HIGH ACCURACY IMAGE CLASSIFICATION OF FRUITS AND VEGETABLES USING GPUs AND CPUs

FINAL REPORT

May 2nd, 2024

Submitted by

Harshavardhan Ambidi(U50513502)
Vignesh Amberru(U46233017)

Karthik Reddy Penninty(U10768951)

Sai Santosh Kumar Devarsetty(U16132241)

Sathvik Reddy Kakulavaram(U33114909)

Lokesh Thambi Mareedu(U87029464)

Rohith Sadanala(U67484898)

Pranav Sai Nikhil Yadlapalli(U48879439)

CIS 6930.003 - Hardware Accelerators for

Machine Learning Dr. Dayane A. Reis

University of South Florida

Table of Contents

Introduction

Nowadays, deep learning relies heavily on hardware accelerators like GPUs (Graphics Processing Units) and CPUs (Central Processing Units) to process large datasets and train complex models efficiently. However, the performance of these accelerators can vary a lot, depending on factors such as the complexity of the model, the size of the dataset, and the specific hardware specifications. Therefore, it's important to evaluate the performance of different hardware accelerators when training deep learning models to ensure optimal efficiency and resource usage.

In this study, we focus on image classification, specifically classifying images using a Convolutional Neural Network (CNN) model. Our goal is to compare the performance of this CNN model when trained on two popular hardware accelerators: GPUs and CPUs.

GPUs are widely used in deep learning applications because they are good at parallel processing and have high memory bandwidth, which makes them well-suited for training large models with massive datasets. On the other hand, CPUs are more versatile and can handle a wide range of applications, but they generally have lower memory bandwidth compared to GPUs.

By carefully analyzing the training time and model accuracy of the CNN model on GPUs and CPUs, we aim to identify the best deep learning models for image classification tasks. This study is important for researchers and practitioners because it provides valuable insights into effectively using hardware accelerators to improve model performance and efficiency in the fields of deep learning and computer vision.

Project Choice:

In today's world, accurate image classification of fruits and vegetables has become crucial in various domains, such as agriculture, food processing, and retail. To achieve high accuracy in this task, the use of Graphics Processing Units (GPUs) and Central Processing Units (CPUs) has gained significant attention. The project focused on developing an accurate image classification system for fruits and vegetables using convolutional neural networks (CNNs) implemented on GPUs and CPUs. This application has significant relevance in the domains mentioned.

GPUs are specialized processors designed to handle complex computations in parallel, making them well-suited for image classification tasks involving deep learning techniques. By leveraging the parallel processing capabilities of GPUs, image classification can be performed with remarkable speed and accuracy, enabling efficient analysis of large datasets.

On the other hand, CPUs, while not as specialized as GPUs for parallel processing, are still capable of executing image classification tasks using deep learning models. CPUs offer flexibility and can be optimized for specific applications, providing an efficient alternative, especially in environments where resources are limited.

By combining the strengths of GPUs and CPUs, a comprehensive solution for high-accuracy image classification of fruits and vegetables can be achieved. In this project, we aim to research and evaluate the performance of GPUs and CPUs in accurately classifying images of various fruits and vegetables, leveraging the power of deep learning techniques. By conducting a comparative analysis, we can identify the optimal hardware platform for different scenarios, considering factors such as accuracy, speed, energy efficiency, and resource utilization.

Motivation:

High-accuracy image classification of fruits and vegetables using GPU and CPU technology has emerged as a crucial requirement in various applications, including agriculture, food safety, and supply chain management. These hardware architectures offer distinct advantages that make them suitable for this task. In agricultural and food processing settings, speed and accuracy are very crucial. GPU architectures, with their parallel processing capabilities, can rapidly analyze large datasets of fruit and vegetable images, enabling real-time monitoring and decision-making. This is particularly valuable for applications such as crop monitoring, yield estimation, and quality control.

