ECE/SIOC 228 Project Final Report

Peiyuan Zhang , Pengfei Ye , Renwei Bai
University of California , San Diego
p7zhang@ucsd.edu, pye@ucsd.edu, rbai@ucsd.edu

Abstract

The project present results from training different models on dataset Fruit-360. A good dataset called fruit-360 is found on Kaggle, base on which we decided to implement a fruit recognition model via ML. The aim with this project is to get a better understanding of the different models and function of specific layers. Two CNN models: VGG11,Prototype(one model we created) are applied first to observe the performances between them. Furthermore, some modifications are made to improve the performance of our model. Finally, the effects of some hyper parameters are looked into.

https://github.com/UnlimitedSSS/ECE228-Team51.git

1 Introduction

1.1 Background

Image recognition, a subcategory of Computer Vision and Artificial Intelligence, represents a set of methods for detecting and analyzing images to enable the automation of a specific task. It is a technology that is capable of identifying places, people, objects and many other types of elements within an image. Recent work has seen the resurgence of feature-based methods, used in conjunction with machine learning techniques and complex optimization frameworks.The advancement of Deep Learning techniques has brought further life to the field of computer vision. The accuracy of deep learning algorithms on several benchmark computer vision data sets for tasks ranging from classification, segmentation and optical flow has surpassed prior methods.

1.2 Motivation

Our motivation is to have a better understanding of what we have learnt about image recognition in the course and test our skills, furthermore we the effects of the hyper parameter is not touched deep in the lecture, so we decided to carry out some experiment in order to have a better understanding.

1.3 Overview

We first augment the image data with gray scale value and flip them to have them further augmented. Then we implement our original model to test its performance and make modification on our original model to improve its performance. Later, we compare its performance with VGG11. Last, we carry out experiment on hyper parameters to figure out what influence they will have and further improve our model.

1.4 Contribution

1. Implement a CNN model and tested its performance.
2. Improving our understanding of CNN model like VGG.
3. Experimented on ReLU and batch-norm.
4. Experimented on hyper parameters.

NeurIPS 2021 AI for Science Workshop.

2 Related works

There are many related works using the dataset fruit-360, people are not only implementing image recognition but also trying to use ML to extract information from images, for example how much water is in the fruit and how much fructose. Additionally, we trained a deep neural network that can recognize fruits from photographs. This follows the current trend of businesses experimenting with augmented reality. Google stated that it is working on a Google Lens application during its annual I/O conference. This software will provide users with a wealth of information about the objects they are pointing at. As a preliminary step, an application like this must be accurately identified. The program was later integrated into the Google Assistant and Google Photos apps later in 2017. To detect objects, the software currently uses a deep neural network. To perform fruit recognition, researchers describe fruit at- tributes with quantitative measurement and formulate the controlled vocabulary and equation to include traits features [1].To facilitate consistent terminology pertaining to shape, a controlled vocabulary focusing specifically on fruit shape traits was developed. Mathematical equations were established for the attributes so that objective, quantitative measurements of fruit shape could be conducted,combined with PCA and subsequent QTL analyses. Some researchers perform random decision forest algorithm on sorting papaya into different ripeness stage [2].

3 Data

3.1 Fruits-360 data set

We will detail how the data set was built and what it contains in this part. The visuals were created by recording the fruits as they revolved around a motor and then removing frames. A 20-second movie was 20 recorded after fruits were placed in the shaft of a low-speed motor (3 rpm). The background was not homogeneous due to fluctuations in lighting conditions, so the dataset was processed using a background-removal algorithm. The resulted dataset has 90380 images of fruits and vegetables spread across 131 labels. Each image contains a single fruit or vegetable. Up to the experiment we have done, we use 17628 images of 35 fruits as data set. After augmentation, we have 35256 among which 30000 are used as training set, 5256 used as test set and a random pick of 5916 used as validation set. The table of data set used in our experiment is shown in **Table 1**.And several sample images of the data set are given in **Figure 1**.

