AIML421 Project

Hien Nguyen, 300199540

Due: 24 October 2021

Introduction

For many years, deep learning and neural networks have been well-known for their ability to tackle large and highly complex Machine Learning tasks such as classifying billions of images, powering speech recognition services, or recommending systems. Moreover, they are versatile, robust, and scalable. As a result, for a classification problem like this one, classifying three different fruits (cherry, strawberry, and tomato) from a set of images, Convolutional Neural Networks (CNNs) is the first choice. We call it CNN because a mathematical operation called convolution is the core of this method. It is a specialized kind of linear operation. In other words, CNN is a type of neural network that uses convolution in place of general matrix multiplication in at least one of its layers (Goodfellow, 2016). Many components decide a neural network architecture. This report will investigate the effects of the various components such as different loss functions (such as Cross Entropy vs. Negative Log-Likelihood), batch sizes (such as 100 vs. 32 vs. 64 vs. 128), optimizers (such as Adam vs. AdaGrad vs. RMSProp), and different regularization strategies (such as Dropout vs. L2 Regularization) on the model's accuracy score and loss. Additionally, the report will also compare the performance between a CNN model built from scratch with a simple, baseline neural network using Multi Layer Perceptron (MLP) model and the pre-trained AlexNet model.

Problem Investigation

As the name suggests, exploratory data analysis, or in short EDA, is a process of summarizing, visualizing, and becoming intimately familiar with the essential characteristics of the data. Results obtained from EDA could help significantly in data preprocessing and transforming, thus improving model performance later.

The dataset includes 4500 images, which are distributed equally to each type of fruit. Thus we have 1500 images of cherry, 1500 of strawberry, and 1500 of tomato. Each dataset is split into an 80% training set of 1200 images and a 20% test set of 300 images. Therefore,

the total training set size is 3600 images, and the total test set size is 900. The EDA will only be performed on the training set.

Though most image sizes are 300x300 pixels, the minimum width is 179 pixels, the maximum width is 870 pixels, the minimum height is 138 pixels, and the maximum height is 957 pixels. Therefore, the images should be resized so that they can all have a consistent size.

Table 1: Summary of images' width

count	3600.0
mean	299.5
std	14.9
min	178.0
25%	300.0
50%	300.0
75%	300.0
max	870.0

Table 2: Summary of images' height

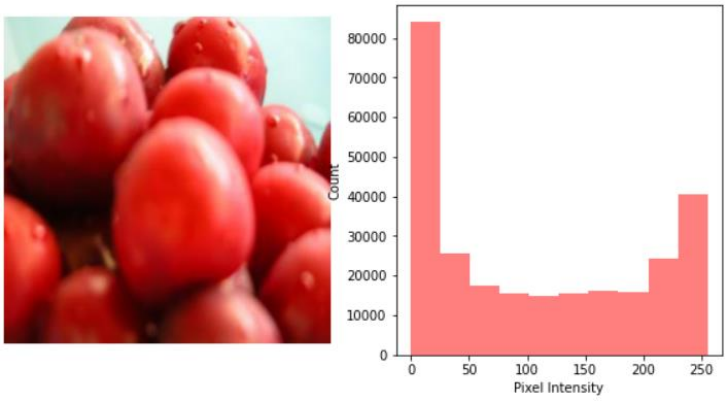
count	3600.0
mean	299.3
std	18.0
min	138.0
25%	300.0
50%	300.0
75%	300.0
max	957.0