When it comes to saving energy, CPUs are a good choice for classifying images. They use less power, which is important in places where there's not much energy available, like out in the fields or in small food factories. CPUs are great for devices that need to be portable or that have limited power. They can be customized to work better with specific ways of identifying what is in images or with different types of computers. This customization can make them work faster and more effectively than usual, which is suitable for tasks like sorting out fruits and veggies or adapting to different growing conditions.

When it comes to handling more images or more complex tasks, GPUs are the way to go. You can just add more parts to make them handle larger amounts of data models. This means your system for telling fruits and veggies apart can grow and handle more images as needed, without any issues. Accuracy is of utmost importance in applications such as food safety and quality control, where misclassification can have severe consequences. Both GPUs and CPUs can be optimized for highly accurate image classification by leveraging techniques like deep learning, ensuring reliable and consistent results.

Bibliographical Study:

Several studies have explored the use of deep learning techniques and hardware accelerators like GPUs and CPUs for image classification tasks:

- Cireşan et al. (2011) demonstrated deep neural networks achieving state-of-the-art results on image datasets and highlighted the importance of GPUs for reducing training times [1].
- Krizhevsky et al. (2012) introduced AlexNet, a groundbreaking deep CNN that leveraged GPUs to achieve breakthrough performance on the ImageNet dataset [2].
- Huang et al. (2017) compared deep learning framework performance across CPUs and GPUs, showing significant GPU speedups for image classification [3].

While these studies focus on GPU comparisons, their insights into the trade-offs between accuracy and performance are relevant to our project, which aims to evaluate the performance of GPUs and CPUs for accurate image classification of fruits and vegetables.

Detailed Description of Methods:

In this project, we employed a Convolutional Neural Network (CNN) architecture for the image classification task. CNNs are a type of deep learning model widely used for computer vision tasks due to their ability to automatically learn and extract relevant features from image data.

The CNN Sequential model was built using the Keras deep learning library with TensorFlow as the backend. The model consisted of several convolutional layers (Conv2D) responsible for extracting features from the input images. These convolutional layers were followed by max-pooling layers (MaxPooling2D) for dimensionality reduction and spatial invariance. Dropout layers were also incorporated to prevent overfitting during training. After going through the convolutional and pooling layers, the data was reshaped into a flat format and passed into fully connected dense layers.

Dataset:

For training the model, we utilized a dataset consisting of images of various fruits and vegetables. This dataset was divided into three subsets: training, validation, and testing. The training set, comprising over 3,000 images, was used to train the model and adjust its parameters. The validation set, containing approximately 300 images, was employed for monitoring the model's performance during training and adjusting to prevent overfitting. The testing set contains around 300 images, and it is reserved for evaluating the final performance of the trained model on unseen data.

Validation set consists of nearly 300 images for evaluating the performance of the model and measuring its effectiveness.

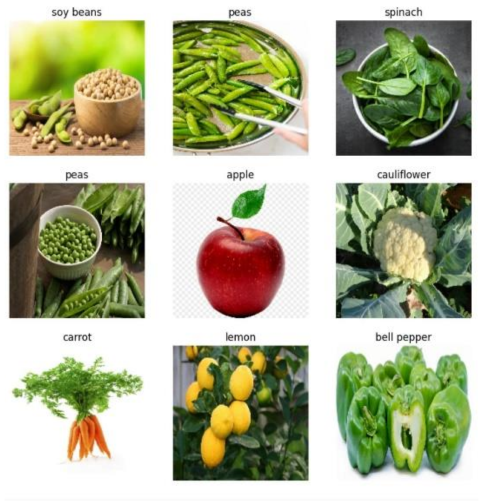


Figure 2: Dataset Image classes

Usually, every CNN model consists of Training and Testing sets for building the model and measuring the performance of it. We specifically included the Validation set to validate the results of the model trained using the Training set against the Validation set to further improve its accuracy.

