Implementing Object Detection Models for Reducing False Positives in Pedestrian Detection

Gurjot Singh
MASc. Software Engineering
Memorial University of Newfoundland
gurjots@mun.ca

Rushil Deshwal
MASc. Computer Engineering
Memorial University of Newfoundland
rdeshwal@mun.ca

Ehsan Salehin
MASc. Computer Engineering
Memorial University of Newfoundland
esalehin@mun.ca

***Abstract*—Advanced driver assistance systems (ADAS), driverless vehicles, and road safety all depend on pedestrian detection systems. But these systems frequently experience false positive detections, which can result in incorrect judgments and even safety hazards. Modern object detection models are combined with specific techniques for false positive reduction in the suggested strategy. The implemented models on publicly available datasets are thoroughly evaluated in the report.**

1. Introduction

The accuracy and dependability of automated systems must be ensured by implementing object detection models to decrease false positives in pedestrian recognition, especially when it comes to traffic safety and human-robot interactions. False positives, which happen when the model mistakenly considers an object to be a pedestrian when it isn't, can have detrimental effects like accidents or inefficient use of resources. Therefore, improving the performance of these systems requires creating reliable object detection models that efficiently reduce false positives.

One of the core tasks in computer vision is object detection, which is finding objects in pictures or movies. However, because of a number of issues such occlusions, changing stances, and congested situations, it is still difficult to achieve high object detection accuracy, especially for pedestrians. Developing advanced models that can accurately discern pedestrians from other objects and background features is necessary to address these issues.

Reducing false positives in pedestrian identification is even more important in the setting of smart cities, where monitoring pedestrian movements is key for guaranteeing safety and averting accidents. To improve traffic safety and human-robot interactions, it is possible to increase the overall efficiency and efficacy of automated systems by putting in place object detection models that can effectively recognize pedestrians while avoiding false positives.

2. Related work

A two-stage pedestrian detection algorithm is presented in one study, which combines a more intricate classification stage with a computationally efficient feature extraction technique. While the second stage uses a more complex classification method to reduce false positives, the first stage generates candidate pedestrian zones using a straightforward yet quick feature extraction method. The authors present a brand-new feature-based classification technique that makes use of local binary patterns (LBP) and histograms of oriented gradients (HOG). They test their strategy using the CityPersons dataset and show that, in comparison to a single-stage method, their methodology significantly reduces false positives while keeping a high detection rate [1].

According to a different study, in the context of autonomous driving, pedestrian identification is a crucial task requiring a high degree of accuracy and dependability. In order to reduce false positives in pedestrian detection, a cooperative communication system between automobiles and infrastructure is proposed in this work. The authors present a multi-sensor fusion method that integrates information from roadside devices and other cars with data from onboard sensors (such as cameras and LiDAR). The system can reduce false positives by better differentiating between pedestrians and other objects by sharing and integrating data from many sources [2].

Furthermore, a significant obstacle in pedestrian detection is the requirement for copious amounts of labeled data in order to train precise detectors. An innovative method for learning scene-specific pedestrian detectors without the need for actual data is presented in another work. The authors suggest a technique that uses 3D human models rendered in a variety of stances and environments to provide artificial training data. The system can adapt to different scenes and surroundings by training detectors on this synthetic data, which lowers the number of false positives brought on by environmental factors or scene-specific challenges [3].

Another work bridges the gap between detection and tracking to address the issue of false positives in pedestrian detection. In order to improve consistency between detected

items and tracked ones, the authors suggest a unified strategy that integrates tracking and detection into a single framework. Through the application of tracking limitations and temporal information, the system can lessen false positives that result from sporadic or inconsistent detections. Further lowering false positives, the unified methodology also makes it possible to employ tracking data to enhance and improve the detection results [4].

In one study, the authors present a technique that uses spatiotemporal consistency to cut down on false positives in pedestrian identification. The suggested method considers the geographical and temporal correlations among detections across several frames. The system is able to distinguish and eliminate false positives that do not display the anticipated spatiotemporal patterns of pedestrian mobility by examining the consistency of detections over time and space. Using a variety of datasets, the authors assess their approach and show a notable decrease in false positives while keeping high detection rates [5].

3. Methods

In this research, we tackle the crucial problem of false positives in pedestrian detection systems by utilizing the capability of deep learning and convolutional neural networks (CNNs). Because CNNs can learn hierarchical representations straight from raw data, they have shown impressive performance in a variety of computer vision tasks, including object detection. since of this ability, CNNs are a good fit for the pedestrian identification task since they can identify intricate patterns and characteristics linked to pedestrians while also differentiating them from other objects and background clutter. In particular, we use the modern deep learning model known as YOLO (You Only Look Once) object identification method, which has become quite popular because of its excellent accuracy, real-time performance, and effective end-to-end training process. YOLO is a great fit for our pedestrian detection challenge because of its innovative architecture, which treats object detection as a single regression problem and enables simultaneous object location and classification.

The TensorFlow deep learning framework, which offers an adaptable and effective environment for creating, training, and deploying neural networks, makes it easier to implement the YOLO technique. TensorFlow is the perfect tool for our study because of its wealth of documentation, vibrant community, and interoperability with a wide range of hardware accelerators, including GPUs and TPUs. Furthermore, we employ the popular computer vision package OpenCV for a number of image processing operations, including loading, resizing, and preprocessing. We are able to improve and assure consistent input to our YOLO model by streamlining our data preprocessing pipeline with OpenCV's vast capability and its efficient implementation, which includes hardware acceleration. A smooth workflow is made possible by the combination of TensorFlow and OpenCV, from data preprocessing to model training and deployment. Furthermore, we used the ResNet architecture as the foundation network for our YOLO models, utilizing its skip connections and residual connections to boost the representational capability of the models and optimize gradient flow during training.

We utilize the potent method of transfer learning to train our YOLO-based pedestrian detector. This entails initializing our model with pre-trained weights from a model that has been trained on a large-scale dataset of various objects and scenarios. Using this method, we can take advantage of the feature extractors and learnt representations from these pre-trained models, which have previously been extensively trained on a large amount of picture data to capture generic object patterns and low-level features. Compared to training the model from scratch, we can get faster convergence and possibly better performance by fine-tuning these pre-trained weights on our pedestrian detection dataset. Transfer learning allows the model to take advantage of the knowledge gathered from large-scale datasets and tailor it to the particular job at hand, which is especially useful in situations where the target dataset is limited or domain-specific.

3.1. Convolutional Neural Networks

Convolutional neural network (CNN) is a regularized type of feed-forward neural network that learns feature engineering by itself via filters (or kernel) optimization. Vanishing gradients and exploding gradients, seen during backpropagation in earlier neural networks, are prevented by using regularized weights over fewer connections. [6] [7] For example, for each neuron in the fully-connected layer, 10,000 weights would be required for processing an image sized 100×100 pixels. However, applying cascaded convolution (or cross-correlation) kernels, [8] [9] only 25 neurons are required to process 5x5-sized tiles. [10] [11] Higher-layer features are extracted from wider context windows, compared to lower-layer features.
They have applications in:

- image and video recognition, [12]
- recommender systems, [13]
- image classification,
- image segmentation,
- medical image analysis,
- natural language processing, [14]
- brain–computer interfaces, [15] and
- financial time series. [16]

3.1.1. Architecture. A convolutional neural network consists of an input layer, hidden layers and an output layer. In a convolutional neural network, the hidden layers include one or more layers that perform convolutions. Typically this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix. This product is usually the Frobenius inner product, and its activation function is commonly ReLU. As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization

layers. Here it should be noted how close a convolutional neural network is to a matched filter. [17]

In a CNN, the input is a tensor with shape:

(number of inputs) × (input height) × (input width) × (input channels)

After passing through a convolutional layer, the image becomes abstracted to a feature map, also called an activation map, with shape:

(number of inputs) × (feature map height) × (feature map width) × (feature map channels).

Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus.[21] Each convolutional neuron processes data only for its receptive field.

3.1.2. Distinguishing features. In the past, traditional multilayer perceptron (MLP) models were used for image recognition.[example needed] However, the full connectivity between nodes caused the curse of dimensionality, and was computationally intractable with higher-resolution images. A 1000×1000-pixel image with RGB color channels has 3 million weights per fully-connected neuron, which is too high to feasibly process efficiently at scale.

For example, in CIFAR-10, images are only of size 32×32×3 (32 wide, 32 high, 3 color channels), so a single fully connected neuron in the first hidden layer of a regular neural network would have 32*32*3 = 3,072 weights. A 200×200 image, however, would lead to neurons that have 200*200*3 = 120,000 weights.

Also, such network architecture does not take into account the spatial structure of data, treating input pixels which are far apart in the same way as pixels that are close together. This ignores locality of reference in data with a grid-topology (such as images), both computationally and semantically. Thus, full connectivity of neurons is wasteful for purposes such as image recognition that are dominated by spatially local input patterns.

Convolutional neural networks are variants of multilayer perceptrons, designed to emulate the behavior of a visual cortex. These models mitigate the challenges posed by the MLP architecture by exploiting the strong spatially local correlation present in natural images

3.1.3. Building blocks. A CNN architecture is formed by a stack of distinct layers that transform the input volume into an output volume (e.g. holding the class scores) through a differentiable function. A few distinct types of layers are commonly used. These are further discussed below.

Convolutional layer

The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the filter entries and the input, producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.

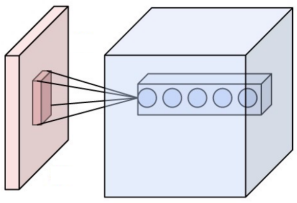


Figure 1. Neurons of a convolutional layer (blue), connected to their receptive field (red)

Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input. Each entry in an activation map use the same set of parameters that define the filter.

Local connectivity

When dealing with high-dimensional inputs such as images, it is impractical to connect neurons to all neurons in the previous volume because such a network architecture does not take the spatial structure of the data into account. Convolutional networks exploit spatially local correlation by enforcing a sparse local connectivity pattern between neurons of adjacent layers: each neuron is connected to only a small region of the input volume.

The extent of this connectivity is a hyperparameter called the receptive field of the neuron. The connections are local in space (along width and height), but always extend along the entire depth of the input volume. Such an architecture ensures that the learned (British English: learnt) filters produce the strongest response to a spatially local input pattern.

3.2. YOLO - You Only Look Once

What is YOLO: You Only Look Once (YOLO) is a state-of-the-art, real-time object detection algorithm introduced in 2015 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi in their famous research paper "You Only Look Once: Unified, Real-Time Object Detection".

The authors frame the object detection problem as a regression problem instead of a classification task by spatially separating bounding boxes and associating probabilities to each of the detected images using a single convolutional neural network (CNN).

By taking the Image Processing with Keras in Python course, you will be able to build Keras based deep neural networks for image classification tasks.

If you are more interested in Pytorch, Deep Learning with Pytorch will teach you about convolutional neural networks and how to use them to build much more powerful models. [18]

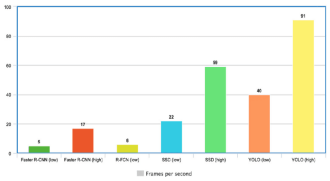


Figure 2. YOLO Speed compared to other state-of-the-art object detectors [19]

Some of the reasons why YOLO is leading the competition include its:

- Speed
- Detection accuracy
- Good generalization
- Open-source

1 **Speed**

YOLO is extremely fast because it does not deal with complex pipelines. It can process images at 45 Frames Per Second (FPS). In addition, YOLO reaches more than twice the mean Average Precision (mAP) compared to other real-time systems, which makes it a great candidate for real-time processing.

From the graphic below, we observe that YOLO is far beyond the other object detectors with 91 FPS.

2 **High detection accuracy**

YOLO is far beyond other state-of-the-art models in accuracy with very few background errors.

2 **Better generalization**

This is especially true for the new versions of YOLO, which will be discussed later in the article. With those advancements, YOLO pushed a little further by providing a better generalization for new domains, which makes it great for applications relying on fast and robust object detection.

For instance the Automatic Detection of Melanoma with Yolo Deep Convolutional Neural Networks paper [**?**] shows that the first version YOLOv1 has the lowest mean average precision for the automatic detection of melanoma disease, compared to YOLOv2 and YOLOv3.

2 **Open source**

Making YOLO open-source led the community to constantly improve the model. This is one of the reasons why YOLO has made so many improvements in such a limited time.

3.2.1. YOLO Architecture. YOLO architecture is similar to GoogleNet. As illustrated below, it has overall 24 convolutional layers, four max-pooling layers, and two fully connected layers.

The architecture works as follows:

- Resizes the input image into 448x448 before going through the convolutional network.

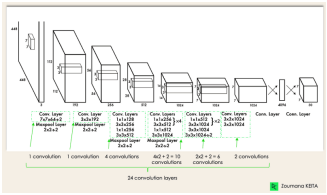
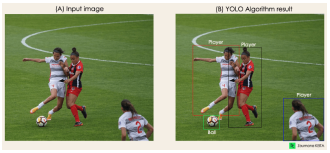


Figure 3. YOLO Architecture from the original paper [20]



- A 1x1 convolution is first applied to reduce the number of channels, which is then followed by a 3x3 convolution to generate a cuboidal output.
- The activation function under the hood is ReLU, except for the final layer, which uses a linear activation function.
- Some additional techniques, such as batch normalization and dropout, respectively regularize the model and prevent it from overfitting.

3.2.2. How Does YOLO Object Detection Work?. let's have a high-level overview of how the YOLO algorithm performs object detection using a simple use case. "Imagine you built a YOLO application that detects players and soccer balls from a given image. But how can you explain this process to someone, especially non-initiated people?

→ That is the whole point of this section. You will understand the whole process of how YOLO performs object detection; how to get image (B) from image (A)"

The algorithm works based on the following four approaches:

- Residual blocks
- Bounding box regression
- Intersection Over Unions or IOU for short
- Non-Maximum Suppression.

1 **Residual blocks**

This first step starts by dividing the original image (A) into NxN grid cells of equal shape, where N in our case is 4 shown on the image on the right. Each cell in the grid is responsible for localizing and predicting the class of the object that it covers, along with the probability/confidence value.

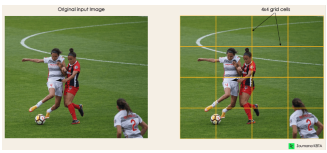


Figure 4. Residual blocks [20]

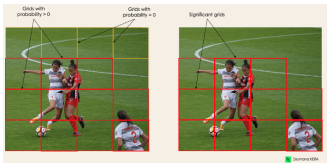


Figure 5. Bounding box regression [20]

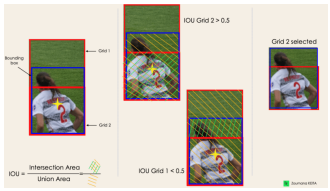


Figure 6. Intersection Over Unions or IOU [20]

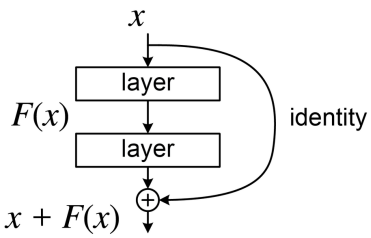


Figure 7. A Residual Block in a deep Residual Network. Here the Residual Connection skips two layers.

2 Bounding box regression

The next step is to determine the bounding boxes which correspond to rectangles highlighting all the objects in the image. We can have as many bounding boxes as there are objects within a given image.

YOLO determines the attributes of these bounding boxes using a single regression module in the following format, where Y is the final vector representation for each bounding box.

Y = [pc, bx, by, bh, bw, c1, c2]

This is especially important during the training phase of the model.

- pc corresponds to the probability score of the grid containing an object. For instance, all the grids in red will have a probability score higher than zero. The image on the right is the simplified version since the probability of each yellow cell is zero (insignificant).
- bx, by are the x and y coordinates of the center of the bounding box with respect to the enveloping grid cell.
- bh, bw correspond to the height and the width of the bounding box with respect to the enveloping grid cell.
- c1 and c2 correspond to the two classes Player and Ball. We can have as many classes as your use case requires.

3 Intersection Over Unions or IOU

Most of the time, a single object in an image can have multiple grid box candidates for prediction, even though not all of them are relevant. The goal of the IOU (a value between 0 and 1) is to discard such grid boxes to only keep those that are relevant. Here is the logic behind it:

- The user defines its IOU selection threshold, which can be, for instance, 0.5.
- Then YOLO computes the IOU of each grid cell which is the Intersection area divided by the Union Area.
- Finally, it ignores the prediction of the grid cells having an IOU $<=$ threshold and considers those with an IOU $>$ threshold.

4 Non-Max Suppression or NMS

Setting a threshold for the IOU is not always enough because an object can have multiple boxes with IOU beyond the threshold, and leaving all those boxes might include noise. Here is where we can use NMS to keep only the boxes with the highest probability score of detection.

3.3. ResNet

A residual neural network (also referred to as a residual network or ResNet) [21] is a seminal deep learning model in which the weight layers learn residual functions with reference to the layer inputs. It was developed in 2015 for image recognition and won that year's ImageNet Large Scale Visual Recognition Challenge (ILSVRC). [22] [23]

ResNet behaves like a highway network whose gates are opened through strongly positive bias weights. This enables deep learning models with tens or hundreds of layers to train easily and approach better accuracy when going deeper. The identity skip connections, often referred to as "residual connections", are also used in the original LSTM network, transformer models (e.g., BERT and GPT models such as ChatGPT), the AlphaGo Zero system, the AlphaStar system, and the AlphaFold system.

3.3.1. Formulation. Residual learning

In a multi-layer neural network model, consider a subnetwork with a certain number of stacked layers (e.g., 2 or 3). Denote the underlying function performed by this subnetwork as $H(x)$, where x is the input to the subnetwork. Residual learning re-parameterizes this subnetwork and lets the parameter layers represent a "residual function"

$$F(x) := H(x) - x$$

The output y of this subnetwork is then represented as: y = F(x) + x The operation of "$+x$" is implemented via a "skip connection" that performs an identity mapping, connecting the input of the subnetwork with its output. This connection is commonly referred to as a "residual connection" in subsequent studies. The function $F(x)$ is often constructed using matrix multiplication interspersed with activation functions and normalization operations (e.g., batch normalization or layer normalization). Collectively,

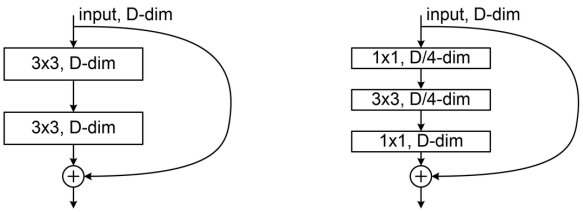


Figure 8. Two variants of convolutional Residual Blocks. [21] Left: a Basic Block that has two 3x3 convolutional layers. Right: a Bottleneck Block that has a 1x1 convolutional layer for dimension reduction (e.g., 1/4), a 3x3 convolutional layer, and another 1x1 convolutional layer for dimension restoration.

this configuration of components is known as a "residual block."

A deep residual network is constructed by simply stacking these blocks. Importantly, the underlying principle of residual blocks is similar to that of the original LSTM cell, a type of recurrent neural network. The LSTM cell predicts the output at time $t + 1$ as: $y_{t+1} = F(x_t) + x_t$ which simplifies to y = F(x) + x during backpropagation through time. [24]

3.3.2. Variants of residual blocks. A **Basic Block** is the simplest building block studied in the original ResNet. This block consists of two sequential 3x3 convolutional layers and a residual connection. The input and output dimensions of both layers are equal.

A **Bottleneck Block** [21] consists of three sequential convolutional layers and a residual connection. The first layer in this block is a 1x1 convolution for dimension reduction, e.g., to 1/4 of the input dimension; the second layer performs a 3x3 convolution; the last layer is another 1x1 convolution for dimension restoration. The models of ResNet-50, ResNet-101, and ResNet-152 in [21] are all based on Bottleneck Blocks.

4. Experiments

In the exploration of pedestrian detection and its challenges, particularly in mitigating false positives, my experimental approach focused on implementing and evaluating state-of-the-art object detection models. This study was inspired by the insightful findings of Karthika and Chandran (1970), which underscored the prevalence of false positives in pedestrian detection systems and emphasized the need for a well-structured dataset that distinguishes between humans and human-like objects. Leveraging a specialized dataset that encompasses persons and person-like objects (PnPLO), this project aimed to assess the efficacy of advanced machine learning models in enhancing detection precision.

The primary models selected for this study were Convolutional Neural Networks (CNN), You Only Look Once (YOLO), and Residual Networks (ResNet) configured for RetinaNet frameworks. These models were chosen for their proven capabilities in object detection tasks and their potential to reduce false positives effectively. The experimentation involved training each model using the PnPLO dataset, followed by rigorous testing to evaluate performance across varied scenarios, focusing on urban environments where pedestrian detection systems are most commonly deployed.

Each model underwent a series of evaluations to determine its accuracy and ability to differentiate between actual pedestrians and similar non-pedestrian objects. The performance metrics included precision, recall, and the F1-score, which provided a balanced view of model effectiveness. Comparative analysis was also conducted against existing pedestrian detection methods to benchmark advancements and identify any substantial improvements or lingering challenges.

The expected outcomes of this study were multifaceted. Firstly, the successful implementation and operational evaluation of each model demonstrated their practical utility in real-world scenarios. Secondly, the findings validated the effectiveness of these advanced models in significantly reducing false positives, which is crucial for the adoption of autonomous systems in public spaces. Lastly, the comprehensive performance evaluation offered insights into the strengths and limitations of each model, contributing valuable knowledge to the field of object detection and its application in pedestrian safety.

This project not only reaffirms the capabilities of CNN, YOLO, and ResNet in addressing critical issues in pedestrian detection but also enhances our understanding of how these advanced techniques can be adapted for more reliable and effective real-world applications.

4.1. YOLO Model Implementation

The YOLO (You Only Look Once) model is well-regarded for its speed and efficiency in real-time object detection tasks. In this project, we adapted the YOLOv5 architecture to enhance pedestrian detection capabilities, focusing specifically on minimizing false positives which are common when person-like objects are present in the visual field.

4.1.1. Dataset and Training. Our dataset was sourced from Kaggle and included a variety of images labeled not only with pedestrians but also with person-like objects such as statues, mannequins, scarecrows, and robots. This diversity is crucial as these objects often share visual characteristics with humans, leading to high rates of false positives in less sophisticated models. We created the model using a colab made by Roboflow which allowed us to make a model based on our training dataset.A.

Training using a custom configuration of YOLOv5 on Roboflow, provided a streamlined setup for model tuning and optimization. This process involved adjusting layers and parameters to specifically enhance the model's ability to discriminate between humans and person-like objects effectively.

4.1.2. Model Configuration and Training Process. The model was configured with a high confidence threshold of

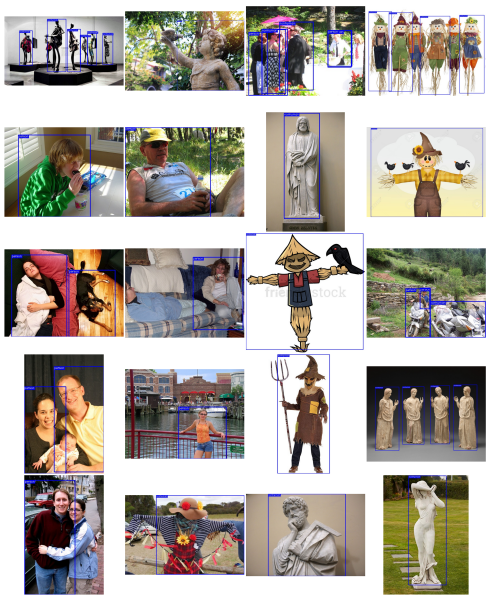


Figure 9. Compilation of YOLO on a random set of Val Images

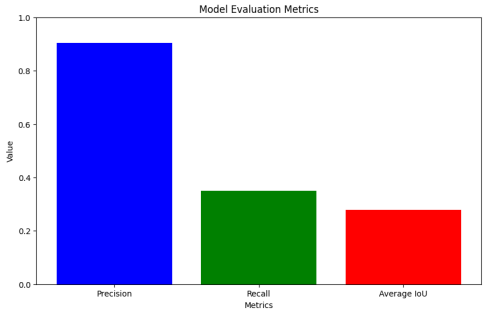


Figure 10. YOLO Metrics

80% and an overlap threshold of 40% to strictly manage what constitutes a detection. This strict configuration was crucial in reducing the number of false positives, a key goal for this project. The training process was monitored using Roboflow's built-in analytics to ensure that the model was learning appropriately and not overfitting or underfitting.

4.1.3. Testing and Validation. Testing and validation were conducted by applying the trained model to both seen and unseen images from our validation set. This was crucial to ensure that our model could generalize well to new data. The validation process involved randomly selecting images, which ensures that our performance metrics are representative of the model's capability in real-world scenarios.

4.1.4. Results and Metrics. The evaluation of the model was conducted using the standard metrics of precision, recall, and Intersection over Union (IoU). Precision of 90.43% indicates a high rate of true positive detections versus total detections, while the recall of 34.96% reflects the model's sensitivity in detecting all relevant instances. The average IoU of 27.82% provides insight into the average overlap between the predicted bounding boxes and the ground truth, indicating the accuracy of the location of predictions. These metrics demonstrate the model's effectiveness in accurately identifying true pedestrians and significantly reducing false positives.

4.1.5. Visualization. Visualizations of the model predictions were created to provide a direct comparison between the model's outputs and the actual annotations. This qualitative analysis is critical as it allows for visual confirmation of the quantitative metrics, providing a deeper understanding of model performance. Detailed visual examples of these predictions are included in the appendix, showing a variety of scenarios to highlight the model's capabilities and areas for improvement.

4.1.6. Conclusion. The customization and rigorous testing of the YOLO model for our specific dataset have led to significant advancements in reducing false positives in pedestrian detection. The detailed configuration and careful validation process have endowed the model with robustness and accuracy, crucial for applications in real-world scenarios. The outcomes not only support our initial hypothesis but also contribute meaningful insights into the capabilities and adaptability of advanced detection models in practical applications.

4.2. ResNet-based Object Detection

ResNet (Residual Network) is a deep convolutional neural network architecture known for its effectiveness in image classification tasks. In this experiment, we explored the utility of a pre-trained ResNet-based object detection model using the ImageAI library. The objective was to assess its performance in pedestrian detection, particularly in scenarios involving person-like objects (PnPLO).

4.2.1. Model Selection and Configuration. We opted for a pre-trained RetinaNet model based on the ResNet50 architecture, which is pre-trained on the COCO (Common Objects in Context) dataset. This choice was motivated by RetinaNet's ability to achieve high accuracy in object detection while maintaining efficiency in inference speed. The model was loaded and configured with specific parameters to detect objects labeled as "person" with a minimum confidence threshold set at 30%.

4.2.2. Dataset Preparation. Our dataset comprised images labeled with pedestrians and various person-like objects such as statues, mannequins, scarecrows, and robots. These objects introduce challenges in pedestrian detection due to their visual similarity to humans, leading to potential false positives. Annotations followed the PASCAL VOC format, enabling compatibility with the ResNet-based model.

4.2.3. Detection and Evaluation. We employed the ResNet-based object detection model to perform inference

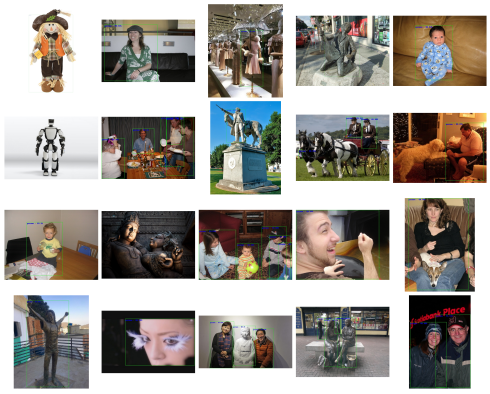


Figure 11. Compilation of ResNet on a random set of Val Images

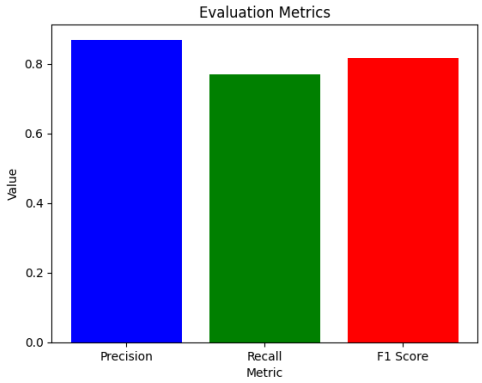


Figure 12. Resnet Metrics

on images from our dataset. Detection results were evaluated against ground truth annotations to calculate performance metrics, including precision, recall, and F1 score. The evaluation focused on assessing the model's ability to accurately identify pedestrians while minimizing false positives and false negatives.

4.2.4. Results and Analysis. Overall precision, recall, and F1 score were computed based on the aggregate performance across all tested images. The ResNet-based model achieved an overall precision of **86.91%**, recall of **76.97%**, and F1 score of **81.64%**. These metrics indicate the model's effectiveness in detecting pedestrians and differentiating them from person-like objects. Qualitative analysis through visual inspection of detection outputs provided additional insights into the model's performance under various conditions.

4.2.5. Visualization. Visualizations of detection outputs were generated to provide a visual representation of the model's performance. Collages comprising sample images and their corresponding detection results were created to demonstrate the model's capability in detecting pedestrians amidst complex scenes containing person-like objects.

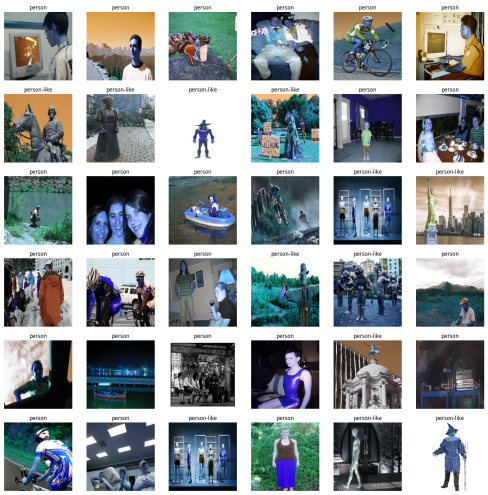


Figure 13. CNN Results

4.2.6. Discussion. The experiment results demonstrate the applicability of the ResNet-based object detection model in pedestrian detection tasks, even in scenarios with person-like objects. The achieved precision, recall, and F1 score validate the model's effectiveness and robustness in accurately identifying pedestrians while mitigating false detections. However, further optimization and fine-tuning may be necessary to address specific challenges encountered in real-world deployment scenarios.

4.3. CNN

The visual examination of the training images demonstrated that our dataset contains a wide range of examples for both 'person' and 'person-like objects.' The complexity and variety observed within the images (as seen in Figure X) suggest that the model is exposed to a comprehensive set of features necessary to learn the distinction between the two classes effectively. This diversity is expected to contribute to the model's ability to generalize well to new, unseen data

The training process highlighted the model's ability to learn and make accurate predictions on the training set, as evidenced by the steady increase in training accuracy and decrease in training loss. Nonetheless, the divergence between training and validation accuracy, along with the rise in validation loss after the third epoch, may indicate that the model is beginning to overfit. To combat this, future work could explore the implementation of regularization techniques, adjustments to the model's complexity, or data augmentation strategies to enhance the model's generalization capabilities on unseen data.

In the validation phase, the CNN's predictions were visually inspected for a random subset of the validation set, as shown in a collection of 36 images (Figure Y). Each image was automatically titled with the predicted class based on the highest probability from the model's output. The displayed images include both correct classifications

Figure 14. CNN Results

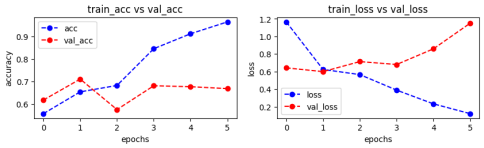
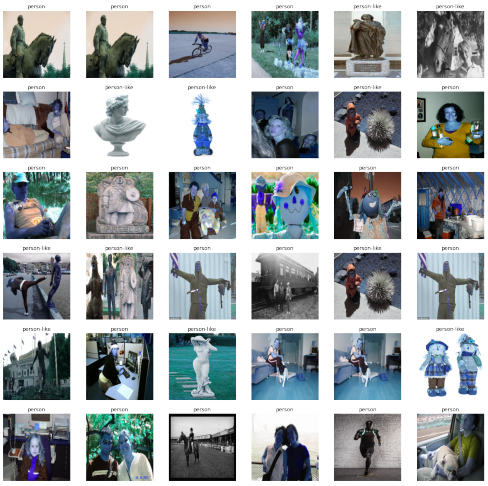


Figure 15. CNN Metrics

and misclassifications, reflecting the model's capability to discern between 'person' and 'person-like' objects. The titles reveal the model's decision-making process, providing an opportunity for qualitative analysis to complement quantitative metrics.

5. Conclusion

This project's objective was to implement and critically assess the effectiveness of several state-of-the-art object detection models in minimizing false positives in pedestrian detection tasks. Specifically, we evaluated Convolutional Neural Networks (CNN), You Only Look Once (YOLO), and Residual Networks (ResNet) using a comprehensive dataset that included both pedestrians and person-like objects.



- The YOLO model demonstrated impressive precision (*0.904*), which indicates a strong capability to correctly identify true positives. However, its recall (*0.350*) and Average Intersection over Union (IoU) (*0.278*) suggest limitations in detecting all relevant objects within the dataset.
- The ResNet model provided a balanced performance with a high overall precision (*0.869*), recall (*0.770*), and F1 Score (*0.816*), suggesting a robust capacity for accurately detecting pedestrians while effectively minimizing false positives.
- The CNN model, as shown in the attached results, achieved a performance indicative of its ability to learn features and generalize to the test data effectively, as evidenced by its accuracy and loss metrics over multiple epochs.

From the analysis of the training accuracy and validation accuracy curves, we observed that the models improved their detection capabilities as the number of epochs increased. However, the validation loss and accuracy indicated potential overfitting, suggesting a need for improved regularization strategies.

These results substantiate our project's aim by confirming that advanced object detection models can significantly reduce false positives in pedestrian detection systems. The performance of each model highlights their strengths and areas for improvement, offering essential insights for further enhancements in real-world applications.

In conclusion, our research contributes to the field of computer vision by providing evidence of the efficacy of these models in distinguishing pedestrians from person-like objects, thereby enhancing the safety and reliability of intelligent systems such as autonomous vehicles and surveillance networks.

References

[1] D. Schulz and C. A. Perez, "Two-stage pedestrian detection model using a new classification head for domain generalization," *Sensors*, vol. 23, no. 23, p. 9380, Nov. 2023.

[2] J. Cao, Y. Pang, J. Xie, F. S. Khan, and L. Shao, "From handcrafted to deep features for pedestrian detection: A survey," 2020.

[3] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. Miami, FL: IEEE, Jun. 2009, pp. 304–311.

[4] Viola, Jones, and Snow, "Detecting pedestrians using patterns of motion and appearance," in *Proceedings Ninth IEEE International Conference on Computer Vision*. Nice, France: IEEE, 2003, pp. 734–741 vol.2.

[5] S. Munder, C. Schnorr, and D. M. Gavrila, "Pedestrian detection and tracking using a mixture of view-based shape–texture models," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 2, pp. 333–343, Jun. 2008.

[6] R. Venkatesan and B. Li, *Convolutional Neural Networks in Visual Computing: A Concise Guide*. CRC Press, 2017, archived from the original on 2023-10-16. Retrieved 2020-12-13.

[7] V. E. Balas, R. Kumar, and R. Srivastava, *Recent Trends and Advances in Artificial Intelligence and Internet of Things*. Springer Nature, 2019, archived from the original on 2023-10-16. Retrieved 2020-12-13.

[8] Y. Zhang, H. G. Soon, D. Ye, J. Y. H. Fuh, and K. Zhu, "Powder-bed fusion process monitoring by machine vision with hybrid convolutional neural networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 5769–5779, 2020, archived from the original on 2023-07-31. Retrieved 2023-08-12.

[9] N. I. Chervyakov, P. A. Lyakhov, M. A. Deryabin, N. N. Nagornov, M. V. Valueva, and G. V. Valuev, "Residue number system-based solution for reducing the hardware cost of a convolutional neural network," *Neurocomputing*, vol. 407, pp. 439–453, 2020, archived from the original on 2023-06-29. Retrieved 2023-08-12.

[10] H. H. Aghdam and E. J. Heravi, *Guide to convolutional neural networks: a practical application to traffic-sign detection and classification*. Springer, 2017.

[11] Atlas, Homma, and Marks, "An artificial neural network for spatio-temporal bipolar patterns: Application to phoneme classification," in *Neural Information Processing Systems (NIPS 1987)*, vol. 1, 1987, archived (PDF) from the original on 2021-04-14.

[12] M. V. Valueva, N. N. Nagornov, P. A. Lyakhov, G. V. Valuev, and N. I. Chervyakov, "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation," *Mathematics and Computers in Simulation*, vol. 177, pp. 232–243, 2020, s2CID 218955622.

[13] A. van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 2643–2651, archived from the original (PDF) on 2022-03-07. Retrieved 2022-03-31.

[14] R. Collobert and J. Weston, "A unified architecture for natural language processing," in *Proceedings of the 25th International Conference on Machine Learning - ICML '08*. New York, NY, US: ACM, 2008, pp. 160–167, s2CID 2617020.

[15] O. Avilov, S. Rimbert, A. Popov, and L. Bougrain, "Deep learning techniques to improve intraoperative awareness detection from electroencephalographic signals," in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, vol. 2020. Montreal, QC, Canada: IEEE, 2020, pp. 142–145, archived from the original (PDF) on 2022-05-19. Retrieved 2023-07-21.

[16] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Forecasting stock prices from the limit order book using convolutional neural networks," in *2017 IEEE 19th Conference on Business Informatics (CBI)*. Thessaloniki, Greece: IEEE, 2017, pp. 7–12.

[17] L. Stankovic and D. Mandic, "Convolutional neural networks demystified: A matched filtering perspective based tutorial," 2022.

[18] DataCamp, "Yolo: Object detection explained," 2023.

[19] Y. Nie, P. Sommella, M. O'Nils, C. Liguori, and J. Lundgren, "Automatic detection of melanoma with yolo deep convolutional neural networks," 11 2019.

[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[22] "Ilsvrc2015 results," https://image-net.org/challenges/LSVRC/2015/results.php, accessed: 2023-04-19.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. [Online]. Available: https://ieeexplore.ieee.org/document/5206848

[24] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," 2016. [Online]. Available: https://arxiv.org/abs/1602.07261

Appendix

The following are links to the Google Colab notebooks used for the project development and the YOLO model training.

Project Notebook

This notebook contains the code and experiments for the main project.

- My Project Notebook on Google Colab

YOLO Model Training Notebook

This notebook illustrates the model training process using the Roboflow platform.

- Roboflow YOLO Training Notebook on Google Colab