Hardware Accelerators for Machine Learning Department Of Computer Science and Engineering

Course Instructor: Dr.Dayane A.Reis
CIS 4930/6930

Project Title(Group 7)

Accelerating Pedestrian Detection with CPU Vs GPU Architecture

CPU(TEAM A)

Mythreye Pasula (U88527486)

Yaswanth Sai Ram Pavan Jami (U22367338)

Chaitanya Sai Dangeti (U77695453)

Vineeth Reddy Betham (U36652271)

GPU(TEAM B)

Sree Vardhani Iragavarapu (U13255029)

Lakshmi Venkata Chaitanya Raju Gedela(U42289643)

Kumar Ganesh Chowdary Naidu(U05647359)

Akshay Reddy Sudini(U44174638)

Table of Contents

Bibliography:

N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005

This research presents the Histogram of Oriented Gradients (HOG), a novel feature descriptor created especially for image recognition by humans. The method by which this descriptor operates is to create a histogram of the edge orientations or gradient directions within each of the small areas, or cells, that make up the image. The strength of the HOG descriptor is its resilience to variations in illumination and shadowing, which enables it to recognize persons in a variety of backdrops and lighting settings with great effectiveness. Tested extensively on pedestrian datasets, the HOG descriptor performed exceptionally well at separating individuals from the background when combined with a linear support vector machine (SVM) classifier.

Additionally, the paper discusses the application of HOG descriptors to areas beyond human detection to areas beyond vehicle detection and facial recognition, which is the focus of the project. Therefore, the papers utilize the HOG descriptor for Image recognition which is relevant to the project's aim of detecting pedestrians through image recognition.

P. Dollar, C. Wojek, B. Schiele and P. Perona, "Pedestrian detection: A benchmark," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009

This study provides a thorough analysis of pedestrian detection systems. In the field of computer vision, this work is crucial, especially for applications including robot vision systems, car safety, and surveillance. To evaluate the efficacy of several pedestrian recognition techniques under a variety of demanding real-world scenarios, such as intricate backgrounds, fluctuating lighting, and varied poses and resolutions, the authors benchmark the techniques.
The main contribution of this study is the creation of a benchmark that allows for the systematic comparison of various detection approaches, allowing for a better understanding of how these systems perform in a variety of contexts. This benchmark assesses not only the accuracy of detection algorithms, but also their failure rates and the effect of image resolution on performance. By offering a detailed study of several methodologies, including both traditional and new techniques, the research shows the strengths and limits of current pedestrian detection technology.

We use benchmarks for assessing the performance of pedestrian detection systems, including those utilizing HOG features or SVM classifiers. By detailing performance across various environmental conditions, which is relevant to the project.

Zhou, L., Wang, H., Jin, Y., Qian, W., Li, J., & Li, J. (2020). Robust visual tracking based on Adaptive Multi-Feature Fusion using the tracking Reliability criterion. Sensors

This paper proposes a better classification that uses an adaptive multi-feature fusion strategy. Its key objective in solving challenging scenarios. It utilizes a feature filter as the baseline and then it introduces a feature adaptive fusion strategy during the filter training process. The paper uses the Histogram of Oriented Gradients (HOG) feature as one of the key components in the multi-feature fusion. This one helps in finding the pedestrians in presence of motion blur and illumination changes which is very helpful for the project in identifying the pedestrians in various environmental situations.

S. Ghosh, A. Dasgupta and A. Swetapadma, "A Study on Support Vector Machine based Linear and Non-Linear Pattern Classification," 2019 International Conference on Intelligent Sustainable Systems (ICISS), Palladam, India, 2019

This paper presents a comprehensive study on the use of Support Vector Machines (SVM) for both linear and non-linear pattern classification tasks. It aims to highlight the versatility and effectiveness of the SVM algorithm in handling different types of classification problems. This paper provides an overview of the theoretical foundations of SVM, including the concept of the optimal hyperplane, the role of the kernel function, and the mathematical formulation of the SVM optimization problem. It discusses the application of SVM for linear classification tasks, where the decision boundary is a hyperplane in the input feature space, and it explains the process of training a linear SVM model and the factors that influence its performance, such as the choice of the regularization parameter. It leverages the interpretability of linear SVM by analyzing the learned model weights to understand which features are most important for the classification task. It Visualizes the decision boundary of the linear SVM to gain insights into how the model is separating the classes which is relevant to project in SVM implementation for image classification.