During the training process, we employed techniques such as data augmentation to increase the diversity of the training data and improve the model's generalization ability. The model was trained using the Adam optimizer. After training, the model's performance was evaluated on the test set using metrics such as accuracy, loss, inference time, and resource utilization (memory usage and power consumption). These metrics were calculated separately for the GPU and CPU implementations.

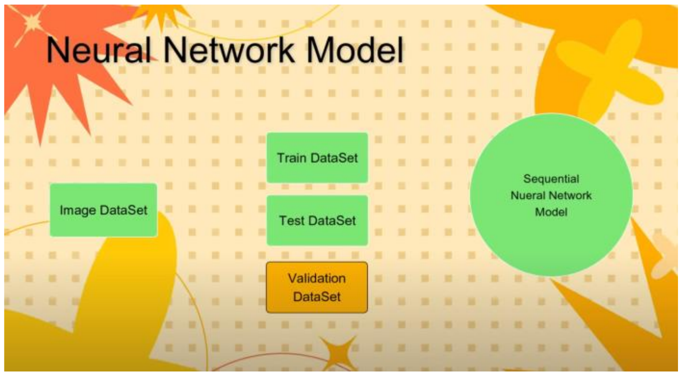
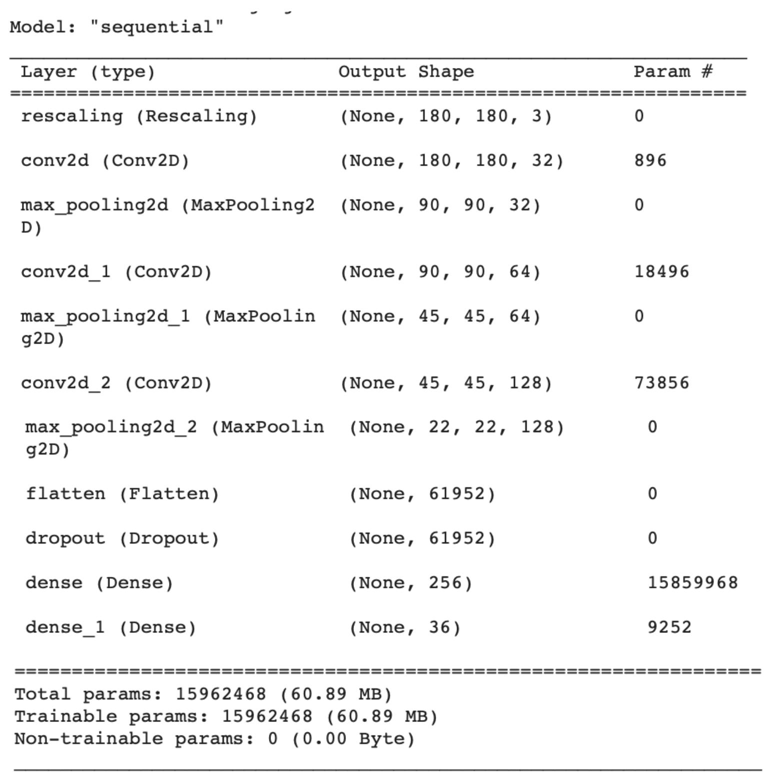


Figure 2: Building the Sequential Model

Progress:

In the initial phase of the project, we implemented a Convolutional Neural Network (CNN) model for Image classification during the project's initial phase. To get the best results on the fruit and vegetable image dataset, we experimented with various architectures, hyperparameters, and optimization strategies. After researching, we settled on a CNN architecture (Sequential model) consisting of several convolutional layers, max-pooling layers, dropout layers for regularization, and fully connected dense layers for classification. We divided the dataset into training, validation, and testing subsets to monitor the model's performance and prevent overfitting.