Table 1: Data set structure

Part	
Parameters	Value
Training size	30000
Test size	5256
Validation size	5916

Figure 1: Sample Images

4 Problem formulation

- Improve over task T
  T - Recognize fruit pictures
- With respect to performance measure P
  P – percentage of fruits picture correctly classified
- Base on experience E
  E - database of images of fruits with given classifications
- Input
  $X := [x_1, x_2, x_3, x_4] \in \mathbb{R}^{4\times100\times100}$, represent 4 channel input (RGB and Gray values)
- Target function $y : Image \to \mathbb{R}^n$, n represent the number of classifications
- Training weights and bias to best fit the set of training examples:

$$\langle X, y_{train}(X)\rangle$$

- Minimizing the squared error:

$$CrossentropyLossfunction : L(y, \hat{y}) \equiv -\sum_{x\in X} y(x)\log\hat{y}(x)$$

5 Methods and approaches

5.1 Design of CNN structure

CNN means Convolution in Neural Networks. And convolution is a mathematical operation on two functions (f and g) that produces a third function (fg) that expresses how the shape of one is modified by the other.

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau = (g * f)(t)$$

The basic layer of our CNN structure is a convolution layer followed by a maxpool layer, this forms the unit layer of our model. Then we repeat the unit layer several times to increase the flexibility of our model to let it better fit the data, in order to avoid overfiting we did not repeat the unit layer too many times. Next, we flat our data and feed them into full-connected layer. At last, we choose softmax function to be our activation function and this was our original model.

$$Softmaxfunction : \sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

5.2 Data preparation and processing

Because of the dataset coming from a relatively controlled environment, limited pre-processing was needed. Following the thought of the paper we read, different from traditional way to prepare the image data, in addition to RGB values from the image, we also add a gray scale value to augment the data, as a result our single input is a tensor of size $4 \times 100 \times 100$ (images in the dataset are preprocessed into a 100×100 JPG file). At last, all the images is flipped up and down to further augment the training data we have in order to fit our flexible model and in case we have a overfiting there. As for the training/testing split part, our code have the parameter that can split the data into two part as parameter is given.

5.3 Prototype model implementation

The final structure of our original model is one that have four unit layers (convolution layer+maxpool layer) and have three FC layers followed by a softmax activation layer. Then train the model for some epochs with back propagation using mini-batch gradient descent and cross-entropy as the loss function. The final classification number was 35 (8 in the milestone) which means the output size of

the softmax function is 35. We feed the model with our data and received a relatively good result which means the model is feasible. The optimizer we use is Adam:

$$w \leftarrow w - \eta \frac{V_{dw}}{\sqrt{S_{dw} + \varepsilon}}$$

$$b \leftarrow b - \eta \frac{V_{db}}{\sqrt{S_{db} + \varepsilon}}$$

Where,

$$Momentum \begin{cases} V_{dw} = \beta_1 V_{dw} + (1 - \beta_1)\, dw \\ V_{db} = \beta_1 V_{db} + (1 - \beta_1)\, db \end{cases} \quad RMSprop \begin{cases} S_{dw} = \beta_2 S_{dw} + (1 - \beta_2)\, dw^2 \\ S_{db} = \beta_2 S_{db} + (1 - \beta_2)\, db^2 \end{cases}$$

Structure of prototype is shown in **Table 2** below:

Table 2: Structure of Prototype

Part		
Layer type	Dimensions	Output
Convolution	$5{\times}5 \times 4 \quad (padding : 2)$	16
Max pooling	$2{\times}2 \quad (stride : 2)$	-
Convolution	$5{\times}5 \times 16 \quad (padding : 2)$	32
Max pooling	$2{\times}2 \quad (stride : 2)$	-
Convolution	$5{\times}5 \times 32 quad (padding : 2)$	64
Max pooling	$2{\times}2 \quad (stride : 2)$	-
Convolution	$5{\times}5 \times 64 \quad (padding : 2)$	128
Max pooling	$2{\times}2 \quad (stride : 2)$	-
Fully connected	4608	2048
Fully connected	2048	64
Fully connected	64	35
Softmax	35	35

5.4 Model modification

After implementing the prototype, we have proved our model to be feasible, next step is to figure out how to improve the performance of our model.First, in order to decrease the variance of our train/test loss to have a smoother loss curve, we add a batch-norm layer after the maxpool layer in each unit. Second, to improve the validate accuracy, we decided to add more non-linearity into our model to better fit our data for we have a input size of 35256. After weighting, we choose ReLU following the batch-norm to be the extra non-linearity.

$$ReLu(x) = \max(0, x)$$

5.5 Comparison with VGG models

To figure out good or bad our model is, we carried out some comparison with other classic CNN model, for example VGG11 and VGG16. We construct VGG11 and VGG16 network without modification and then feed them with the same input for our model to see what train/test loss and validate accuracy them can achieve and how much our model differs from them.

