JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 35 (2025) 202–208
Received: 16 February 2025
Accepted: 02 April 2025

Analysis the efficiency of object detection in images using machine learning libraries in Python

Analiza efektywności wykrywania obiektów na obrazach z zastosowaniem bibliotek uczenia maszynowego w języku Python

Patryk Kalita*, Marek Miłosz

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The purpose of this paper is to analyze and compare the accuracy of object detection in images using Python machine learning libraries such as PyTorch and Tensorflow. The paper describes the use of both libraries to train and test object detection models, considering architectures such as SSD and Faster R-CNN. The experiment was conducted on the Pascal VOC dataset to evaluate the effectiveness and performance of the models. The results include a comparison of metrics such as recall, precision and mAP which allows to choose the best solutions depending on the situation. The article concludes with a summary and final conclusions, allowing practical recommendations to be made for those working on object detection projects.

Keywords: python; machine learning; object detection

Streszczenie

Celem artykułu jest dokonanie analizy i porównanie dokładności wykrywania obiektów na zdjęciach przy pomocy bibliotek uczenia maszynowego w języku Python, takich jak: PyTorch i Tensorflow. W pracy opisano wykorzystanie obu bibliotek do trenowania i testowania modeli detekcji obiektów, uwzględniając takie architektury jak SSD i Faster R-CNN. Eksperyment został przeprowadzony na zestawie danych Pascal VOC, aby ocenić skuteczność i wydajność modeli. Wyniki obejmują porównanie takich metryk jak: recall, precision i mAP co pozwala na wybranie najlepszych rozwiązań w zależności od sytuacji. Artykuł kończy się podsumowaniem i końcowymi wnioskami, pozwalającymi dokonać praktycznych rekomendacji dla osób pracujących nad projektami związanymi z detekcją obiektów.

Słowa kluczowe: python; uczenie maszynowe; detekcja obiektów

*Corresponding author

Email address: patryk.kalita@pollub.edu.pl (P. Kalita)

Published under Creative Common License (CC BY 4.0 Int.)

1. Introduction

In the field of computer vision, one of the key aspects is the detection of objects in images. This has broad applications in issues such as autonomous vehicles or surveillance systems. Thanks to significant advances in machine learning, the development of these fields has become much simpler. Two platforms: PyTorch and Tensorflow have played a key role in the development of deep neural network models in the context of object detection.

The two solutions have many similarities, but each has its own unique characteristics that are critical when choosing a technology for a given problem. PyTorch exhibits a flexible architecture and intuitive approach, and is easy to prototype and debug. Tensorflow, is developed by Google, excels in production support and high scalability, making it popular in the industry.

This paper compares and analyzes the accuracy of object detection in images using models based on PyTorch and Tensorflow libraries. The purpose of this paper is to compare the effectiveness and performance of Faster R-CNN (Faster Region-based Convolutional Neural Network) and SSD (Single Shot MultiBox Detector) algorithms, which are among the most widely used solutions in the aspect of object detection. The advantages and disadvantages of each model will be presented in the context of implementation, allowing to make the most optimal choice depending on the needs and requirements of the user. The metrics that will be considered are recall, precision and mAP (Mean Average Precision). The work aims to fill the gap in comparative analyses of the differences between the mentioned libraries and their impact on the performance and quality of object detection systems.

2. Literature review

Object detection in images has recently become a key aspect in the research field of computer vision. The Tensorflow and PyTorch libraries enable the implementation of advanced models, such as Faster R-CNN and SSD. The following is a literature review of the mentioned solutions.

Some of the most widely used libraries in the implementation of object detection models are TensorFlow and PyTorch precisely. TensorFlow, offers a wide range of tools for optimization of large data sets, making it a reasonable choice for production applications [1]. PyTorch, on the other hand, features dynamic computational graphs, making it a frequent choice for in scientific research [2]. Each platform has its advantages. Tensorflow is efficient in computing distributed, but PyTorch performs better if we are talking about rapid prototyping [3].

Popular models for object detection in images are SSD and Faster R-CNN, but they have different architectures. SSD allows for fast, real-time analysis of images by using a single-phase approach. Liu's work [4] shows that the SSD model for large data sets, exhibits high accuracy while minimizing computational complexity. On the other hand, Faster R-CNN using a two-step strategy, based on generating regions of interest, which allows for more precise detection, especially on complex scenes. Howal and his associates have conducted research using this solution in autonomous vehicles [5]. As it turned out, this was a very good choice. Ren and co-authors [6] highlighted that the Faster R-CNN performs better and achieves better results for small objects due to more precise localization of their boundaries.