In the next phase, we implemented the CNN model on both GPU and CPU platforms to compare their performance. We have maintained the batch size of the model as 64 mainly for faster convergence (accurate estimate of gradients) and for providing more generalizable solutions. This indeed had a great impact on the performance of the model, but its memory consumption also increased. We also adjusted the number of filters to simplify the model. Filter adjustment involved increasing the filter size which led to faster training times while still capturing important features in the dataset. This change is made to better deal with smaller datasets (our considered dataset size is 2GB). Moreover, we used the learning rate 0.001. This also helped with faster convergence. We referred to some of the Image classifying projects in the online community to better understand how learning rates work. Furthermore, we also increased the Layer Dropout value to 0.5 to prevent overfitting, and also building more robust and generalizable model. We used the number of units in the first dense layer as 256 to increase the capacity of the model to learn complex representations from the features extracted by the convolutional layers, considerably improving its ability to discriminate between different classes in the dataset.

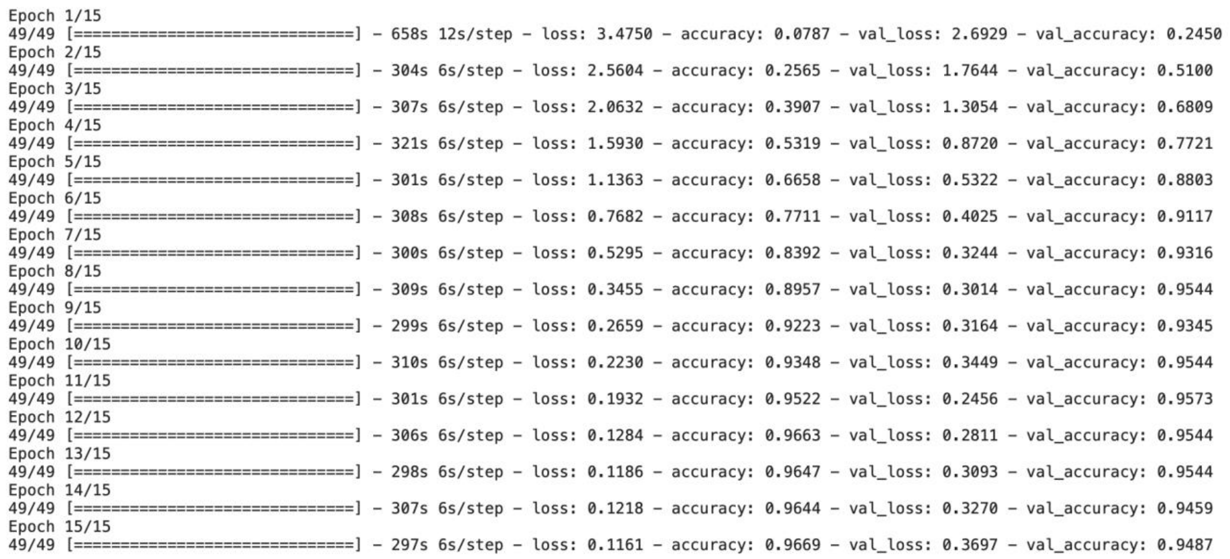


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 180, 180, 3)       0

 conv2d (Conv2D)             (None, 180, 180, 32)      896

 max_pooling2d (MaxPooling2  (None, 90, 90, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 90, 90, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 45, 45, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 45, 45, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 22, 22, 128)       0
 g2D)

 flatten (Flatten)           (None, 61952)             0

 dropout (Dropout)           (None, 61952)             0

 dense (Dense)               (None, 256)               15859968

 dense_1 (Dense)             (None, 36)                9252

=================================================================
Total params: 15962468 (60.89 MB)
Trainable params: 15962468 (60.89 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 3: Model Summary

Throughout the project, we continuously evaluated the model's performance using metrics such as accuracy, loss, inference time, and resource utilization. We also explored techniques to optimize the model for better performance on both GPU and CPU platforms.