6 Experiments and results

6.1 Experiments with hyper parameters

After comparison, we find that there is still big gap between them. So we tuned some hyper parameters, to improve our model's performance. We tuned learning rate, epoch numbers etc. Apart from improving the performance of our model, during the experiment, we also achieved a better understanding of the effects of these hyper parameters which can help us in the future work.

6.2 Result

6.2.1 Fruits-360 data set

We will detail how the data set was built and what it contains in this part. The visuals were created by recording the fruits as they revolved around a motor and then removing frames. A 20-second movie was 20 recorded after fruits were placed in the shaft of a low-speed motor (3 rpm). The background was not homogeneous due to fluctuations in lighting conditions, so the dataset was processed using a background-removal algorithm. The resulted dataset has 90380 images of fruits and vegetables spread across 131 labels. Each image contains a single fruit or vegetable. Up to the experiment we have done, we use 17628 images of 35 fruits as data set. After augmentation, we have 35256 among which 30000 are used as training set, 5256 used as test set and a random pick of 5916 used as validation set.

6.2.2 Hyper parameters

The hyper parameters being used and the train/test loss curve can be seen below (**Table 3**). We also experimented on the hyper parameters so the table is just the final choice.

Table 3: Hyper parameters of prototype

Part	
Parameters	Value
Training size	30000
Training batch	100
Validation size	5916
Test size	5256
Initial learning rate	1e-5
Learning rate decay rate	0.2
Epochs	300
Optimizer	Adam
Number of classes	35

6.2.3 Quantitative result

The final validate accuracy together with comparison with other model and hyper parameter choice is given by **Table 4** and **Figure 2,3,4,5** below.

Table 4: Comparison of different models

Part			
Model	Learning Rate	Training Epochs	Test Accuracy(%)
Prototype	1e-5	300	96.64
Prototype	1e-5	600	96.62
Prototype+batchnorm	1e-5	300	96.87
Prototype+batchnorm+ReLU	1e-5	300	98.38
Prototype+batchnorm	1e-6	600	92.77
Prototype+batchnorm+ReLU	1e-6	600	98.24
Prototype+batchnorm	1e-6	1000	92.99
VGG-11	1e-5	300	98.64

7 Discussion and summary

After all the tuning and experiments, our model achieved a validate accuracy of 98.24% which is given by the model that have batch-norm and ReLU added with learning rate: 1e-6 ,training data:

30000, training epoch: 600,training batch:100 and learning rate decay:0.2. However, we still can't reach the accuracy of the VGG11 which is 98.64%. For ReLU and batch-norm layers, the batch-norm can improve the performance of the train/test loss curve but barely improve the validating accuracy and for the ReLU function,it can improve the validate accuracy obviously. Lastly, for the hyper parameters, we found that with a lower learning rate,the training/testing loss will decrease slower but sometimes result in a higher accuracy(not too small) its a trade off between training efficiency and the accuracy. For the training epochs, more epoch will lead to a higher accuracy and lower loss but its efficiency decreases as epoch grows,when epoch is too much there is rarely any improvement.

8 Future works

In the future, We will try to combine our model with hardware, complete some auxiliary code part and see how the model performs. We will also test the model in some more complex picture and try to let it distinguish between more similar fruits. Later, we will also try to classify more complex objects like face and cars.

9 Link to our project Github Report

`https://github.com/UnlimitedSSS/ECE228-Team51.git`

10 Individual contribution

Alltheteam members resolve every problem of the project together. Usually, we find a suitable time for everyone and find a place such as library and study room to work on the project. No individual tasks are assigned.