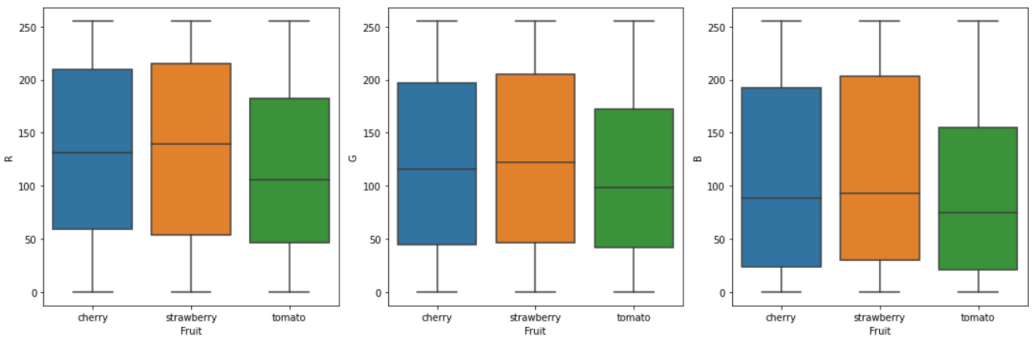
After randomly opening some images, we can see that we have quite a variety of pictures. The fruits in the photos were taken in different settings and have various color variants.

Some fruits are just by themselves. Some are presented with other things, and they are also taken from different angles. However, it is known that we need as many as tens of thousands of images to get robust networks (like the MNIST dataset). Unfortunately, data collection is an expensive and time-consuming process.

Nonetheless, we could still perform some transformations to expand our dataset without requiring us to collect more new images. Those transformation approaches could be center cropping images, flipping the images on the horizontal axis, or rotating images. These transformations significantly increase the number of data points and make the model more robust to variations on an image. Thus, random rotation and random horizontal flip will be applied to the training dataset. In addition, some photos have lots of background noise around the targeted object, so the center crop would help train the model.

Examining the image, we can see that the pixels are not normalized, and on average, strawberry images have more red, green, and blue (RGB) than the other two fruits' images. Cherry's RGB is quite close to strawberry, while tomato's RGB is much less than them, especially red.





A neural network requires only numerical features. Generally, they also need to be normalized (Géron, 2019). The function ToTensor will help transform the image file into a PyTorch Tensor and make the pixels normalized between 0 and 1. However, the dataset will also be fed to the pre-trained model AlexNet which has a different set of normalized values. It uses the ImageNet's parameter, with mean equals [0.485, 0.456, 0.406] and standard deviation equals [0.229, 0.224, 0.225], which are known to work well in practice. Therefore, these values will also be chosen for my model.

Methodology

As mentioned in the Problem Investigation part, several transformations are applied to the training set. These transformations will also be the same for the test set, except the Random Rotation and Random Horizontal Flip method. It is because we will be using the test set to evaluate the model's performance, not for training. On the other hand, we need to resize and normalize the test images to ensure they have the correct dimensions and the same value range that the network is trained on. The given dataset was then loaded to objects called train_loader and test_loader, but only the training set knows the fruit classes. Therefore, images in the test set are hidden from the model.

There are many investigations on the model's hyperparameters. One thing worth noting is that, when comparing different approaches for a particular parameter, all other parameters will be kept the same.

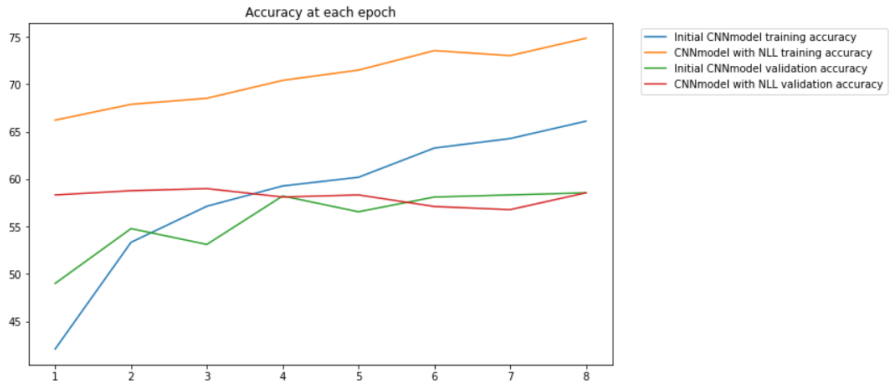
The initial CNN model's parameters are:

- Loss function: Cross-Entropy Loss
- Batch size: 100
- Optimizer: Adam with learning rate equals 0.001
- Regularization strategy: Dropout with the Dropout rate equals 0.25

Table 1: Hyperparameters investigation summary

	Model	Train loss	Test loss	Train accuracy (%)	Test accuracy (%)
0	Initial CNN model	0.76	0.99	66.11	58.56
1	CNN model with NLL loss function	0.54	1.16	74.86	58.56
2	CNN model with batch size 32	0.30	0.62	80.25	56.33
3	CNN model with batch size 64	0.33	0.98	86.83	60.33
4	CNN model with batch size 128	0.30	1.04	92.25	61.56
5	CNN model with optimizer AdaGrad	0.19	2.20	94.56	63.56
6	CNN model with optimizer RMSProp	0.24	1.80	91.00	58.33
7	CNN model with batch L2 Regularization	0.76	0.91	66.86	58.56
8	CNN model with Dropout rate 0.5	0.73	0.92	64.25	57.22
9	Baseline MLP model	0.96	2.95	56.94	42.78
10	Pre-trained AlexNet model	0.07	0.50	98.17	84.56

The first parameter being looked at is the loss function. The loss function compares the desired output and the output produced by the network, then returns some measure of error. For classification problems, Cross-entropy loss is a popular choice. Since the objective of a classification algorithm is to predict a high probability for the target class, if the model estimates a low probability, the Cross-Entropy will penalize the model and get greater by an amount called the Kullback Leibler (KL) divergence (Géron, 2019). On the other hand, the Negative Log-Likelihood loss function (NLL) punishes the model for making the correct prediction with smaller probabilities. The goal is to maximize the model's log-likelihood and minimize the NLL. The combination of NLL and a LogSoftmax layer is very similar to the Cross-entropy loss function. It is the case as the test accuracy at the final epoch for both models is identical (58.56%). The loss and accuracy trends between the two models on the test set vary differently; however, they are very similar for the training set. Additionally, while the gap between training accuracy and test accuracy of the initial model is around 8%, the gap of the model with NLL is double (16%). Thus, it seems that this model is heavily overfitted.



While keeping all other hyperparameters constant, batch size 128 has the highest accuracies (61.56%), but batch size 32 has the lowest test losses (0.62) among four batch size options. The main benefit of using a large batch size is that the training algorithm will see more instances per second, thus helping with efficient GPUs processing. However, a large batch size often affects the training stabilities. Therefore, it is believed that the maximum batch size should be 32 as the model may generalize better than the model trained with a larger batch size (Géron, 2019).

Regarding different types of optimizers, in this case, AdaGrad's accuracy is better than Adam's and RMSProp's. By scaling down the gradient vector along the steepest dimensions, AdaGrad helps to correct gradient descent's direction earlier and point it a bit more toward the global optimum. Moreover, It is called adaptive learning rate because it decays the learning rate faster for vertical dimensions than for dimensions with gentler slopes. On the other hand, RMSProp uses an exponential decay rate to reduce the risk of stopping entirely before reaching the global optimum due to the learning rate getting scaled down so much. Adam stands for adaptive moment estimation and is a combination of momentum optimization and RMSProp. The main difference is that it computes an exponentially decaying average rather than an exponentially decaying sum (Géron, 2019).

Regularization method of Dropout rate 0.25 from initial model yields very similar result to L2 regularization, while keeping all other hyperparameters the same. For these models, the gap between training and test accuracy is 8%, which is substantial and indicates that the model is still overfitted. However, increasing the dropout rate (from 0.25 to 0.5) reduces the training set's accuracy score slightly while the gap is still significant. However, if we decrease the rates further, the model will be even more overfitted.

After all the hyperparameters and models investigation, transfer learning is the best approach (84.6% accuracy). The AlexNet architecture was chosen for this problem. AlexNet was the first to stack convolutional layers directly on top of one another instead of stacking a pooling layer on top of each convolutional layer. It is a series of large convolutional layers, pooling layers, and fully connected layers at the end (Géron, 2019). The output layer of AlexNet is 1000; hence this output layer was edited to three to reflect the three fruit classes for this problem. To preserve the architecture, we freeze the pre-trained weights and biases and only change the classifier sequence. The new classifier sequence uses the Rectified Linear Unit (ReLU) activation function, Dropout rate 0.4, and the logarithmic softmax function. The primary purpose is we hold in state all the convolutional layers that the AlexNet has already been trained on and focus on the last series of fully connected layers. It is to leverage the well-trained filters and apply them to our dataset. One thing worth noting is that the chosen optimizer for this transfer learning model is AdaGrad since it gave the best result compared to other optimizers.

Result discussions

The MLP baseline model is basically an Artificial Neural Network (ANN) with an input size of 150,528 (computed from 224x224x3, the image is 224x224 pixels with three color channels), output size of three, 120 neurons in one hidden layer, and then 84 neurons in the next hidden layer. First Linear fully connected layer was created with 120 neurons. As one layer passes to the next, the second layer has 84 neurons, and the last layer is the output layer with three neurons corresponding to three classes. For the forward function, ReLU is the activation function for all the layers, except for the last output layer as this is a multiclass classification problem, the activation function for this layer is logarithmic softmax. For this MLP model, we have a huge number of parameters (18,063,360), thus a longer training time. If we sum the parameters up, we have 18,073,899 parameters compared to only 9,441,283 parameters from the transfer learning model.

MLP model's parameters

```
18063360
     120
   10080
      84
     252
       3
_______
18073899
```

Transfer learning model's parameter

```
9437184
   1024
   3072
      3
_______
9441283
```

As shown in Table 1, the test accuracy of the MLP model is the lowest. Though the MLP model has some fundamental building blocks such as fully connected layers and sigmoid activation functions, it lacks the crucial ones: convolutional layers and pooling layers to detect complex patterns in the visual field. Furthermore, the MLP model easily breaks down for larger images because of the huge number of parameters it requires. Otherwise, we have to severely restrict the information transmitted to the next layer, hence the lower performance score. On the other hand, CNN's convolutional layer allows neurons to only connect to pixels in their receptive fields and not every single pixel in the input image. Therefore, the network could first collect small low-level features in the first hidden layer, then combine them into more extensive higher-level features in the next hidden layer, and so on. The pooling layer's primary purpose is to reduce computations, memory usage, and the number of parameters, thus, makes it more computational efficient that requires much fewer parameters than an MLP model (Géron, 2019).

Conclusions and future work

Though using the pre-trained AlexNet model for transfer learning gives a good performance score, AlexNet is considered an old model as it was first introduced in 2012. Nowadays, many other architectures yield remarkable results in the image classification field that are worth trying. They are CoAtNet-7, CoAtNet-6 or ViT-G/14. However, AlexNet is fairly simple and easy to use.

Regarding the hyperparameters investigation, cross-entropy loss function, mini-batch size of 128, AdaGrad optimizer, and Dropout regularization could be the go-to parameters for this dataset. It is because they give better results than their peer parameters.

Furthermore, for image classification, CNN should be the primary choice since it emerged from studying the brain's visual cortex to detect complex patterns. With some advanced

computational power and the available training data, CNNs have achieved superhuman performance on some complex visual tasks.

References

Géron, Aurélien. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition. 2nd edition. O'Reilly Media, Inc., 2019. Print.

Goodfellow, Ian., Bengio, Yoshua., Courville, Aaron. (2016). Deep Learning. MIT Press. http://www.deeplearningbook.org