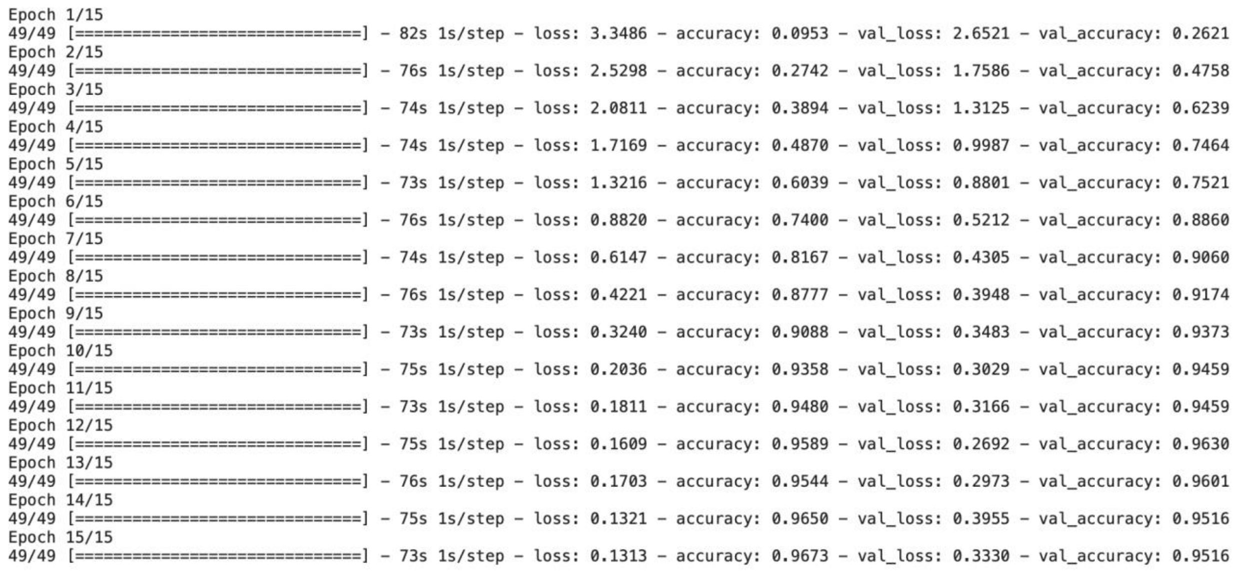
Results and Discussion:



```
Epoch 1/15
49/49 [==============================] - 658s 12s/step - loss: 3.4750 - accuracy: 0.0787 - val_loss: 2.6929 - val_accuracy: 0.2450
Epoch 2/15
49/49 [==============================] - 304s 6s/step - loss: 2.5604 - accuracy: 0.2565 - val_loss: 1.7644 - val_accuracy: 0.5100
Epoch 3/15
49/49 [==============================] - 307s 6s/step - loss: 2.0632 - accuracy: 0.3907 - val_loss: 1.3054 - val_accuracy: 0.6809
Epoch 4/15
49/49 [==============================] - 321s 6s/step - loss: 1.5930 - accuracy: 0.5319 - val_loss: 0.8720 - val_accuracy: 0.7721
Epoch 5/15
49/49 [==============================] - 301s 6s/step - loss: 1.1363 - accuracy: 0.6658 - val_loss: 0.5322 - val_accuracy: 0.8803
Epoch 6/15
49/49 [==============================] - 308s 6s/step - loss: 0.7682 - accuracy: 0.7711 - val_loss: 0.4025 - val_accuracy: 0.9117
Epoch 7/15
49/49 [==============================] - 300s 6s/step - loss: 0.5295 - accuracy: 0.8392 - val_loss: 0.3244 - val_accuracy: 0.9316
Epoch 8/15
49/49 [==============================] - 309s 6s/step - loss: 0.3455 - accuracy: 0.8957 - val_loss: 0.3014 - val_accuracy: 0.9544
Epoch 9/15
49/49 [==============================] - 299s 6s/step - loss: 0.2659 - accuracy: 0.9223 - val_loss: 0.3164 - val_accuracy: 0.9345
Epoch 10/15
49/49 [==============================] - 310s 6s/step - loss: 0.2230 - accuracy: 0.9348 - val_loss: 0.3449 - val_accuracy: 0.9544
Epoch 11/15
49/49 [==============================] - 301s 6s/step - loss: 0.1932 - accuracy: 0.9522 - val_loss: 0.2456 - val_accuracy: 0.9573
Epoch 12/15
49/49 [==============================] - 306s 6s/step - loss: 0.1284 - accuracy: 0.9663 - val_loss: 0.2811 - val_accuracy: 0.9544
Epoch 13/15
49/49 [==============================] - 298s 6s/step - loss: 0.1186 - accuracy: 0.9647 - val_loss: 0.3093 - val_accuracy: 0.9544
Epoch 14/15
49/49 [==============================] - 307s 6s/step - loss: 0.1218 - accuracy: 0.9644 - val_loss: 0.3270 - val_accuracy: 0.9459
Epoch 15/15
49/49 [==============================] - 297s 6s/step - loss: 0.1161 - accuracy: 0.9669 - val_loss: 0.3697 - val_accuracy: 0.9487
```