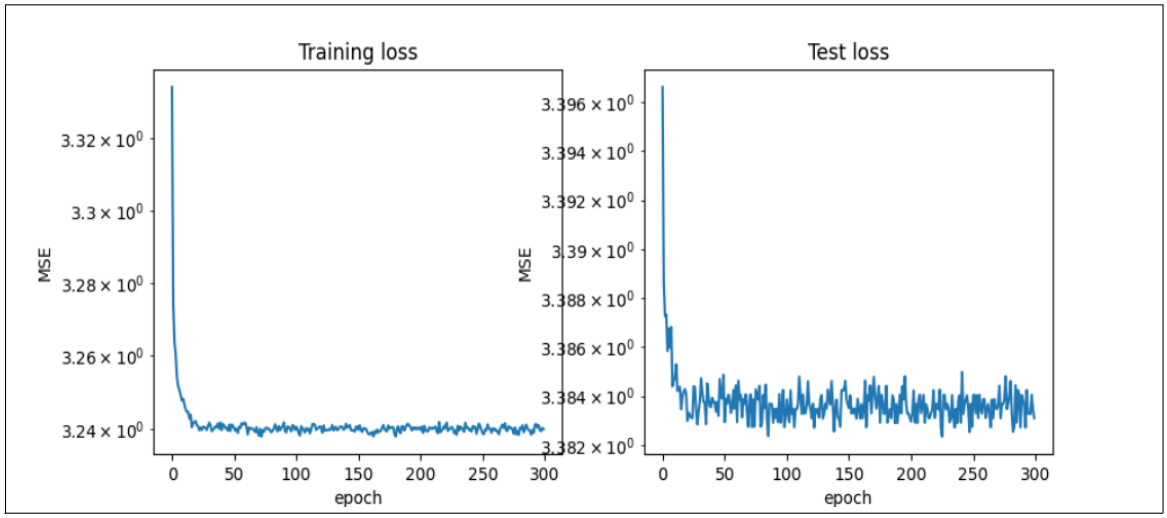

Figure 2: Training loss and test loss for prototype.

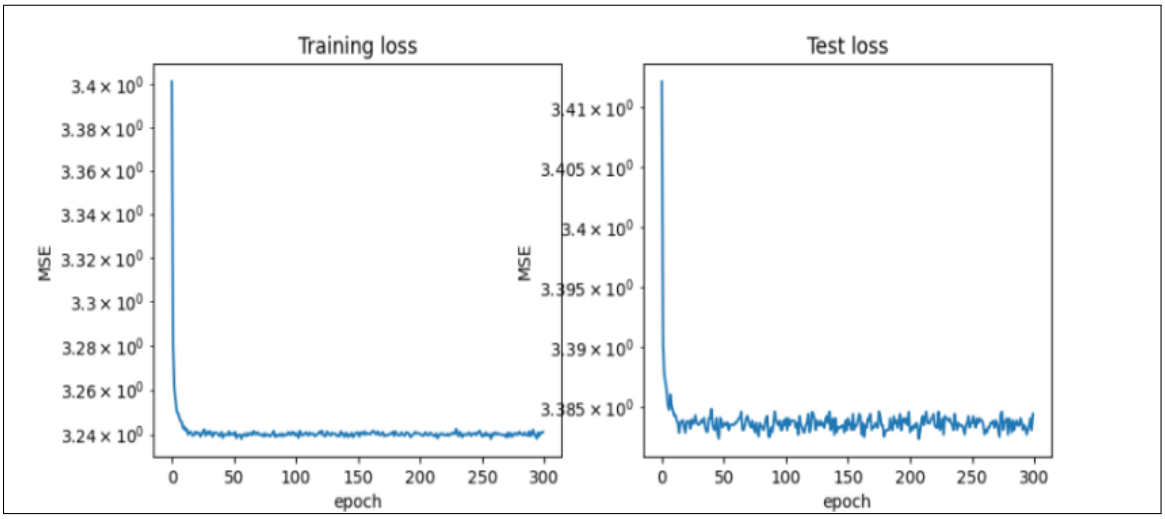

Figure 3: Training loss and test loss for prototype with batchnorm.

