Classical Method on New Datasets: Pedestrian Detection with HOG and CNN

Asher Mai
hanlinm2@illinois.edu

Matthew Tang
mhtang2@illinois.edu

Sid Ahuja
ahuja12@illinois.edu

Summary

As neural network models get larger and more capable, the time it takes to train them are also getting progressively longer. This project aims to reduce the training and inference time of pedestrian detection task by pre-processing the images into Histogram of Oriented Gradients (HOG) Descriptors [1]. Our results show that training neural networks on HOG features instead of images pixels can provide up to 125 times speed up in training time, while maintaining comparable accuracy at the same time.

Method

We train 4 neural network models:
1. A ResNet18 [3] with RGB image pixels as input
2. A ResNet18 with derived HOG feature descriptors as input
3. A 1-layer Convolutional Neural Network (CNN) with HOG feature descriptors as input
4. A fully connected neural network with HOG feature descriptors as input

Given an image, our models detect whether there is a pedestrian in the image or not. We compare the accuracy, precision, recall, receiver operating characteristic (ROC) curves, and time it takes to train one epoch. We aim to find out the speed/accuracy trade off of these four models.

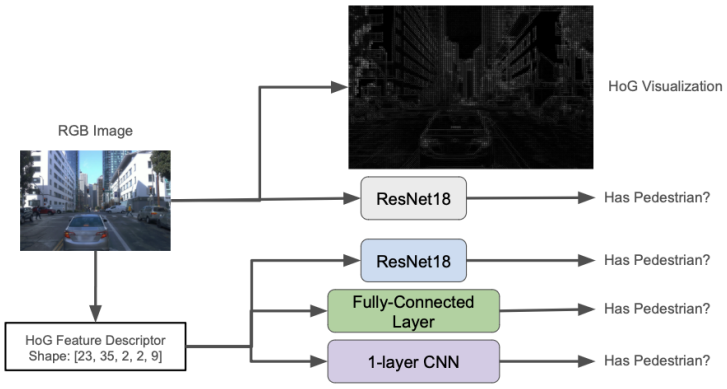

Figure 1. The schematic represents our pipeline

Result

Our results show that training ResNet18 on HOG feature descriptors gives us a high accuracy of 98.1%, compared to 99.6% on ResNet18 trained on raw image pixels. The former only takes 3.8 seconds to train one epoch on a GPU while the latter takes 475.88 seconds, giving us a 125.23× speed up.

Abstract

Neural network models are increasingly becoming deeper, more capable, and larger, having more parameters. However, while they are able to solve challenging tasks with high accuracy, it is also becoming harder and harder to train these deep neural networks in terms of time and the computational resources needed. Our project aims to reduce the training time and computational resources needed, by training a neural network with Histogram of Oriented Gradients (HOG) feature descriptors [1] to detect the presence (or absence) of a pedestrian in a given image. We show that training a ResNet18 [3] on HOG feature descriptors have a 125× speed up compared to training on image pixels, while maintaining comparable accuracy at 98.1% compared to 99.6%.

1. Introduction

In the realm of computer vision, the detection of pedestrians within an image is a critical challenge with significant implications for various applications, ranging from autonomous vehicles to surveillance systems. This project aims to explore and evaluate different methodologies for pedestrian detection, focusing on their effectiveness and efficiency. The core of this investigation involves the development and comparison of four distinct classifiers:

1. A ResNet18 [3] with RGB image pixels as input
2. A ResNet18 with derived HOG feature descriptors as input
3. A 1-layer Convolutional Neural Network (CNN) with HOG feature descriptors as input
4. A fully-connected neural network with HOG feature descriptors as input

The first classifier is trained directly on the pixel values of RGB images. This approach leverages the raw visual data as input, hypothesizing that the color and intensity variations within these images can provide sufficient information for effective pedestrian detection.

In contrast, the second classifier is trained using Histogram of Gradients (HoG) feature descriptors. The HoG method focuses on the structure or the shape of objects within the image by capturing the direction and magnitude of brightness gradients, as illustrated in Figure 2. This approach is grounded in the theory that the outline and texture of pedestrians can be more distinctive and informative than their color information, especially in diverse lighting conditions.

The third classifier also takes HOG feature descriptors as input, but it consists of only one convolutional layer followed by a fully-connected layer. This classifier allows us to potentially visualize what the feature descriptor of a pedestrian might look like.