Gidudu, A., Hulley, G., & Marwala, T. (Year). Classification of images using support vector machines. Department of Electrical and Information Engineering, University of the Witwatersrand. Johannesburg, Private Bag X3, Wits, 2050, South Africa.

The paper presents a detailed approach to using linear Support Vector Machines (SVM) for image classification tasks, outlining several critical steps in the process. Initially, features are extracted from images, including color values, edge detection, and texture information. These features are then used as inputs for the linear SVM classifier. Following feature extraction, a linear SVM model is trained with these features. The paper emphasizes the advantages of using linear SVM for this task, such as its capability to handle high-dimensional data and its robustness against overfitting, which is often superior to other algorithms like neural networks. The feature extraction techniques discussed include Histogram of Oriented Gradients (HOG), binned color features, and color histogram features. These techniques provide a comprehensive set of characteristics that capture the essential visual information from the images.

Problem Statement and Motivation

The primary objective of the project is to for the safety of the pedestrians importantly on the cross walks this present model helps in the betterment of the ADAS (advanced driver assistance system). we tried proposing a pedestrian detection system to detect the vulnerable road users. We used the benchmark INRIA dataset for the validation of the model, we tried it with two approaches one using the CPU and the other using the GPU Execution to understand the differences. We wanted to investigate the right platform for the secure Pedestrian Detection Model and see how it performs with the real time images on both CPU and GPU.

Dataset and Preprocessing:

We're loading the INRIA dataset from the drive, which has a size of approximately 1.1 GB(650 Positive samples, 1400 negative samples). The dataset has been preprocessed to align with the Histogram of Oriented Gradients (HOG) descriptor. We've defined the length and breadth of the pedestrian around the Region of Interest (ROI), and the left-top and right-bottom coordinates help in determining the coordinates of these ROIs. Additionally, parameters for the HOG computation, such as orientations, pixels per cell, and cells per block, have been specified. Overall, our aim is to enhance the visualization of the HOG and the feature representation to make them suitable for input to the pedestrian detection system.

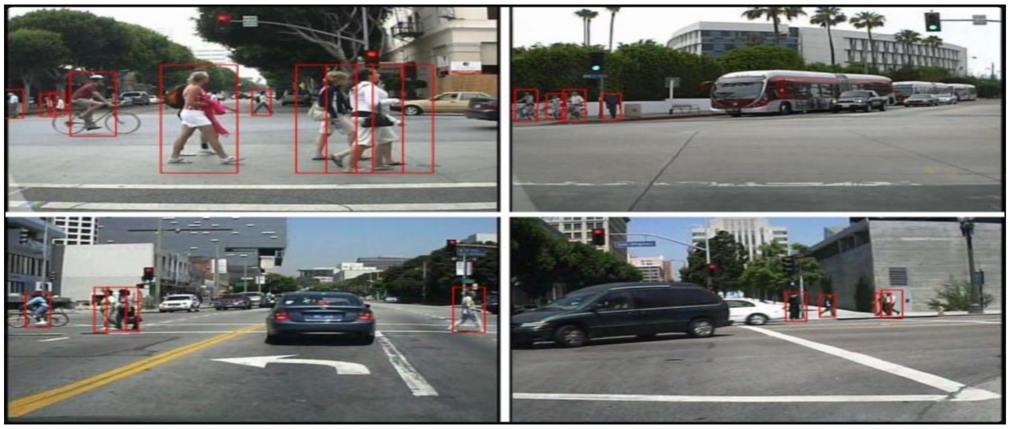


Fig1: Pedestrian Detection

Description and Methodology

HOG Descriptor:

The HOG (Histogram of Oriented Gradients) method is a descriptor used for object detection in image classification tasks. It captures gradients in the image by dividing it into cells, then computing histograms of gradients within these cells. These histograms represent the distribution of gradient orientations in localized portions of the image, known as the Region of Interest (ROI). In our current method, we use the hog() function to compute these histograms from the ROI.

The working of the HOG method involves following steps:

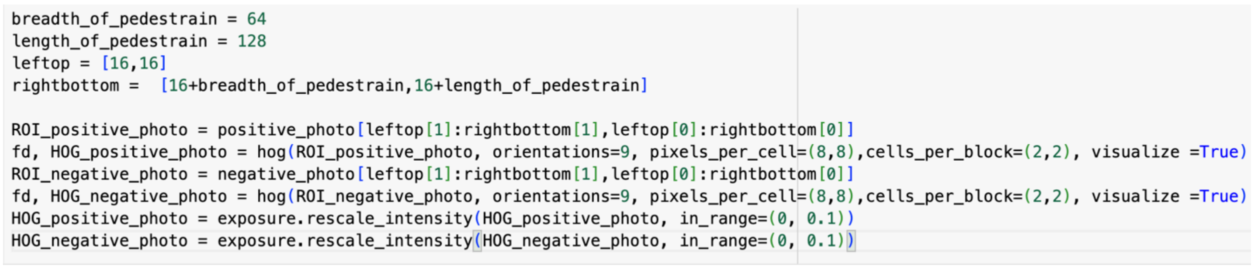
Working of the HOG:

→ Dividing the cells within the Image (preprocessing)

→Computation of each cell using based on the orientations and pixels within the cells

→The grouping of the cells within the regions are made into blocks by the HOG (normalizations on the ROI)

→The block histograms are normalized, and parameters are adjusted based on the object detection window size 64x128 || Cell size 8x8 ||Block size 16x16



```
breadth_of_pedestrain = 64
length_of_pedestrain = 128
leftop = [16,16]
rightbottom =  [16+breadth_of_pedestrain,16+length_of_pedestrain]

ROI_positive_photo = positive_photo[leftop[1]:rightbottom[1],leftop[0]:rightbottom[0]]
fd, HOG_positive_photo = hog(ROI_positive_photo, orientations=9, pixels_per_cell=(8,8),cells_per_block=(2,2), visualize =True)
ROI_negative_photo = negative_photo[leftop[1]:rightbottom[1],leftop[0]:rightbottom[0]]
fd, HOG_negative_photo = hog(ROI_negative_photo, orientations=9, pixels_per_cell=(8,8),cells_per_block=(2,2), visualize =True)
HOG_positive_photo = exposure.rescale_intensity(HOG_positive_photo, in_range=(0, 0.1))
HOG_negative_photo = exposure.rescale_intensity(HOG_negative_photo, in_range=(0, 0.1))
```

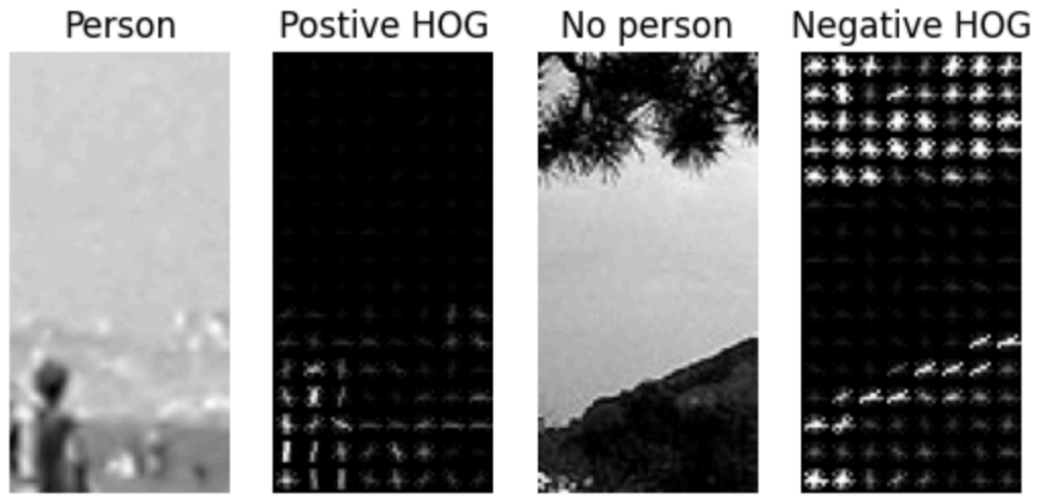


Fig-2: HOG descriptors after plotting the histograms

After the features are extracted from the HOG based on the ROI, the complete features are appended to the feature list, and target labels are assigned to similar instances, such as pedestrian images.

```
features.append(fd)
target.append(0)
```

Structure of the SVM

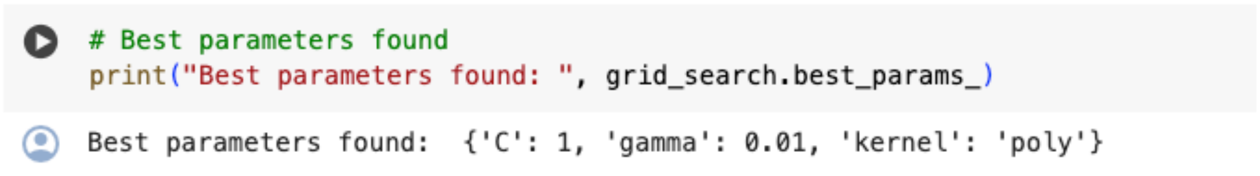
SVM mainly operates on labeled training data, where features are mapped to vectors. A hyperplane is then designed based on these data points to best separate the classes, thereby maximizing the margin. The trick involves optimizing the hyperplane to find the best possible parameters and preventing overfitting as much as possible. This often entails experimenting with different kernel functions to find the best fit for the model. The regularization parameter, denoted as C, is crucial in-this-process.

We found that using SVM with the data from HOG would be much more suitable, as HOG creates feature vectors for us. Additionally, using the grayscale images provided by the HOG descriptor significantly improved the performance of the SVM kernel. We observed a training accuracy of 98% on the dataset. The features obtained from HOG are most suitable, as they are best used for the SVM kernel to find the hyperplane. However, we encountered the problem of overfitting with the SVM classifier. We adjusted the SVM classifier accordingly with the right parameters to fit our model.

Using our first non-optimized model with the train and test data from the INRIA dataset we had a test accuracy of 65%(approximately)

Hyperparameters Tuning

We tried tuning the values of three parameters: the C value, which controls the tradeoff. However, we found that as the C value increased, the model began overfitting with the training data. To achieve a better decision boundary, we adjusted the gamma value of the model. Additionally, to accurately map the input data, we experimented with different kernels ('linear', 'rbf', 'poly'). We used GridSearchCV to try various hyperparameters specified with different combinations, which took approximately 40 minutes to find the best fit. The code snippet below shows the results.



```
# Best parameters found
print("Best parameters found: ", grid_search.best_params_)
```

```
Best parameters found:  {'C': 1, 'gamma': 0.01, 'kernel': 'poly'}
```

Re-Evaluation with the Parameters:

Our model improved significantly after tuning the hyperparameters. It mitigated the issue of overfitting to a certain extent using GridSearchCV, and we were able to find the best parameters for the SVM.

Experimental Results- before and after Tuning with Hyperparameters

Metric	Before Tuning	After Tuning
Test Accuracy	64.37%	71.93%
C	1	1
Kernel	linear	poly

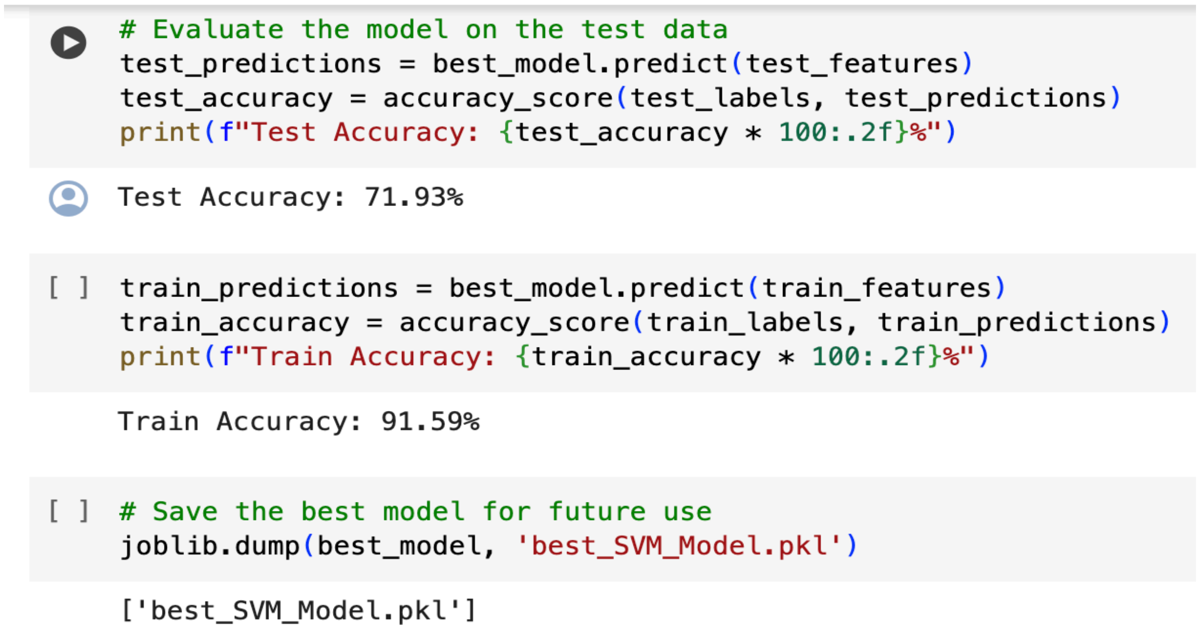
Results and Discussion

We are trying to compare the accuracy from both the CPU and the GPU as part of our model. The runtime of the training data was around 11 seconds on the SVM, and the accuracy on the test data was approximately 72%. On the training data, it ranged from 88% and above, with variations observed in each iteration on both the GPU and the CPU, as well as in the runtime. We tried to plot an average of the runtime accuracies for each iteration in the table. Parameter tuning helped us to reduce the problem with overfitting in the data, but we observed that the training accuracy increased with each run due to the random data split on each iteration in the code. When we consider the power consumption we used the value obtained from the energy to calculate it its taking around 9 W for GPU and CPU it is ranging from (11-15)W

From the experiment, we can conclude that the accuracies are mostly similar for both the CPU and the GPU. However, the time taken, and energy used by the CPU are significantly higher compared to those of the GPU.

Table: Comparison with CPU and GPU

Metric	CPU Training	CPU Testing	GPU Training	GPU Testing
Time	11s	4s	5s	2s
Accuracy	94.5	72	91.59	71
Runtime Variations	Observed	-	Observed	-
Overfitting	Reduced	-	Reduced	-
Data Splitting	Random	-	Random	-



```
# Evaluate the model on the test data
test_predictions = best_model.predict(test_features)
test_accuracy = accuracy_score(test_labels, test_predictions)
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")
```

```
Test Accuracy: 71.93%
```

```
[ ]  train_predictions = best_model.predict(train_features)
     train_accuracy = accuracy_score(train_labels, train_predictions)
     print(f"Train Accuracy: {train_accuracy * 100:.2f}%")
```

```
Train Accuracy: 91.59%
```

```
[ ]  # Save the best model for future use
     joblib.dump(best_model, 'best_SVM_Model.pkl')
```

```
['best_SVM_Model.pkl']
```

Fig 3: snippets on the accuracies

Future Work:

Dataset Preprocessing: The INRIA dataset was preprocessed to optimize its performance for our classification task. Additionally, we explored the feasibility of collecting data directly from VR and FIR wavelengths over time, enabling real-time analysis. This approach enhances the data pool and its relevance to real-world scenarios.

References

1. N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005, pp. 886-893 vol. 1

2. P. Dollar, C. Wojek, B. Schiele and P. Perona, "Pedestrian detection: A benchmark," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 304-311, doi: 10.1109/CVPR.2009.5206631.

3. Zhou, L., Wang, H., Jin, Y., Qian, W., Li, J., & Li, J. (2020). Robust visual tracking based on Adaptive Multi-Feature Fusion using the tracking Reliability criterion. Sensors, 20(24), 7165.

4. S. Ghosh, A. Dasgupta and A. Swetapadma, "A Study on Support Vector Machine based Linear and Non-Linear Pattern Classification," 2019 International Conference on Intelligent Sustainable Systems (ICISS), Palladam, India, 2019, pp. 24-28, doi: 10.1109/ISS1.2019.8908018.

5. Gidudu, A., Hulley, G., & Marwala, T. (Year). Classification of images using support vector machines. Department of Electrical and Information Engineering, University of the Witwatersrand. Johannesburg, Private Bag X3, Wits, 2050, South Africa.

Collab-link -
https://colab.research.google.com/drive/1O2ULo5mFgNtkW_5M3ENUmSczrTkjqeNQ#scrollTo=L9i36alVSWT_