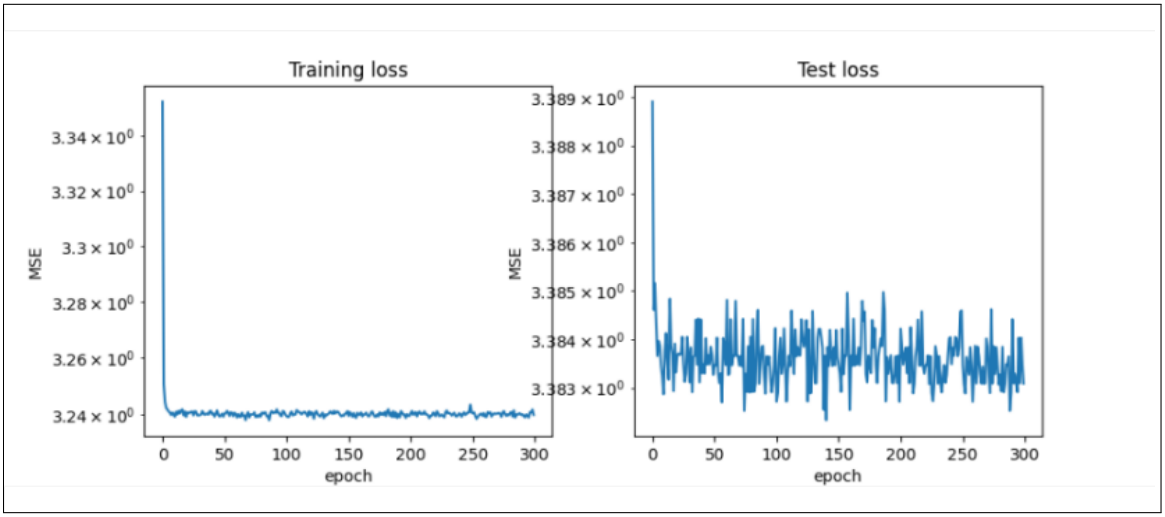

Figure 4: Training loss and test loss for prototype with batchnorm+ReLU.

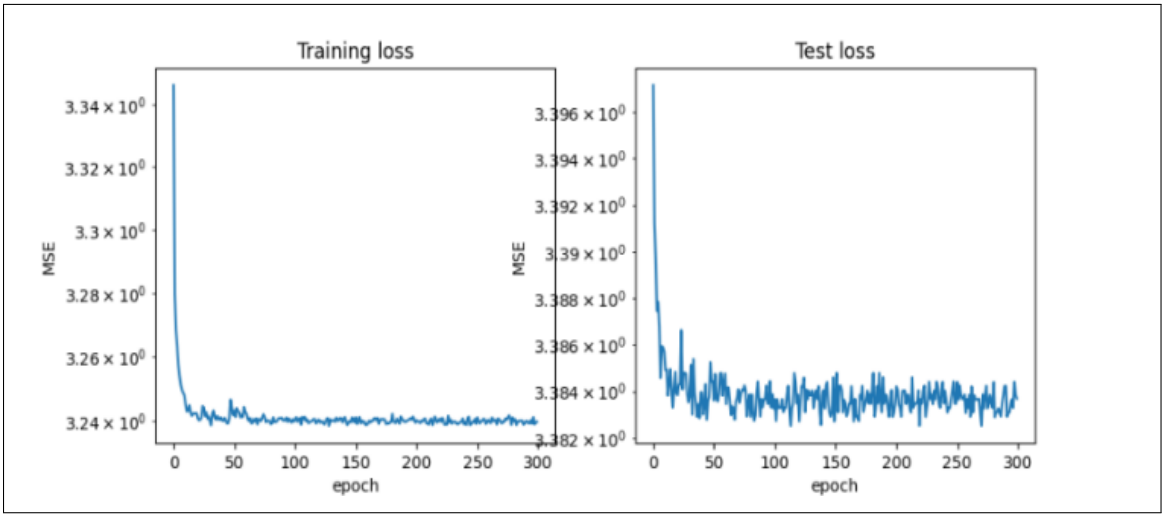

Figure 5: Training loss and test loss of VGG-11.

References

[1] Oltean, M., and Muresan, H. Fruits 360 dataset on Kaggle. [Online; accessed 11.04.2022]. https://www.kaggle.com/datasets/moltean/fruits

[2] L. F. Santos Pereira, S. Barbon, N. A. Val- ous, and D. F. Barbin, “Predicting the ripen- ing of papaya fruit with digital imaging and ran- dom forests,” Computers and Electronics in Agri- culture, vol. 145, pp. 76–82, 2018.

[3] O’Boyle, B., and Hall, C. What is google lens and how do you use it? [Online; accessed 11.04.2022]. https://www.pocket-lint.com/apps/news/google/141075-what-is-google-lens-and-how-does-it-work-and-which-devices-have-it

[4] Sengupta, Abhronil, et al. "Going deeper in spiking neural networks: VGG and residual architectures." Frontiers in neuroscience 13 (2019): 95.

[5]Claesen, Marc, and Bart De Moor. "Hyperparameter search in machine learning." arXiv preprint arXiv:1502.02127 (2015).