1	4	0.1	0.994667
4	resnet	10	0.005	0.1	3	0.1	0.994000
7	resnet	10	0.005	0.1	4	0.5	0.993667
1	resnet	10	0.005	0.0	3	0.5	0.993000
3	resnet	10	0.005	0.0	4	0.5	0.993000
5	resnet	10	0.005	0.1	3	0.5	0.993000
0	resnet	10	0.005	0.0	3	0.1	0.991667
2	resnet	10	0.005	0.0	4	0.1	0.991667

The model that was trained with momentum, step size, and gamma equal to 0.1, 4, 0.1 respectfully was found to be the best. In the graph below the training procedure was illustrated:



After hyperparameter tuning it was tested on data set with 0, 10, 20, and 30% noise. In the table below results for testing data with 10% noise is illustrated:

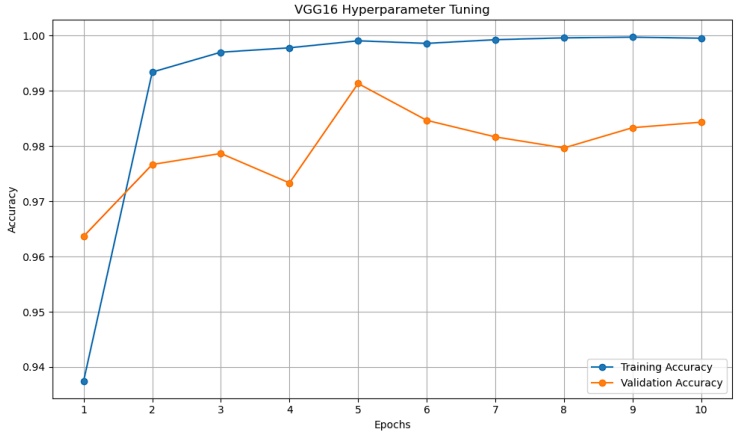
	Class	Accuracy	Recall	Precision	F_score
0	Bean	100.0	1.00	0.99	0.99
1	Bitter_Gourd	99.0	0.99	0.99	0.99
2	Bottle_Gourd	97.5	0.97	0.99	0.98
3	Brinjal	97.0	0.97	0.95	0.96
4	Broccoli	99.5	0.99	0.98	0.98
5	Cabbage	98.5	0.98	1.00	0.99
6	Capsicum	97.0	0.97	1.00	0.98
7	Carrot	100.0	1.00	1.00	1.00
8	Cauliflower	97.5	0.97	0.99	0.98
9	Cucumber	99.0	0.99	0.97	0.98
10	Papaya	89.5	0.90	0.99	0.94
11	Potato	100.0	1.00	0.99	0.99
12	Pumpkin	100.0	1.00	0.93	0.96
13	Radish	100.0	1.00	1.00	1.00
14	Tomato	100.0	1.00	1.00	1.00
15	Total	98.3	0.98	0.98	0.98

VGG16

Results of hyperparameter tuning:

	model	num_epochs	learning_rate	momentum	step	gamma	accuracy
4	alexnet	10	0.005	0.1	3	0.1	0.991333
6	alexnet	10	0.005	0.1	4	0.1	0.990000
5	alexnet	10	0.005	0.1	3	0.5	0.987667
3	alexnet	10	0.005	0.0	4	0.5	0.987000
7	alexnet	10	0.005	0.1	4	0.5	0.987000
1	alexnet	10	0.005	0.0	3	0.5	0.984333
0	alexnet	10	0.005	0.0	3	0.1	0.982333
2	alexnet	10	0.005	0.0	4	0.1	0.980333

The model that was trained with momentum, step size, and gamma equal to 0.1, 3, 0.1 respectfully was found to be the best. In the graph below the training procedure was illustrated:



After hyperparameter tuning it was tested on data set with 0, 10, 20, and 30% noise. In the table below results for testing data with 10% noise is illustrated:

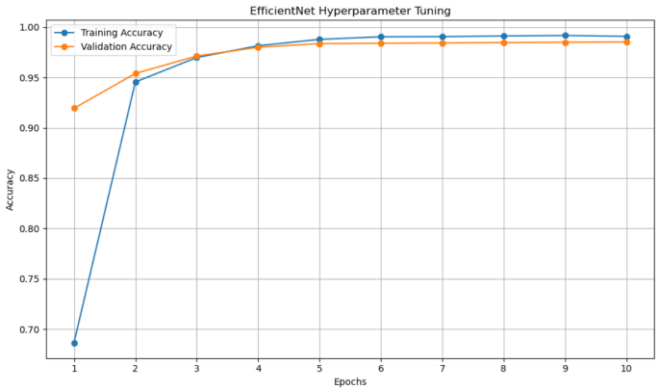
	Class	Accuracy	Recall	Precision	F_score
0	Bean	100.00	1.00	0.98	0.99
1	Bitter_Gourd	99.00	0.99	0.99	0.99
2	Bottle_Gourd	100.00	1.00	1.00	1.00
3	Brinjal	98.00	0.98	0.98	0.98
4	Broccoli	99.50	0.99	0.99	0.99
5	Cabbage	98.00	0.98	1.00	0.99
6	Capsicum	90.50	0.91	1.00	0.95
7	Carrot	99.50	0.99	1.00	0.99
8	Cauliflower	98.50	0.98	0.99	0.98
9	Cucumber	99.00	0.99	0.99	0.99
10	Papaya	100.00	1.00	0.97	0.98
11	Potato	100.00	1.00	0.94	0.97
12	Pumpkin	100.00	1.00	0.99	0.99
13	Radish	98.50	0.98	1.00	0.99
14	Tomato	99.00	0.99	0.97	0.98
15	Total	98.63	0.99	0.99	0.98

EfficientNet

Results of hyperparameter tuning:

	model	num_epochs	learning_rate	momentum	step	gamma	accuracy
4	efficientnet	10	0.005	0.1	3	0.1	0.988000
0	efficientnet	10	0.005	0.0	3	0.1	0.987333
6	efficientnet	10	0.005	0.1	4	0.1	0.987333
7	efficientnet	10	0.005	0.1	4	0.5	0.986333
3	efficientnet	10	0.005	0.0	4	0.5	0.985667
1	efficientnet	10	0.005	0.0	3	0.5	0.985333
2	efficientnet	10	0.005	0.0	4	0.1	0.984667
5	efficientnet	10	0.005	0.1	3	0.5	0.983667

The model that was trained with momentum, step size, and gamma equal to 0.1, 3, 0.1 respectfully was found to be the best. In the graph below the training procedure was illustrated:



After hyperparameter tuning it was tested on data set with 0, 10, 20, and 30% noise. In the table below results for testing data with 10% noise is illustrated:

	Class	Accuracy	Recall	Precision	F_score
0	Bean	99.50	0.99	0.99	0.99
1	Bitter_Gourd	99.00	0.99	0.96	0.97
2	Bottle_Gourd	91.00	0.91	0.99	0.95
3	Brinjal	98.50	0.98	0.95	0.96
4	Broccoli	98.00	0.98	0.99	0.98
5	Cabbage	99.00	0.99	0.99	0.99
6	Capsicum	97.50	0.97	0.99	0.98
7	Carrot	99.50	0.99	1.00	0.99
8	Cauliflower	97.00	0.97	0.99	0.98
9	Cucumber	99.00	0.99	0.93	0.96
10	Papaya	95.50	0.95	0.98	0.96
11	Potato	99.50	0.99	0.99	0.99
12	Pumpkin	99.50	0.99	0.98	0.98
13	Radish	99.50	0.99	1.00	0.99
14	Tomato	100.00	1.00	1.00	1.00
15	Total	98.13	0.98	0.98	0.98

What is left to do?

1. Statistical analysis of the results
2. Analysis of feature extraction capabilities of models
3. Visualization of the results.