Figure 4: Training and Validating the CPU model

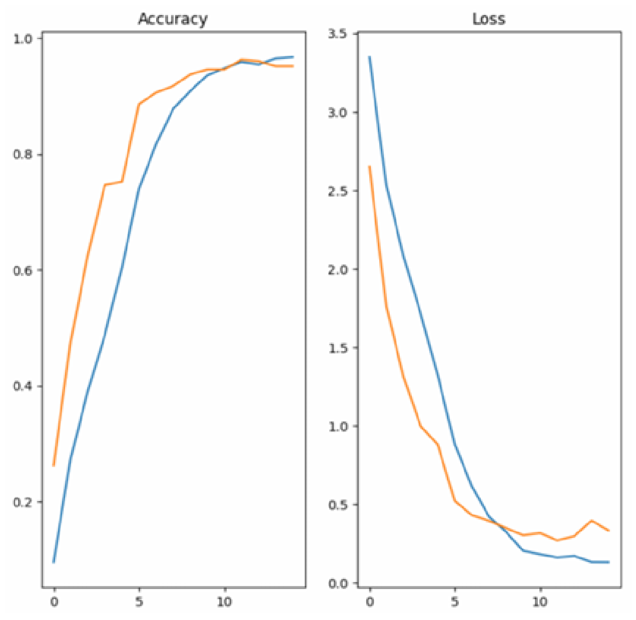
From the above results, we observe that the CPU model has a testing accuracy of 96.69% and a validation accuracy of 94.87%.



```
Epoch 1/15
49/49 [==============================] - 82s 1s/step - loss: 3.3486 - accuracy: 0.0953 - val_loss: 2.6521 - val_accuracy: 0.2621
Epoch 2/15
49/49 [==============================] - 76s 1s/step - loss: 2.5298 - accuracy: 0.2742 - val_loss: 1.7586 - val_accuracy: 0.4758
Epoch 3/15
49/49 [==============================] - 74s 1s/step - loss: 2.0811 - accuracy: 0.3894 - val_loss: 1.3125 - val_accuracy: 0.6239
Epoch 4/15
49/49 [==============================] - 74s 1s/step - loss: 1.7169 - accuracy: 0.4870 - val_loss: 0.9987 - val_accuracy: 0.7464
Epoch 5/15
49/49 [==============================] - 73s 1s/step - loss: 1.3216 - accuracy: 0.6039 - val_loss: 0.8801 - val_accuracy: 0.7521
Epoch 6/15
49/49 [==============================] - 76s 1s/step - loss: 0.8820 - accuracy: 0.7400 - val_loss: 0.5212 - val_accuracy: 0.8860
Epoch 7/15
49/49 [==============================] - 74s 1s/step - loss: 0.6147 - accuracy: 0.8167 - val_loss: 0.4305 - val_accuracy: 0.9060
Epoch 8/15
49/49 [==============================] - 76s 1s/step - loss: 0.4221 - accuracy: 0.8777 - val_loss: 0.3948 - val_accuracy: 0.9174
Epoch 9/15
49/49 [==============================] - 73s 1s/step - loss: 0.3240 - accuracy: 0.9088 - val_loss: 0.3483 - val_accuracy: 0.9373
Epoch 10/15
49/49 [==============================] - 75s 1s/step - loss: 0.2036 - accuracy: 0.9358 - val_loss: 0.3029 - val_accuracy: 0.9459
Epoch 11/15
49/49 [==============================] - 73s 1s/step - loss: 0.1811 - accuracy: 0.9480 - val_loss: 0.3166 - val_accuracy: 0.9459
Epoch 12/15
49/49 [==============================] - 75s 1s/step - loss: 0.1609 - accuracy: 0.9589 - val_loss: 0.2692 - val_accuracy: 0.9630
Epoch 13/15
49/49 [==============================] - 76s 1s/step - loss: 0.1703 - accuracy: 0.9544 - val_loss: 0.2973 - val_accuracy: 0.9601
Epoch 14/15
49/49 [==============================] - 75s 1s/step - loss: 0.1321 - accuracy: 0.9650 - val_loss: 0.3955 - val_accuracy: 0.9516
Epoch 15/15
49/49 [==============================] - 73s 1s/step - loss: 0.1313 - accuracy: 0.9673 - val_loss: 0.3330 - val_accuracy: 0.9516
```