A factor that is often considered when comparing the two models is speed and accuracy. In a study conducted by Huang and co-authors [7] showed that Faster R-CNN is more efficient for detection of small and complex objects, while SSD is a suitable choice for systems requiring low latency, which includes real-time monitoring [8]. In both approaches, a transfer learning technique can be applied using ResNet models [9], which allows for improved performance.

Both platforms, Tensorflow and PyTorch, offer robust tools for solutions based on object detection objects. The SSD and Faster R-CNN models have many differences in terms of approach and efficiency, which determines the choice depending on the scenario. The next direction for research in this area should focus on optimizing current solutions and adapting them for systems with limited computational resources.

3. Purpose of research

The purpose of this research is to analyze and compare the accuracy of object detection in images using SSD and Faster R-CNN models, made using the using TensorFlow and PyTorch libraries. This study is designed to evaluate the impact of choosing of a given solution on detection performance on different data sets and under different computational conditions, as well as to identify the key factors affecting the quality of results and runtime. The results obtained may influence a better understanding of the applications and limitations of the technologies used in computer vision systems.

4. Research method

The object of this research is to analyze the accuracy of object detection in images using machine learning libraries in Python. The main focus is on evaluating the effectiveness and accuracy of the libraries and their algorithms in identifying and locating objects in images. The research aims to understand which tool offers the best results in the context of different types of data and model parameters. The goal, too, is to isolate the advantages and disadvantages of each solution.

4.1. Object and scope of research

The subjects of the research are machine learning libraries and their object detection models in Python. In particular, the following libraries and models will be studied TensorFlow, PyTorch, SSD and Faster R-CNN.

The scope of the study includes few steps. Selection and preparation of an appropriate dataset (Pascal VOC). Implementation and training of models for object detection (e.g. SSD, Faster R-CNN) using selected libraries. Evaluating the accuracy of models using specific metrics (Precision, Recall, mAP). Analyze the results and compare the performance of the libraries.

4.2. Research bench

The tests were conducted on computer hardware with the appropriate specifications (Table 1).

Table 1: Specification of equipment – research bench

Processor	Intel Core i5-8300H
Graphics	NVIDIA GeForce GTX 1050
Memory	24 GB DDR5
Mass memory	512 GB
Operating System	Windows 11

5. Formulae

The SSD model has a simple architecture and demonstrates computational efficiency, enabling real-time object detection. The SSD enables simultaneous class prediction and object localization in a single network run, eliminating the need to use complex region generation steps.

5.1. SSD loss function

The loss function formula for the SSD model is as follows:

$$L(x,c,l,g) = \frac{1}{N}\left(L_{conf}(x,c) + \alpha L_{loc}(x,l,g)\right) \quad (1)$$

where $L_{conf}(x,c)$ is the loss of classification as measured by cross entropy, $L_{loc}(x,l,g)$ is the loss of location, it is calculated as the sum of the distance between the predicted and actual coordinates of the frame, x indicates whether the window proposal is matched to the actual object, a value of 1 indicates a match, while 0 indicates no match, c is a prediction about the class of the object by the model, the predicted bounding box coordinates of a given object are denoted by l, while g is the actual coordinates of the bounding box, α is a factor that balances the two parts of the loss, N is the number of positive matches.

The SSD accurately performs object detection, but its performance decreases when detecting small-sized objects [10].

5.2. Faster R-CNN loss function

The Faster R-CNN model is a more advanced solution, combining region proposal generation using RPN (Region Proposal Network) and frame location classification and regression in a single network flow.