Finally, the fourth classifier only consists of one fully-connected layer. Convolutional neural networks such as the ResNet [3] and VGG [7] are made up of two steps: feature extraction and classification. Feature extraction usually consists of multiple convolutional layers and the classification step uses one or more fully-connected layers. We notice that the HOG feature descriptors themselves are already a type of feature extration, allowing us to skip the first step and go straight into the classification step. This motivated us to experiment with this classifier.

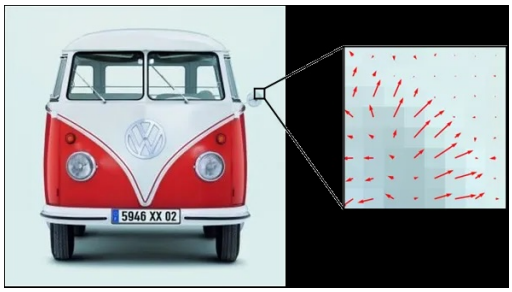

Figure 2. The gradient of each pixel used to calculate HOG feature descriptors, represented by direction and magnitude. [6]

The comparison of these four classifiers will be conducted through several key metrics: accuracy, precision, recall, and training time. Accuracy measures the overall effectiveness of the classifier in identifying pedestrians correctly. Precision evaluates how many of the identified "pedestrian" cases were actually pedestrians, while recall assesses how many of the actual pedestrians were correctly identified by the classifier. Training time is also a critical factor, as it reflects the practicality and efficiency of each method in real-world applications.

Through this project, we aim to gain insights into the strengths and limitations of each approach, providing a comprehensive understanding of their practicality in the field of pedestrian detection. This will not only contribute to the academic understanding of image processing techniques but also have practical implications in enhancing the performance of systems where pedestrian detection is a crucial component.

2. Background

2.1. Training Dataset

The "ROAD++" dataset [5], an extension of the original "ROAD" dataset, is utilized for this project. This dataset comprises around $55,000$ videos from the Waymo Open Dataset [8], each 20 seconds long, and spans various U.S. cities. It's a multi-label, multi-instance dataset with approximately 4.6 million detection bounding boxes, including 3.9 million agent labels, 4.3 million action labels, and 4.2 million location labels.

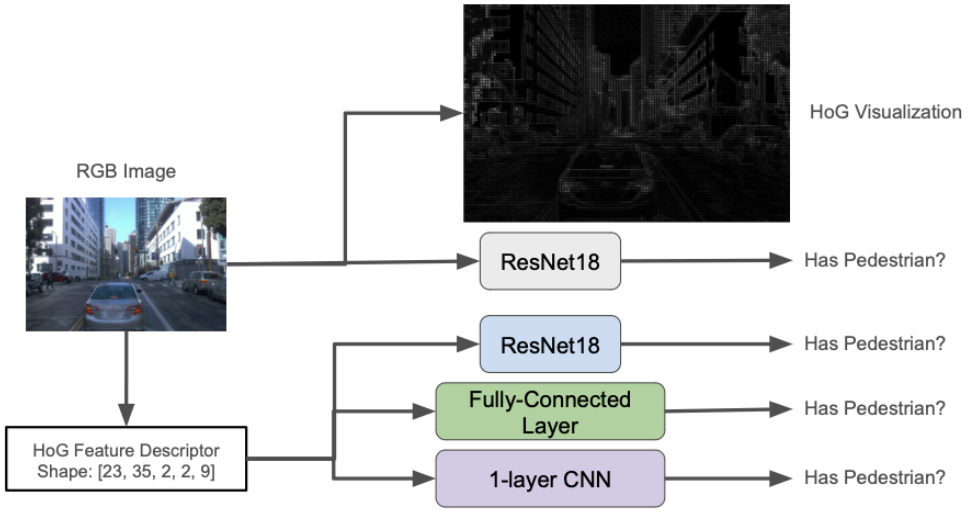

Figure 3. The schematic represents our pipeline

Figure 4. Sample images from the Road++ dataset [5]. The pedestrians can appear at many locations and in different scales. These images are always taken from the perspective of a car driving on the road, and span across a wide variety of weather and lighting conditions, as well as different urban and suburban areas.

Pedestrians are a key aspect of this dataset. Each road event in "ROAD++" is defined by agents, actions, and locations, with pedestrians being a significant category of agents. The dataset includes diverse pedestrian actions like walking, turning, or moving away, annotated with precise bounding boxes. This rich pedestrian-focused data makes "ROAD++" ideal for training models in pedestrian detection, providing a nuanced understanding of pedestrian behavior in urban settings.

Due to hardware limitations, we utilized a subset of the dataset comprising 100 videos, totaling around $18,000$ frames. This selection still ensures a rich and varied dataset, maintaining the focus on pedestrian data, which is essential for our project's objectives.

2.2. Histogram of Gradients

The Histogram of Gradients (HoG) is a key feature descriptor in the realm of computer vision, particularly in object detection tasks. It operates by analyzing the direction and magnitude of color intensity gradients within an image. This process effectively captures the contours and edges of objects, translating visual data into a numerical form that can be processed by machine learning models. HoG has been instrumental in enhancing the accuracy of object detection algorithms, particularly in complex visual environments.

In pedestrian detection, HoG descriptors are pivotal due to their ability to capture the shape and silhouette of a human figure. They are adept at differentiating between pedestrian and non-pedestrian elements in an image by examining the structure of gradients. This makes HoG particularly effective in environments where the color or texture of the background is similar to that of the pedestrians, as it relies more on form than color or texture contrasts.

2.3. Existing work

After receiving feedback on our initial project proposal, we were suggested to look at two existing works which leverage HOG in image networks. The first attempts to pretrain a network by having it learn to compute the HOG of an im-

```
image_paths = glob.glob("./road-waymo/rgb-images/*/*")
hog_paths = [path.replace(".jpg", ".npy") for path in image_paths]
hog_paths = [path.replace("rgb-images", "hog-images") for path in hog_paths]

for image_path in tqdm(image_paths, desc = "Extract HOG"):

    # reading the image
    img = imread(image_path)
    img = cv2.resize(img, (0, 0), fx = 0.3, fy = 0.3)

    # Extract HOG feature descriptor and visualization
    fd, hog_image = hog(img, orientations=9, pixels_per_cell=(16, 16),
                        cells_per_block=(2, 2), visualize=True, feature_vector=False, channel_axis=-1)

    # Rescale histogram for better display
    hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))

    # make path for hog image visualizations
    hog_image_path = image_path.replace("rgb-images", "hog-images")
    hog_image_dirname = os.path.dirname(hog_image_path)
    Path(hog_image_dirname).mkdir(parents=True, exist_ok = True)
    plt.imsave(hog_image_path, hog_image_rescaled, cmap = "gray")

    # make path for hog feature descriptors
    hog_fd_path = image_path.replace(".jpg", ".npy").replace("rgb-images", "hog-fd")
    hog_fd_dirname = os.path.dirname(hog_fd_path)
    Path(hog_fd_dirname).mkdir(parents=True, exist_ok = True)
    # print(fd.shape)
    np.save(hog_fd_path, fd)
```

Figure 5. Code used to extract HOG Features from the dataset

age by replicating hand-designed HOG features[4]. This method utilizes past computer vision knowledge to train deep networks that can be fine-tuned, and uses pedestrian detection as one of the evaluated downstream tasks. While this paper attempts to replicate HOG features using a deep network, it doesn't isolate the benefits and drawbacks of the HOG feature map itself, which we seek to analyze in this paper.

The second paper attempts to pretrain the network with a similar objective[9]. Rather than predicting labeled pairs of an image and its respective HOG feature-map, they perform a self-supervised pretraining by masking a part of the input image and predicting the feature map of the masked area. Similar to the first paper, they only consider which feature map used in pretraining gives the best downstream performance. Our aim rather is to examine the direct effects of using HOG in the pedestrian detection task.

3. Experiment Approach

In our research, we adopted a comprehensive approach to train neural networks for the task of object detection, particularly focusing on pedestrian detection in urban environments. The methodology encapsulates a rigorous and systematic approach to training neural networks for object detection tasks. It emphasizes the importance of accuracy, generalization, and robustness in model training, ensuring that the developed models are not only precise in their current environment but are also adaptable to new, unseen scenarios. This approach is instrumental in advancing the field of computer vision, particularly in applications involving pedestrian detection in urban settings.

3.1. HOG Gradient Extraction

We convert each image in the dataset into a HOG feature descriptor using the code shown in Figure 5, and save a copy of the HOG visualization (as shown in Figures 7 and 12) along side the RGB images and HOG feature descriptors themselves.

To generate the HOG feature descriptors, it follows the following steps:

1. Compute a gradient, which consists of magnitude and direction for each pixel of the image, as shown in Figure 2.
2. Divide image into cells, represented by the blue boxes in Figure 6. We use 16×16 for the size of our cells.
3. Group the cells into $Nx_{cell} \times Ny_{cell}$ blocks, represented by the red box in Figure 6. The blocks uses a sliding window approach (e.g. cell 1 and 2 form a block, then cell 2 and 3 form another block, etc). We use 2×2 blocks of 4 cells.
4. For each cell, compute a histogram of gradients with

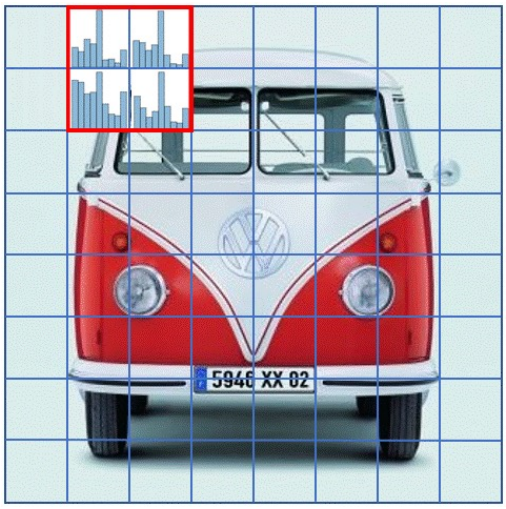

Figure 6. The image being grouped into cells (blue boxes) and cells being grouped into blocks (red box), and each cell having a histogram of gradient [6].

N_{bins} bins. We chose $N_{bins} = 9$ so that the width of each bin is 20 degrees, for a total of 180 degrees of gradient directions. The height of each bin in the histogram is the sum of all magnitudes within the bin width of 20 degrees.

5. Finally, normalize the cell histograms within each block.

A HOG feature descriptor is represented by a 5-dimensional feature vector: $Nx_{block} \times Ny_{block} \times Nx_{cell} \times Ny_{cell} \times N_{bins}$, where Nx_{block} and Ny_{block} are the number of blocks in each direction, Ny_{block} and Nx_{cell} are the number of cells in each block, and N_{bins} is the number of bins.

Our input images extracted from the "ROAD++" dataset [5] have 1920×1280 pixels. After HOG extraction, we get a HOG feature descriptor with size $79 \times 119 \times 2 \times 2 \times 9$. We observe that extracting HOG feature vector for every pixel in the image runs at only 2 fps, which is not ideal for real-time inference. We decide to resize the images by 1/3, speeding up the extraction to 5 fps. This gives us a final HOG feature descriptor of with size $23 \times 35 \times 2 \times 2 \times 9$.

3.2. Transforming HOG Feature Descriptor for ResNet18

ResNet18 is designed to be trained on raw image pixels that are usually $C \times H \times W$ vectors, where C is the number of channels (i.e 3 channels for RGB), and $H \times W$ are the height and width of the images. However, the HOG feature descriptor that we extracted are of shape $23 \times 35 \times 2 \times 2 \times 9$ as described in the section above. In order to use HOG feature descriptor as input to the ResNet18, we have to convert this 5 dimensional vector into the 3 dimensional vector. We decided the last 3 dimensions for number of cells and number of bins can be combined into the C dimension, resulting in $2 \times 2 \times 9 = 36$ channels. We then transpose vector to bring the C dimension to the front, giving us a $C \times H \times W = 36 \times 23 \times 35$ vector.

3.3. Training

The training process is initiated with a custom function, which takes the neural network model, training, and validation data loaders as inputs. This function is meticulously crafted to guide the model through several epochs of learning, balancing the optimization of weights and biases to enhance the model's predictive accuracy. At the core of this process is the use of CrossEntropyLoss, a standard loss function in classification tasks, which measures the performance of the model in classifying the input data correctly. The optimization of the model parameters is achieved through Stochastic Gradient Descent (SGD), a widely used optimization algorithm in machine learning that adjusts the model's parameters iteratively to minimize the loss function.

During each epoch of training, the model undergoes a forward pass where it makes predictions on the training data. Following this, a backward pass is conducted, where the gradients of the loss function with respect to the model parameters are computed and used to update these parameters. This iterative process helps in refining the model's ability to identify and classify objects within images accurately.

3.4. Validation

Validation plays a crucial role in this methodology. Post-training, the model is evaluated on a separate set of data not seen during the training phase. This evaluation is crucial for assessing the generalizability and robustness of the model. Metrics such as validation loss and accuracy are computed to gauge the model's performance. Additionally, precision and recall are derived from the confusion matrix, offering deeper insights into the model's efficacy in correctly identifying objects, particularly pedestrians.

The most effective model, as determined by its performance on the validation dataset, is preserved for subsequent use. This model saving step is crucial, as it allows for the retention of a version of the model that has demonstrated the highest proficiency in classifying objects within the given constraints of the dataset.

Figure 7. Visualization of HOG Feature Descriptors. The lines in the right image show the gradient directions with the highest magnitudes of each histogram.

Figure 8. HOG Visualization of pedestrians

4. Results

4.1. Runtime comparison

As part of our experiments, we gathered data on the model's runtime during both training and inference. At training time we measured the amount of time it took for one full pass through the entire dataset. At inference time we measured the average time it took for one input to be run. We hypothesized that the runtime would be inversely proportional to model complexity and input feature size. We expected the ResNet models to take longer than the perceptron model and 1-layer Conv network. We also expected models using the high dimensional raw RGB image inputs to take longer than the models which used HOG features as inputs. The experiment results are consistent with this hypothesis, with ResNet on RGB images taking the most time, followed by ResNet on HOG features, 1-layer Conv on HOG features, then the single layer perceptron on HOG features.

We see a drastic drop in runtime when switching from RGB inputs to HOG features as inputs, whereas we only see a small speedup when switching from ResNet to the simpler models. From this observation we draw the conclusion that most of the boost in runtime comes from using the HOG feature maps rather than using simpler models.

4.2. Performance comparison

Since we are benchmarking the classification task, we measure the performance using precision, recall, and accuracy. We enforce class balance in the labels, so accuracy is still informative on the model performance. We hypothesized that the performance would be proportional to the model compelxity and input feature complexity. We expected the ResNet models to perform better since they can capture more complex relations, and we expected models using the raw RGB image inputs to perform better since it has access to more information in the data without compression. Our experiment results are consistent with our hypothesis except one case. We originally expected the 1-layer Conv network to outperform the fully-connected network, but noted that the 1-layer Conv network actually has less parameters than the fully connected layer, and is reasonable that it could perform worse.

We see a minor decrease in performance when switching from RGB inputs to HOG features as inputs, whereas we see a larger dropoff when switching from ResNet to simpler models. From this observation we draw the conclusion that using HOG features incurs a smaller drop in performance compared to switching to a simpler model.

4.3. Performance-Runtime tradeoff

From our analyses of our results from the runtime and performance experiments, we can see that HOG feature maps do very well. When switching from RGB image inputs to HOG feature map inputs, we speed up the training time by over 100 times, while only incurring a 1.5% decrease in accuracy. We draw the conclusion that HOG feature maps are a highly effective way to reduce computing resources while

```python
# define model types
def resnet18_hog():
    model = torchvision.models.resnet18(weights = None)
    model.conv1 = torch.nn.Conv2d(2*2*9, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=
    False)
    model.fc = torch.nn.Linear(in_features=512, out_features=2, bias=True)
    return model

def linear_hog():
    model = nn.Sequential(
        nn.Flatten(),
        nn.Linear(28980, 2, bias=True)
    )
    return model

class conv1_hog(nn.Module):
    def __init__(self):
        super().__init__()

        self.model = nn.Sequential(
        nn.Conv2d(9, 1, kernel_size=(16, 8)),
        nn.Flatten(),
        nn.Linear(1953,2)
        )
    def forward(self,x):
        x = x.view(-1,9,46,70)
        return self.model(x)
```

Figure 9. Code showing our three model architectures that take HOG feature descriptors as input

Model	Time (s)	Precision	Recall	Accuracy (%)
RGB ResNet18	475.88	**1.0**	**0.99**	**99.6**
HoG ResNet18	3.80	0.98	0.97	98.1
HoG Fully-Connected	**2.48**	0.95	0.94	93.4
HoG 1-layer Conv	3.42	0.90	0.89	90.1

Figure 10. Comparing our four models. Results show that using HOG as input significantly reduces training time, while maintaining comparable accuracy.

maintaining comparable performance in the task of pedestrian detection.

4.4. Failure cases

We analyzed failure cases on the fully connected network over the HOG features. We present 2 samples from the false positives and 2 samples from the false negatives to examine. Within the false positive predictions, there is one image which actually contains a pedestrian on the sidewalk, indicating some error in the dataset labeling. In the second image we see there is a lot of glare and distortion in the image which possibly caused the error.

Within the false negatives, there is one image in the dark with very low lighting. It is hard to tell by looking at the image if there is a pedestrian, which likely caused the error. In the second image, it doesn't look like there are any pedestrians at all, which likely is another dataset labelling issue.

4.5. Visualized filters

One purpose of including the 1-layer conv network in our experiments was to determine the viability of the classical method of hand-designing a filter over the HOG features. We wanted to compare this technique to modern ones by searching for the optimal filter using backprop with a 1-layer Conv network, then applying the filter over the HOG features. One benefit of this technique is its interpretability, since we can examine the kernels themselves to see what the model is "seeing". Since there is one filter for each chanel, and there is one chanel for each orientation in the HOG, there are 9 total filters. By stacking these filters toether and treating the stacked filters as a HOG with 9 channels, we can visualize it as an HOG as shown in the figure below. Unfortunately, this resulting filter is not highly interpretable. We can make out a general shape which could be a pedestrian, but it is not clear. Though this result is not unexpected, as

Figure 11. False positive predictions

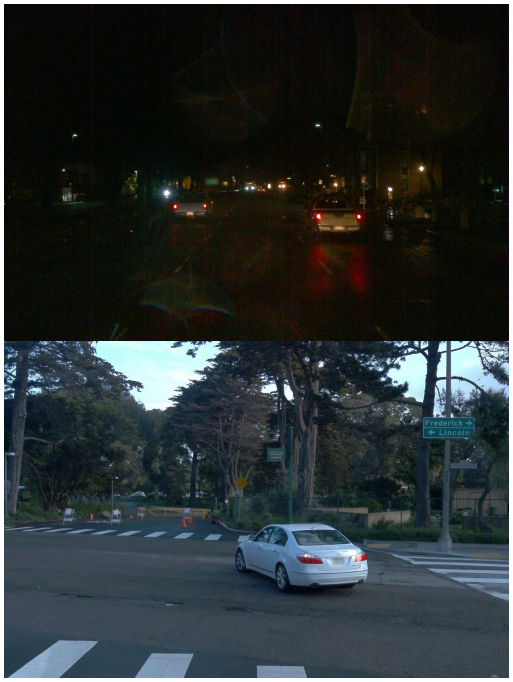

Figure 12. False negative predictions

this kernel operates on the HOG features, which themselves may not be as interpretable by our eyes compared to raw RGB images.

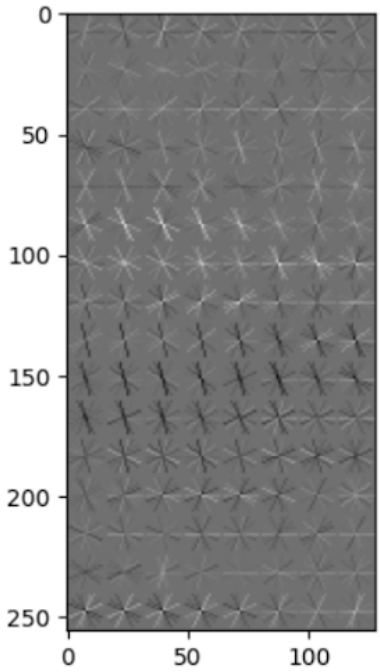

Figure 13. Visualization of 1-layer Conv network filters

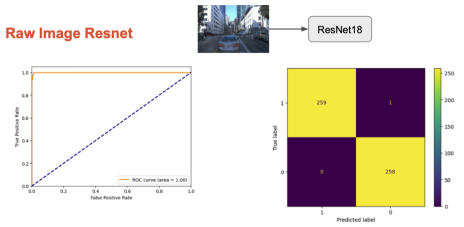

Figure 14. Raw Image Resnet

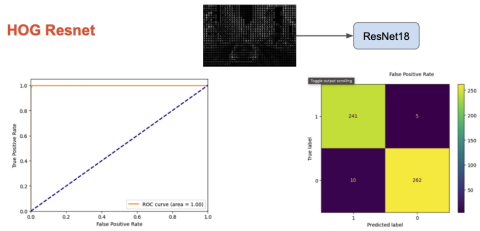

Figure 15. HoG Resnet

5. Discussion

5.1. Reduced Training Time

We showed that training on HOG feature descriptors have about 125× speed up compared to training on raw image pixels. This significant decrease in training time could mean that it takes much shorter time to iterate between different

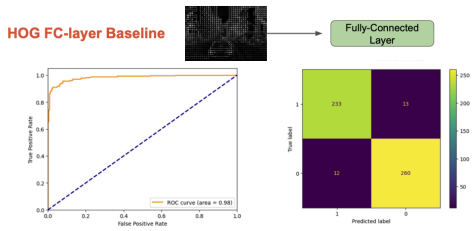

Figure 16. HoG Fully Connected CNN

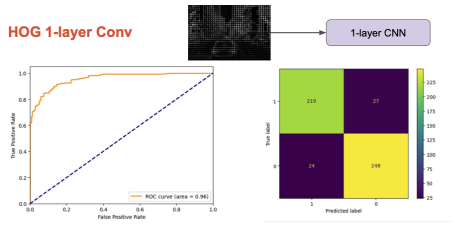

Figure 17. HoG 1-Layer CNN

models and hyper-parameters, allowing companies and research institutions to rapidly test and experiment with new ideas.

5.2. Savings on Computing Resources

The powerful GPUs required for training large neural network models are usually shared resources due to their high costs. Cloud computing allows researchers to borrow GPUs and they are usually priced per hour used. This means that the reduction in training time also means reduction in cost. Moreover, it allows researchers to use personal computers at home to train the models. We experimented with training our HOG ResNet18 model on a laptop *CPU* and per-epoch training time is only about 52 seconds, which is a $9\times$ speed up compared to training on RGB images with a *GPU*.

5.3. Real time applications

Real-time pedestrian detection is a crucial task for enabling safe autonomous vehicles. Not only does training time gets reduced, we also observe significant speed up during inference of the model. However, the HOG extraction becomes the bottleneck, running at only 5 fps during inference time. This is mainly due to the implementation of HOG being on a CPU as opposed to GPU, which can significantly speed up HOG extraction by computing the histogram of each cell in parallel.

5.4. Future Work

Our models are trained to detect whether a person exists in an image, which is consistent with the original HOG paper [1]. In the original paper, the dataset consists of images of human and non-human in which the images with human are all front and center. Our dataset, however, have pedestrians of different scale and locations in the images (as shown in Figure 4), making it harder to classify whether the image has a pedestrian or not.

In future work, we would like to train models that can perform bounding box detection, which provides scale and location of the pedestrian in the image. One approach to this is to utilize the network architecture of Faster R-CNN [2]. The architecture first uses convolutional layers to perform feature extraction, and then feed the feature extraction into a Region Proposal Network (RPN) that provide Regions of Interest (RoIs), which then get fed into another network for classification. We could skip the convolutional layers and feed HOG feature descriptors directly into the RPN and classification layers.

6. Individual Contributions

- Asher Mai: Downloaded dataset, wrote code to extract HOG feature descriptors and visualizations in big batches, and set up experiments for HOG ResNet and RGB ResNet.
- Matthew Tang: Set up experiments for 1-layer CNN, and fully-connected layer, ran all the experiments on desktop with GPU, and visualized the filter for 1-layer CNN.
- Sid Ahuja: experimented with different existing HOG feature extration implementations, did literature review for background and related work, wrote the Introduction, Background, and existing work sections of the paper.

References

[1] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 886–893 vol. 1, 2005. 1, 2, 9

[2] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 9

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2

[4] Ming-Yu Liu, Arun Mallya, Oncel Tuzel, and Xi Chen. Unsupervised network pretraining via encoding human design. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016. 4

[5] Will Maddern, Geoffrey Pascoe, Matthew Gadd, Dan Barnes, Brian Yeomans, and Paul Newman. Real-time kinematic ground truth for the oxford robotcar dataset. *arXiv preprint arXiv: 2002.10152*, 2020. Available at https://sites.google.com/view/road-plus-plus/dataset. 2, 3, 5

[6] Dahi Nemutlu. Hog feature descriptor, 2022. Available at https://medium.com/@dnemutlu/hog-feature-descriptor-263313c3b40d. 2, 5

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[8] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[9] Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training, 2023. 4