Figure 5: Training and Validating the GPU model

From the above results, we observe that the GPU model has a testing accuracy of 96.73% and a validation accuracy of 95.16%.



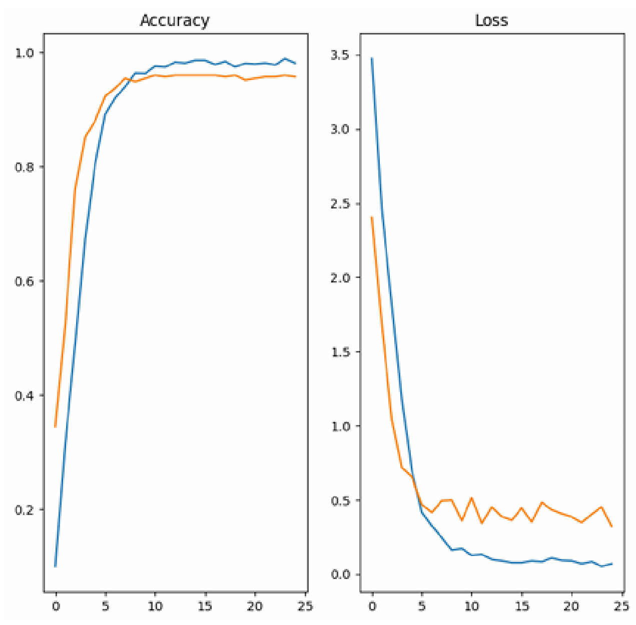
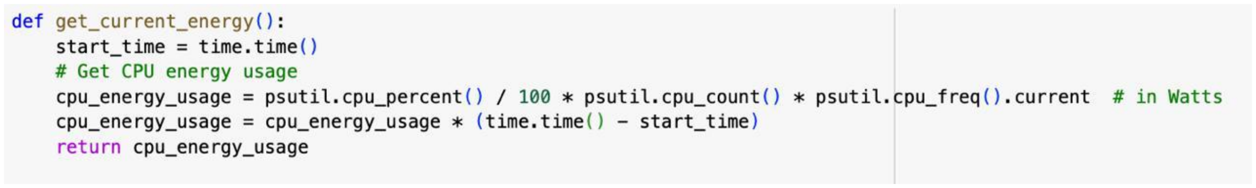