This architecture works well if high precision is desired, but the cost is higher computational complexity. The solution is represented by the following formula:

$$L(p,u,t,v) = \frac{1}{N_{cls}}\sum L_{cls}(p,u) + \leftthreetimes \frac{1}{N_{reg}}\sum[u \geq 1]L_{reg}(t,v) \quad (2)$$

where $L_{cls}(p,u)$ is the loss of classification as measured by cross entropy, $L_{reg}(t,v)$ is the regression loss. It often takes the form of a Smooth L1 function (3), p is the probability that the anchor contains an object, u is the actual class label, it reaches 0 for the background and 1 for the object, t predicts the offset of the bounding box, while v determines the actual offset and $\leftthreetimes$ is the coefficient governing the validity of the regression loss. The following formula improves stability by reducing outliers:

$$L_{Smooth\,L1}(x) = \begin{cases} 0.5x^2, & |x| < 1 \\ |x| - 0.5, & |x| \geq 1 \end{cases} \quad (3)$$

Faster R-CNN is more effective at detecting irregularly shaped objects and complex scenes, making it a good choice for solutions requiring high accuracy.

The metrics that will be taken when evaluating the effectiveness of object detection models are precision, recall and Mean Average Precision (mAP). These metrics are essential when measuring detection quality.

5.3. Precision and recall

Precision (4) and recall (5) are metrics that measure, respectively, the ratio of correctly detected objects relative to all detected (precision) and the ratio of correctly detected objects to all actual objects (recall):

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

$$Recall = \frac{TP}{TP + FN} \quad (5)$$

where TP is the number of real positives, FP is the number of false positives, FN is the number of false negatives.

5.4. Mean Average Precision (mAP)

The mAP metric is used to compare object detection models and the average precision for different classes is calculated:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \quad (6)$$

where N is the number of classes, APi is the average precision for class i. It is calculated based on the area under the precision curve relative to the recall value.

6. Test results

Tests were conducted using implemented models. The testing consisted of implementing code that calculated the recall, precision and mAP metrics for each of the models created. Each model was implemented and trained using the Pascal VOC 2012 dataset, with each model trained on 10 epochs. Two separate datasets were used for training, split into a training set and a validation set. In total, the two sets contain 11530 images. The training dataset has 5717 images, while the validation dataset has 5813 images. The Pascal VOC 2012 dataset has 20 object classes to include: background, aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, table, dog, horse, motorcycle, person, potted plant, sheep, sofa, train and tvmonitor. Before training, the data was preprocessed through images from PIL (Python Imaging Library) format into so-called tensors. A mapping was also applied of class names into identifiers.

The following implementations were used in the study:

- SSD model for the PyTorch library,
- Faster R-CNN model for the PyTorch library,
- SSD model for the TensorFlow library,
- Faster R-CNN model for the TensorFlow library.

The tests were carried out in the Google Colab environment for the indicated computer sprint (Table 1). To speed up the training, the GPU T4 execution environment type was used. In each case, a pre-trained model based on the VGG16 architecture was used for training. The models were trained according to a single configuration. The learning rate was set to 0.001 in the optimizer settings, which means smaller changes in the weights of each update. This results in more stable and slower training. The Momentum parameter has been set to a value of 0.9 which allows "smoothing" of the weight updates. It helps the optimizer to preserve the direction of movement in the parameter space, which avoids local minima and oscillations. Setting the Weight Decay parameter to a value of 5e-4 nullifies model overfitting by adding penalties to the loss function for excessive weight values. The models were trained on 50 epochs, and the data were loaded in batch with a size equal to 8. The results of the study were presented individually for each class of the Pascal VOC dataset.

6.1. PyTorch library

The evaluation was based on precision metrics for each object category (Table 2, Table 3). The Faster RCNN model achieves a mAP value of 0.69. SSD, on the other hand, does slightly better, achieving a score of 0.72. The model is slightly more precise which is not very significant. The Faster R-CNN performs better in categories such as bird, bottle and dog then the SSD model. However, the latter model performs better if we consider the categories: car, chair or tv. The SSD achieves a significant difference in detecting objects in the table and plant categories, reaching 0.61 and 0.46 respectively, compared to 0.55 and 0.4 obtained by the competing model. The values of the metrics are calculated in the range from 0 to 1. The higher the value, the higher the percentage of detected detections. Choosing the right solution will be important when developing solutions for systems that rely on detection of a specific object. This is especially true of the categories mentioned earlier. In this case, the choice of solution matters.

Table 2: Precision and recall metrics values for the PyTorch library SSD model

Class of the Pascal VOC dataset	Precision	Recall
aeroplane	0.85	0.99
bicycle	0.80	0.93
bird	0.70	0.95
boat	0.56	0.83
bottle	0.45	0.74
bus	0.79	0.95
car	0.76	0.90
cat	0.89	0.98
chair	0.53	0.98
cow	0.77	0.94
table	0.61	0.89
dog	0.83	0.93
horse	0.83	0.91
motorbike	0.81	0.90
person	0.79	0.98
potted plant	0.46	0.91
sheep	0.76	0.76
sofa	0.68	0.98
train	0.82	0.93
tvmonitor	0.67	0.95

Table 3: Precision and recall metrics values for the PyTorch library Faster R-CNN model

Class of the Pascal VOC dataset	Precision	Recall
aeroplane	0.84	0.97
bicycle	0.80	0.90
bird	0.74	0.96
boat	0.52	0.84
bottle	0.49	0.71
bus	0.78	0.97
car	0.76	0.89
cat	0.87	0.99
chair	0.45	0.99
cow	0.77	0.92
table	0.55	0.87
dog	0.85	0.91
horse	0.81	0.92
motorbike	0.80	0.91
person	0.79	0.99
potted plant	0.40	0.84
sheep	0.72	0.77
sofa	0.60	0.96
train	0.81	0.92
tvmonitor	0.61	0.95

6.2. TensorFlow library

In this case, the TensorFlow machine learning library was used for implementation. As with the PyTorch library, both precision and recall metrics were used for evaluation. The results are also presented separately for each class of Pascal VOC collection.

Table 4: Precision and recall metrics values for the TensorFlow library SSD model

Class of the Pascal VOC dataset	Precision	Recall
aeroplane	0.84	0.95
bicycle	0.83	0.84
bird	0.68	0.89
boat	0.58	0.74
bottle	0.46	0.79
bus	0.78	0.85
car	0.78	0.92
cat	0.88	0.96
chair	0.54	0.94
cow	0.74	0.93
table	0.63	0.88
dog	0.82	0.87
horse	0.83	0.90
motorbike	0.79	0.96
person	0.81	0.99
potted plant	0.45	0.86
sheep	0.78	0.80
sofa	0.67	0.97
train	0.83	0.93
tvmonitor	0.66	0.94

Table 5: Precision and recall metrics values for the TensorFlow library Faster R-CNN model

Class of the Pascal VOC dataset	Precision	Recall
aeroplane	0.82	0.93
bicycle	0.83	0.81
bird	0.73	0.91
boat	0.54	0.75
bottle	0.48	0.79
bus	0.79	0.89
car	0.73	0.94
cat	0.87	0.95
chair	0.48	0.94
cow	0.76	0.89
table	0.56	0.87
dog	0.83	0.88
horse	0.82	0.90
motorbike	0.80	0.94
person	0.82	0.99
potted plant	0.42	0.79
sheep	0.69	0.81
sofa	0.61	0.97
train	0.79	0.92
tvmonitor	0.63	0.93

The models achieved very similar results (Table 4, Table 5) to those made using the PyTorch library (Table 2, Table 3). Achieving mAP of 0.7 for the Faster R-CNN model and 0.71 for the SSD, respectively. The former model achieved better results for the bird, bottle and bus categories. The SSD, on the other hand, performed better for the car, chair and train categories.

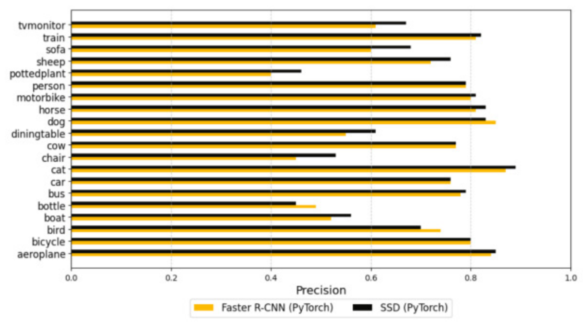

Figure 1: Precision and recall metrics values comparison chart for each class of the Pascal VOC dataset (PyTorch library).

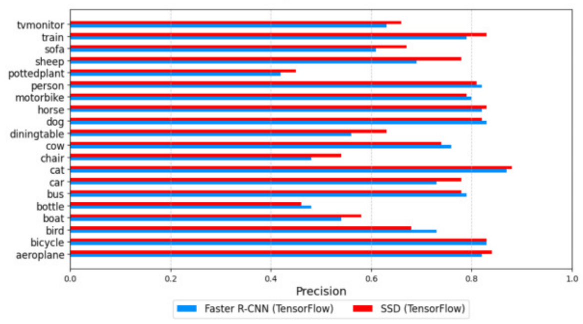

Figure 2: Precision and recall comparison chart for each class of the Pascal VOC dataset (TensorFlow library).

The chart allows for effective interpretation and analysis of the results (Figure 2). The difference in performance between all classes is evident. Overall, the SSD model shows better precision than the Faster R-CNN model. The values of the metrics are calculated in the range from 0 to 1. The higher the value, the higher the percentage of detected detections.

6.3. mAP comparision

Figure number 3 shows a comparison of mAP metric of implemented SSD and Faster R –CNN detection models for PyTorch and TensorFlow libraries. The mAP metric is calculated by calculating the average precision for all classes. The mAP values oscillate between 0.69 and 0.72, showing that the models perform very similarly. However, the SSD models showed slightly higher average precision than the Faster R-CNN models. The classes that received the best results were Cat, Dog and Aeroplane. The classes with the lowest scores were Pottedplant, Chair and Bottle. The largest discrepancies in scores between libraries are for the Bicycle, Car and Bird classes.

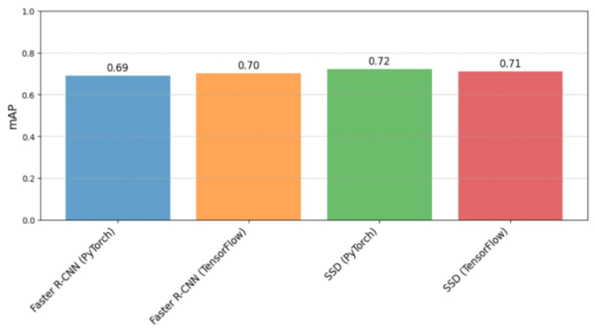

Figure 3: mAP comparison chart for each solution.

6.4. Training time and loss function

Also tested were the training time of each model and their performance and inference time. In terms of training time (Table 6), the model SSD needs significantly less training time than the Faster R-CNN needing almost half as much time. This may be due to the simple and single-stage architecture of the SSD model. Faster R-CNN is a two-stage model and more computationally complex. The SSD also performs better during inference which makes it a more reasonable choice for real-time applications. For both models, the training time for the TensorFlow library was slightly shorter. Also in the case of inference, the library performed up to twice as well than the PyTorch library.

Table 6: Training time and performance of each model

Model	Framework	Training time in minutes	Inference time (ms/image)	FPS (Frames Per Second)
SSD	TensorFlow	170,26	25-30	35-40
SSD	PyTorch	209,26	30-35	30-33
Faster R-CNN	TensorFlow	312,32	65-75	13-15
Faster R-CNN	PyTorch	353,75	75-85	12-13

All models show a sudden drop in the value of the loss function during training during the first five epochs (Figure). There is a rapid improvement in learning quality during the initial phase. For both libraries, the Faster R-CNN model achieves a larger loss during the initial phase than the SSD model. This may be due to the more complex architecture. The lowest values are achieved by the SSD model for the PyTorch library. Around the fifth epoch, the loss stabilizes for all cases and by the end of the training, the values stay at a similar level.

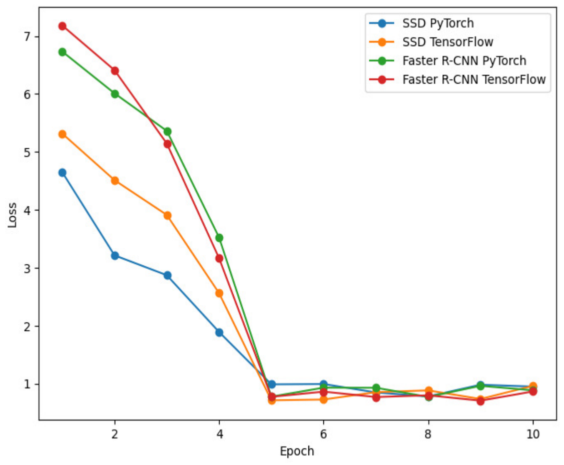

Figure 4: Loss function during training models.

6.5. Detection visualization

To better illustrate the results and how effectively the implemented models work, a visualization showing the detection of an object in a given image was conducted (Figure 5-8). For reliable results, the same image was used in each case.

Each model used a box to mark the area where the predicted object is located. Each implementation does a good job of detecting the object in the images. You can see slight differences in the marked areas, but each model correctly marked the detection area and recognized the category of the detected object with high confidence.

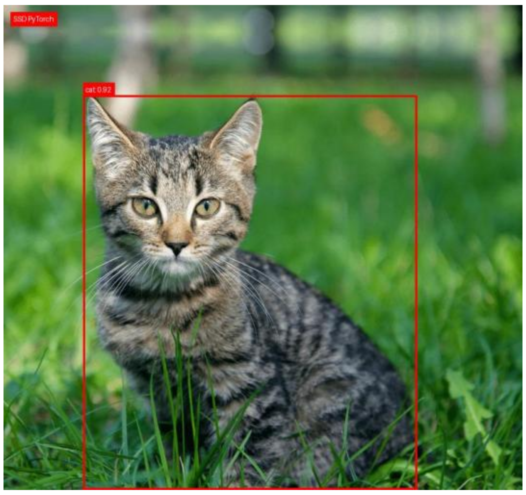

Figure 5: Object detection for the SSD model of the PyTorch library.

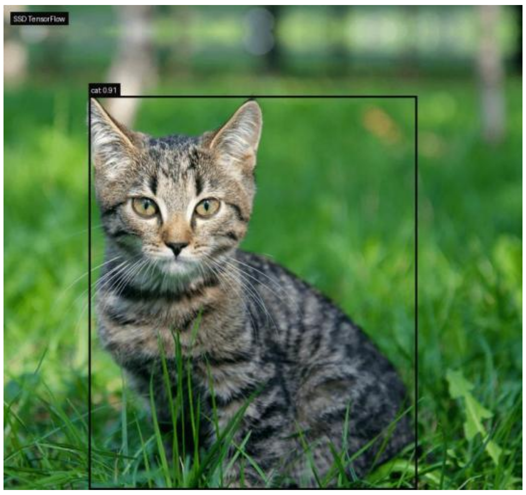

Figure 6: Object detection for the SSD model of the TensorFlow library.

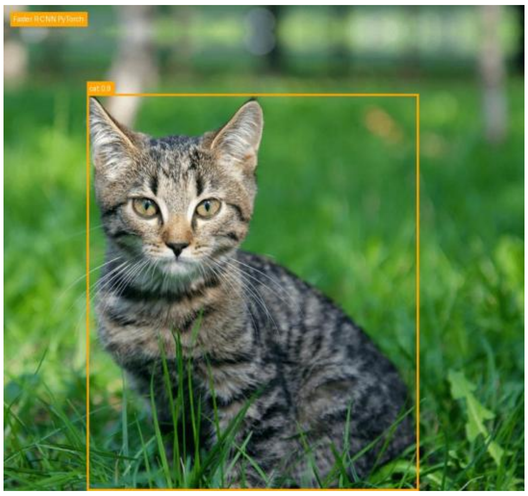

Figure 7: Object detection for the Faster R-CNN model of the PyTorch library.

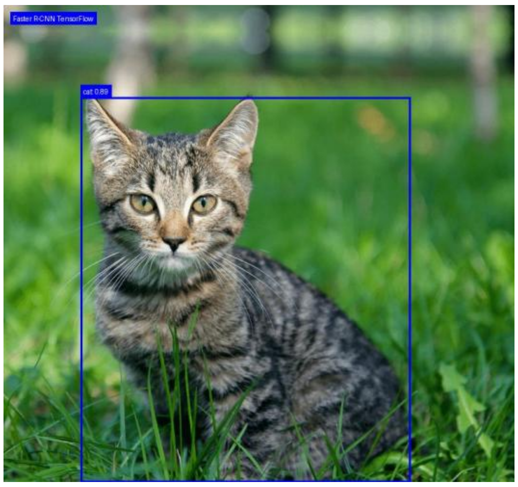

Figure 8: Object detection for the Faster R-CNN model of the TensorFlow library.

6.6. Issues and challenges of detection models

During the implementation as well as the operation of both models, several difficulties were encountered and mistakes made by them. The first was the incompatibility of dimensions between layers which led to errors during execution. An encountered difficulty was the large difference in implementation between the libraries. Required a different approach to model construction, which in the case of the TensorFlow library was more complicated due to its specifics than in the case of the PyTorch library. The SSD model often had difficulty detecting small objects, a result of detectors based on several scales and predefined anchor boxes. Both models had trouble determining the exact boundaries of the frames, which may have been caused by little variation in the training data. In the case of the Faster R-CNN model implementation, incorrect setting of RPN hyperparameters could result in the accumulation of too many false positives or the bypassing of important areas. A significant problem is that the Faster R-CNN model consumes a lot of resources, which leads to GPU memory problems when processing large amounts of data or highresolution images. Another difficulty is the heterogeneity of versions of the PyTorch and TensorFlow libraries and related other tools. There are often compatibility problems which is particularly troublesome for the TensorFlow library. Newer versions of it interfere with other libraries that are not at this point fully adapted to newer versions.

7. Conclusions

Based on the results from the survey and conducting a comparative analysis of the SSD and Faster R-CNN object detection models, several things can be deduced. In both cases, the SSD model achieves higher average precision (mAP) than the Faster R-CNN model. The models' scores are 70.0 and 72.0, respectively. However, this difference is not large enough to matter in terms of overall object detection in images. A bigger difference emerges when we examine the precision for each category separately. In most categories, the SSD performs better. However, there are categories such as bottle or person in which the Faster R-CNN demonstrates to a fairly significant degree greater precision. In situations where we need a solution to work only with a particular category, choosing the right library and model can make a difference.

A big factor in choosing the right solution will be the model's training speed or performance. The SSD model has a lower training loss than the Faster R-CNN model. Another advantage of the SSD model is its performance. Through its less complex architecture, it needs much less time to train and shows better inference. However, if you care about minimizing false detections, the Faster R-CNN will be a better choice.

The choice of the right tool depends on the user's needs and preferences. In terms of object detection accuracy, the choice of technology does not matter much. Both solutions show high accuracy and precision. When you have a weak hardware platform and are working on a small dataset and want to train the model as quickly as possible, SSD will be the right choice. When the priority is to minimize errors during detection, Faster R-CNN is a better choice. Another factor in choosing the right tool may be the complexity of the implementation. If one cares about creating a simple implementation, using the PyTorch library will be a more appropriate choice especially for those who are starting their adventure with the subject of object detection. Using the TensorFlow library may require more knowledge and skills in artificial intelligence.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, X. Zheng, TensorFlow: a system for Large-Scale machine learning, In 12th USENIX symposium on operating systems design and implementation (OSDI) (2016) 265-283, https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi.

[2] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, S. Chintala, Pytorch: An imperative style high-performance deep learning library, Advances in neural information processing systems arXiv preprint arXiv:1912.01703 (NeurlPS) (2019) 32.

[3] S. Bahrampour, N. Ramakrishnan, L. Schott, M. Shah, Comparative Study of Caffe Neon Theano and Torch for Deep Learning (ICLR) (2016) 114.

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, A. C. Berg, SSD: Single Shot MultiBox Detector, In Computer Vision–ECCV 2016: 14th European Conference (2016) 21-37, https://doi.org/10.1007/978-3-319-46448-0_2.

[5] S. Howal, A. Jadhav, C. Arthshi, S. Nalavade, S. Shinde, Object detection for autonomous vehicle using tensorflow, In International Conference on Intelligent Computing Information and Control Systems (ICICCS) (2019) 86-93, https://doi.org/10.1007/978-3-030-30465-2_11.

[6] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, IEEE transactions on pattern analysis and machine intelligence 39(6) (2016) 1137-1149, https://doi.org/10.1109/TPAMI.2016.2577031.

[7] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, K. Murphy, Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors, Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (2017) 7310-7311, https://doi.org/10.48550/arXiv.1611.10012.

[8] S. U. Rehman, M. R. Razzaq, M. H. Hussian, Training of SSD (Single Shot Detector) for Facial Detection using Nvidia Jetson Nano, arXiv preprint arXiv:2105.13906 (2021), https://doi.org/10.48550/arXiv.2105.13906.

[9] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (2016) 770-778.

[10] S. H. Kang, J. S. Park, Aligned Matching: Improving Small Object Detection in SSD, Sensors 23(5) (2023) 2589, https://doi.org/10.3390/s23052589.