Weekly Report

Comparative Analysis of Image Classification Models for Efficient and Accurate Classification across Diverse Image Types

Student: Yusif Mukhtarov

Date: 12/06s/2023

Short info about Research

The fundamental objective of this **quantitative** research is to identify the best image classification models in terms of both time efficiency and accuracy, under different image scenarios such as rotated images, flipped images, images of low quality and so on. Several widely recognized models, including LeNet-5, AlexNet, VGGNet, ResNet, MobileNet, EfficientNet will be considered for this analysis. **Qualitative** analysis has been made to pick those models since they are considered by experts to be the best.

The dataset chosen for this experiment will be vegetable custom datasets that incorporate the required image conditions. Data preprocessing will be carried out to facilitate the suitable conditions for each model, involving adding noise.

To achieve an in-depth and unbiased analysis, the study will take into account several performance metrics. Classification accuracy will primarily measure the model's performance. Simultaneously, precision, recall, and F1-Score will also be considered to provide a good understanding of the model's performance. Moreover, the time taken for training and prediction will be recorded to measure the model's efficiency. Additionally, the model's size in terms of parameters will also be observed.

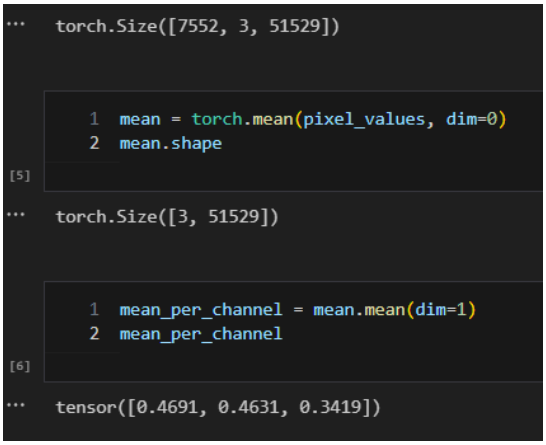
Week 8

Have been done:

Since I still do not have access to the training environment because ADA computers are turned off so I can not SSH into my research environment, and not now I am waiting for access to GW training environment, I decided to research and find out good feature extraction methods to extract features from noisy images. Feature extraction is taking less computation power so my computer could afford it.

1. Firstly, I changed the code for data loading and created method (load_data) where I have added 2 input parameters which will control the percentage of noise which will be added to images and percentage of data that will be taken. The last parameter will help me deal with less data which will require less computational power.

2. I have decided to count from scratch normalization parameters which are applied to images to normalize them. Before I was using default parameters publicly available and calculated from ImageNet dataset, so I decided to calculate for my dataset. Considering computational limitations for mean calculations I first found the mean values of pictures for each pixel, then easily calculated the values for three channels.



Although the mean calculations were easy, the same strategy can not be applied for calculation of standard deviation. Let me firstly show the formula of standard deviation:

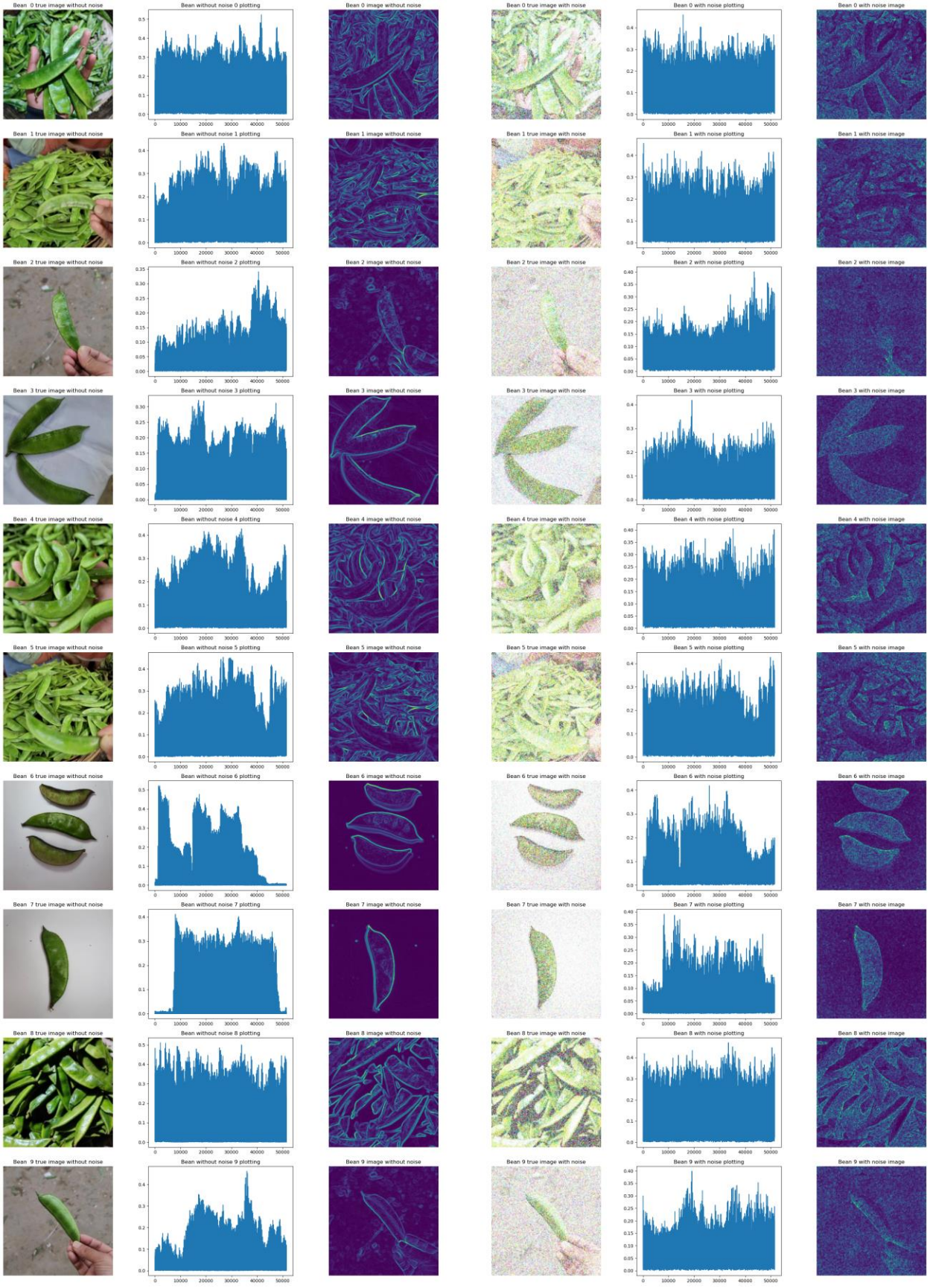
$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{N}}$$

As it is obvious from the formula above, the mean of standard deviations of pictures pixels will not be equal to true standard deviation. For that I should firstly raise to the power of two of standard deviations of pictures pixels, then find their mean and then calculate the square root of the means:

```
[7]
1 std = torch.std(pixel_values, dim=0)
2

[8]
1 std_sq = torch.square(std)
2 std_mean = std_sq.mean(dim = 1)
3 std_mean

... tensor([0.0539, 0.0516, 0.0573])

[9]
1 std_mean = torch.sqrt(std_mean)
2 std_mean

... tensor([0.2322, 0.2272, 0.2394])
```

All the calculations can be found in statistical_analyses.ipynb

3. Then I have tested several feature extraction methods and find out Sobel which showed the best results among all of others:


Bean 0 true image without noise
Bean without noise 0 plotting
Bean 0 image without noise
Bean 0 true image with noise
Bean 0 with noise plotting
Bean 0 with noise image
Bean 1 true image without noise
Bean without noise 1 plotting
Bean 1 image without noise
Bean 1 true image with noise
Bean 1 with noise plotting
Bean 1 with noise image
Bean 2 true image without noise
Bean without noise 2 plotting
Bean 2 image without noise
Bean 2 true image with noise
Bean 2 with noise plotting
Bean 2 with noise image
Bean 3 true image without noise
Bean without noise 3 plotting
Bean 3 image without noise
Bean 3 true image with noise
Bean 3 with noise plotting
Bean 3 with noise image
Bean 4 true image without noise
Bean without noise 4 plotting
Bean 4 image without noise
Bean 4 true image with noise
Bean 4 with noise plotting
Bean 4 with noise image
Bean 5 true image without noise
Bean without noise 5 plotting
Bean 5 image without noise
Bean 5 true image with noise
Bean 5 with noise plotting
Bean 5 with noise image
Bean 6 true image without noise
Bean without noise 6 plotting
Bean 6 image without noise
Bean 6 true image with noise
Bean 6 with noise plotting
Bean 6 with noise image
Bean 7 true image without noise
Bean without noise 7 plotting
Bean 7 image without noise
Bean 7 true image with noise
Bean 7 with noise plotting
Bean 7 with noise image
Bean 8 true image without noise
Bean without noise 8 plotting
Bean 8 image without noise
Bean 8 true image with noise
Bean 8 with noise plotting
Bean 8 with noise image
Bean 9 true image without noise
Bean without noise 9 plotting
Bean 9 image without noise
Bean 9 true image with noise
Bean 9 with noise plotting
Bean 9 with noise image

As is obviously seen from the picture (one in good quality is available in my github, statistical_analyses.ipynb) in most cases it is able to find out the patterns of vegetables. In histograms the pixel values after feature extraction are plotted (50000 on X axes means 227x227 pixels = 51529)

Currently I am limited to computation power of my computer, but after getting access to GW training environment which should be given in a few days I will test my already prepared training models, visualize the last CNN layers with extracted features and analyze them as it was done for Sobel feature extraction model to find out the best model for classification of noisy images. Tomorrow I will start to train data gotten from extracted features on custom neural network, I will also test traditional machine learning algorithms. There is not much left all the code for training and testing for all CNN models are ready, I have just need to train them get, get the data from the trained models, further analyze them and do hyper tuning for some models to get the best accuracy from them, and then finally summarize all analyzes and outcomes of the research and create a table with the resulting parameters of the tested models.