Figure.5 Visualization graph for CPU model **Figure.6 Visualization graph for GPU model**

(Orange line – Training Accuracy/Loss**, Blue line –** Validation Accuracy/Loss**)**

Comparison between evaluation parameters of CPU and GPU:

Parameter	**CPU**	**GPU**
Memory Information (Total)	12978.98 MB	15360 MB
Memory Information (Used)	2980.71 MB	4514.94 MB
Memory Information (Free)	5801.39 MB	10845.06MB
Temperature	79	77
Energy	4.20 J	6.87 J
Time	0.35 S	0.12 s
Energy Efficiency (GOP/s/W)	181.21	110.74
Throughput (GOP/s)	6050.34	6188.85
Power	33.39W	55.89W

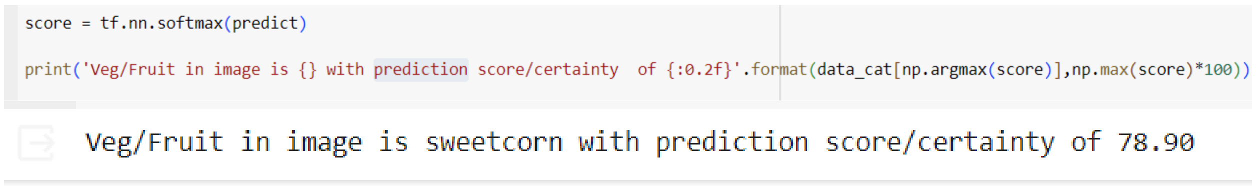


```python
def get_current_energy():
    start_time = time.time()
    # Get CPU energy usage
    cpu_energy_usage = psutil.cpu_percent() / 100 * psutil.cpu_count() * psutil.cpu_freq().current  # in Watts
    cpu_energy_usage = cpu_energy_usage * (time.time() - start_time)
    return cpu_energy_usage
```

CPU energy calculation

Above code calculates the instantaneous power consumption of the CPU in watts (W) using the psutil module. Then it calculates the total energy consumed by the CPU during the time interval between the start time and the current time. This gives energy consumed in Joules(J).

We have used TensorFlow’s SoftMax function to perform the Image Classification process. Both the models have given a prediction score/certainty of more than 75% in predicting the right Image class.



```python
score = tf.nn.softmax(predict)

print('Veg/Fruit in image is {} with prediction score/certainty  of {:0.2f}'.format(data_cat[np.argmax(score)],np.max(score)*100))
```

```
Veg/Fruit in image is sweetcorn with prediction score/certainty of 78.90
```

Conclusion:

In this project, we successfully implemented and evaluated a Convolutional Neural Network (CNN) model for the image classification of fruits and vegetables on both GPU and CPU hardware platforms. The results showcased the superior performance of the GPU implementation in terms of training time and accuracy, while the CPU implementation offered a reasonable alternative with slightly lower accuracy but longer training times.

The findings align with the existing literature, which highlights the advantages of utilizing hardware accelerators like GPUs for computationally intensive tasks like deep learning. However, the relatively small difference in accuracy between the GPU and CPU implementations suggests that CPUs can still be a viable option, especially in resource-constrained environments or when performance is not the primary concern.

Future work could include exploring more advanced model architectures, employing techniques like transfer learning or model compression, and conducting more comprehensive resource utilization and energy efficiency analyses. Additionally, investigating the performance of these implementations on larger datasets or more complex image classification tasks could provide further insights into the strengths and limitations of each hardware platform.

Overall, this project contributes to the understanding of the trade-offs between accuracy, training time, and resource utilization when leveraging GPUs and CPUs for image classification tasks, providing valuable insights for researchers and practitioners in the field of computer vision and deep learning.

Include future usage

References: