MIDDLE EAST TECHNICAL UNIVERSTIY

DEPARTMENT OF COMPUTER ENGINEERING

,

SUMMER PRACTICE REPORT
CENG 400

STUDENT NAME: Murat BAYRAKTAR

ORGANIZATION NAME: STM A.Ş ANKARA İLERİ TEKNOLOJİLER FACILITY

ADDRESS: MUSTAFA KEMAL MAH. 2143 CAD. NO:17 İÇ KAPI NO:1 Ankara / Turkey

START DATE: 07.08.2023

END DATE: 15.09.2023

TOTAL WORKING DAYS: 30

STUDENT'S SIGNATURE ORGANIZATION APPROVAL

Contents

1 INTRODUCTION

In this report I will detail my experience and acknowledgements on the internship that I have completed at **STM** between **07.08.2022 and 15.09.2022** on computer vision. To begin, I will outline the project I was involved in during my internship. I will discuss the challenges I encountered, the solutions I implemented, and instances when I sought assistance from both my colleagues and supervisors. Following that, I will delve into further details about the organization, shedding light on the operational aspects during my internship. Lastly, I will summarize the report by encapsulating the key points discussed.

STM is a company that has been operating in the field of **Defence Industry** since 1981, the day it was eswtablished. STM provides efficient and innovative solutions to a wide range of problems those of which including civil aviation, health, agriculture and energy.

The section that I have worked called the **Autonomous Systems**. There, I was working on the computer vision problems in general, to be more specific, **Few/Zero Shot Object Detection** (the name stands for object detection by very few samples of training or none at all) was the exact problem that I was working on. Apart from the problem being a very challenging task, the first person that worked on this task in the company was me; therefore, we lacked a *know-how* that slowed down the things at first. On the other hand, R& D team of our section was mainly focused on the applied artificial engineering problems; those of which are specialized on the Computer Vision Tasks such as Object Detection, Object Tracking, Superresolution as the company mainly aim producing practical products that army can use in the field.

2 INFORMATION ABOUT PROJECT

Few-Shot Object Detection stands as one of the most demanding and pivotal tasks in the realm of computer vision. This intricate problem revolves around the identification and localization of objects within images, where the system has to perform with limited training data—making it an area of paramount significance for both defense and civilian applications.

During my internship at the defense company, I had the unique opportunity to work on this challenging project. The primary goal was to develop cutting-edge algorithms and techniques that could enable the detection of objects with remarkable accuracy, even when trained on a scant amount of data. This involved tackling numerous technical and conceptual hurdles, which I will elaborate on in the following sections.

My role in this project extended beyond a mere observer, as I actively engaged in problem-solving and solution implementation. The specific problem definition goes **"Identify, detect and localize tanks in an image with few-shots given in the training data (at most 30), as good as the other trained classes such as car, human provided that images are taken at least 100-150m high from a drone or equivalent aerial vehicle."** Throughout this report, I will detail the specific challenges I encountered, the innovative solutions I devised, and instances when I sought guidance and collaboration from my colleagues and supervisors.

2.1 ANALYSIS PHASE

> **2.1 Few Shot Problem**: Having trained on **base classes** (already known and dataset is enough for confident inference), we want to recognize **novel classes** (new objects) in the image that are only seen 1-3-5-10 or 30 (few) times during training.

As the Few-Shot problem 2.1 is much more complex compared to a simple object detection because the model doesn't only learn the new classes, rather it also learns *"how to learn"* too. Before diving in this problem setup; so as to understand the previous solutions and underlying technologies, I had to make a profound research. Starting from the basics of object detection to refresh my memory, I first studied related problem setups including One-Shot (stands for 1 bounding box of the novel class in the training set), Zero-Shot (unseen classes in the training set). Later on I moved on to the previous solutions that are brought up to these specific

problems. On a broader view the main solutions that are presented to solve these problems can be chronologically summed up as below:

2.1.1 Zero Shot Detection

Zero Shot Detection is a specialized branch of object detection that deals with the identification of objects not encountered during the model's training phase. In this scenario, the model is tasked with recognizing entirely novel classes, which were not part of the training dataset. To address this challenge, several approaches have been explored in the literature.

One notable method is Attribute-Based Zero Shot Detection. This approach leverages attributes or textual descriptions associated with objects to facilitate detection. By encoding semantic information about objects, this technique enables the model to generalize its knowledge and detect previously unseen classes.

Another approach involves Embedding Space Alignment, where the model learns to map visual features into a shared embedding space. This space alignment allows for the recognition of novel classes based on their similarity to known classes in the embedding space.

2.1.2 One Shot Detection

One Shot Detection takes the challenge a step further by focusing on scenarios where only a single instance of a novel class is available in the training data. Despite the extreme scarcity of training examples, the model is expected to detect and locate these novel objects accurately.

Siamese Networks [3] have proven to be effective in one-shot detection. These networks learn a similarity metric between image regions, enabling the model to distinguish between novel and known objects based on the similarity score. In Siamese Networks, the model doesn't learn to detect the object A; instead, it learns whether the object A equals the object B.

Another approach is Matching Networks [13], which employ attention mechanisms [12] to weigh the importance of support examples during inference. This attention-based approach aids in precise object localization under one-shot conditions.

2.1.3 Few Shot Detection

Few Shot Detection, as the most challenging of these scenarios, encompasses cases where the model is trained on a very limited number of instances (e.g., 1, 3, 5, 10, or 30) per novel class. In addition to recognizing novel objects, the model must also grasp the meta-knowledge of "how to learn" from these few examples.

One prominent technique in Few Shot Detection is Meta-Learning [11]. Meta-learning frameworks allow models to adapt quickly to new tasks with minimal data. By simulating the few-shot detection problem as a meta-learning task, models can learn effective strategies for recognizing novel classes.

Another approach is Transfer Learning [10], where pretrained models on a large dataset can be fine-tuned on the few-shot object detection task. This approach leverages the knowledge and features learned during the initial training to improve performance on the few-shot problem.

After the many hours of research, I stumbled upon a strong relationship where the attention-based [12] solutions tend to outperform those which don't rely on attention. To make the best out of this research, I took my time on reading "Attention is All You Need" [12] paper to fully comprehend how **Transformers** work and why are they so promising both in Computer Vision and in textual tasks. Then continued with "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" [4] to learn more about **Vision Transformers**. The more I learn the more I was curious, so I checked similar other papers as well as to understand benefits of Transformers, I had to check **RNN** [8] and **LSTM** [6] structures too. After understanding what, how and whys of Transformers, my research expanded on the top of 6 *state-of-the-art* models: **imTED ViT-B** [7], **hANMCL** [9], **DETReg** [2], **META-DETR** [15], **FCT** [5] and **TFA** [14]. The choice of model selection relied upon the 3 facts:

- The credibility of the paper.
- Available source code.
- **Ranking on the MS-COCO 30 Shots Benchmark** [1]

Note that by the time of writing this summer practice report a new paper (DE-ViT) [16] has been published and ranked 1st by outperforming the best model (imTED) by 4 points of AP score (30.2 imTED, 34.0 DE-ViT)

2.2 DESIGN PHASE

The design phase of my internship co-existed with the implementation phase, this is because, with limited time I have to try different things and see what's doable with the resources (time & computing power) I have got in the company. So the iteration between design phase and implementation phase is clearly outlined in the below methodology that I have taken initiative to perform:

Algorithm 1: Best Applicable Model Choosing Algorithm

Data: D: MS-COCO 30 Shots
$M_i \leftarrow$
sorted_by_AP_descending(imTED, hANMCL, DETReg, META-DETR, TFA, FCT);

train(M_i);
$T_M \leftarrow$ *Time it takes to train the model* M;
$T_F \leftarrow$ *Feasible time to train, decided by the team.*;
$AP_N \leftarrow$ *AP result of the novel classes of* M;
$AP_B \leftarrow$ *AP result of the base classes of* M;
$AP_L \leftarrow$ *AP result stated in the paper of* M;
while $T_M \leq T_F$ **do**
  if $AP_N, AP_B = AP_L$ **then**
    $mark_i mplementable(M_i)$;
  else
    discard(M_i);
  end
  $M_i \leftarrow next(M_i)$;
end

After the above iterations of algorithm choosing some of the models are discarded for various reasons:

- Memory requirements are too high. (Requires A100 or equivalent, also given that can't train distributed in the company.)
- Inference time is too high.
- Results are not reproducible.
- Results are too weak.

TFA was the first published one of these models and thus, the results were expected to be weak. I have implemented and tested TFA (see Appendix

A); however, due to poor results I have eliminated this model. This is very normal because novel class predictions are done by only two-stage finetuning. The model is pretrained up to some point and then the backbone is frozen. After that the predictive head is trained to learn the new classes. In my opinion, this is weak because only learning the predict new classes in the box classifier and box regressor causes discrepancies with the backbone and feature extractor.

Later on I tried to implement FCT; yet, the complexity of the code and time it takes to deal with the technical issues wasn't also worth the AP score of the model; therefore, it was discarded too.

Following DETReg and META-DETR both uses DETR, I decided to pick DETReg and implemented and tested. Note that DETReg doesn't completely utilize the Transformers architecture. According to the paper [**?**] "DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call object queries, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a "no object" class.". So again this also has convolutions and even though they say **"End-to-End Object Detection with Transformers"** there are transformers only on the ends. Thus, this was again not fully transformer architecture. (See comparative results of DETR with my unique approach on APPENDIX C.)

The models that took most of my time are the top 2 in the MS-COCO 30 Shots benchmark; namely, imTED and hANMCL (see APPENDIX B for their results). These models are carefully implemented and tested on the MS-COCO. The papers of these models were reproducible and the results were also promising. There are some reasons for that; to elaborate, hANMCL stands for Hierarchical Attention Network via Meta-Contrastive Learning. Basically the authors created two main modules:

- **HAM** (Hierarchical Attention Module): "Combine global attention based blocks with convolution based attention block to compensate shortcomings of each one (HAM). Convolution prepares local context information (this way global's degree of freedom is reduced) Global attention ensembles feature map prediction which convolution does

not." [9] **This stabilizes the feature map, reduce variance that convolution had increased.**

- **MCL** (Meta-Contrastive Learning): Aims to increase the similarity with the correlation of the same class and decrease the similarity with the correlation features of different classes if the anchor is a correlation feature of the same class. In short, it learns dissimilarity, and to similarity by not just class based but by learning from other class similarity anchors. The model learns to recognize if they match in simple words.

On the other hand, imTED stands for Integrally Migrating Pretrained Transformer Encoder-Decoder. This name completely sums up all imTED does. They pretrained transformer encoder and decoder with Masked Autoencoders for a very long time with too much data. Later on these pretrained encoder and decoders are trained on the actual data-sets to the obtain the base models (that predicts the base classes). This enabled both the encoder (acting as feature extractor) and decoder (acting as the predictive head) to completely recognize new objects with few iterations or to easily learn to **learn**. Having pretrained the ViTMAE (1x), and train for the actual data-set (2x) we fine-tune in a very short time for the 30 new shots The comparative study of these models yielded the answer for me: imTED should be implemented with the company data-set.

2.3 IMPLEMENTATION PHASE

In the implementation phase of my internship I had to actually solve the problem. After the model selection I moved on the the actual problem setup by creating the dataset. The implementation happened in 4 steps:

1. Implement model in the server.
2. Employ company dataset.
3. Modify company dataset for few-shot setup.
4. Create a test dataset for few-shot setup.
5. Testing phase.

The company dataset had N many base classes. I have written a tool called "extract_frames.py" that extract frames of a youtube video by given interval of frames, from start second to end second and creates a whole dataset. I picked up videos of military mission from YouTube and created my dataset from 4 videos where tanks exists. I had about 300-400 frames to employ; yet, these were to be labeled. Since I didn't have much time, I used a pretrained model of YOLOv8 to predict the N base classes. I uploaded the frames and the labels to CVAT and only labeled the novel class by hand to earn time for myself.

After the dataset was ready I created a 30 shot version from this dataset and added those who aren't consecutive frames with those 30 to my test set (the company have previously created this dataset). Finishing this task I double check the dataset from 50 random images by 10 times with random seed to see if there is any mistake in the labels. Even though my labels are correct I noticed significant labeling errors in the company dataset that was prepared before I arrived. **I informed my director and he had it fixed, resulting in the test scores of the previous models of the company to be recalculated. A significant improvement of $\approx$ 70 to $\approx$ 90 AP was observed.** After this contribution, I requested the server and I was given a broken server. I had to manually go the lab and try to fix the error. The main error was due to the driver & kernel mismatch of the linux and nvidia. Somehow the previous user have downloaded and modified many drivers of nvidia without checking the kernel version, and didn't bother to fix. Thanks to the one of the engineers, we have reinstalled the Ubuntu and freshly installed packages. I have created my conda environment and installed the **nvidia apex** library. This library seemed to work for the 1x and 2x training yet it threw an error on the 3x training phase causing a troubleshooting and a drawback on my side. After this issue another tiny problem occured which was about the configuration files and the lack of documentation of the repository; yet, I was able to open an issue on their github and solve this problem in a timely manner (see APPENDIX D). Having this fixed, I utilized the dataset with tanks as my novel class an performed the 2x (base training) and 3x (finetuning):

- 2x training is done on the company training dataset (Base Training).
- 3x training is done only on the 30 new shots that I have added (Few Shot training).

In addition to the above contributions, in the small object detection prob-

lem setup of the company their prepared dataset had one class that was very small compared to the others. In other words, the training set representation of this object was bigger compared to the test set representation. *Without using frames from the test set*, I have came up with completely different dataset which was publicly available. Later I fixed annotations of this dataset by labeling the other classes to match the previous one and then merged these frames (around 100-150 frames) into the main dataset. By doing so, performed a retraining on the dataset and saw a 3 points of improvement in the mAP score. Since I didn't have the time and energy to fully utilize the new dataset that I have found, I wanted show even a small part of this dataset is enough for improvement.

2.4 TESTING PHASE

In the testing phase of my internship, I needed to test the accuracy of the imTED model on few-shot problem; namely, I needed to measure how accurate are my tank detections provided that only 30 shots are seen during the training period. Having carefully set up the testing procedure, I got the results from the model and presented them to my team. The tank precision score is measured as **0.5776** AP, which is very significant because the mAP over the novel classes on the MS-COCO was around the **0.3** (for more see APPENDIX E). On the other hand I have collected 50 random detections from the test set to also observe the qualitatives on my detections as well as the quantitatives (see APPENDIX E for some of these).

3 ORGANIZATION

3.1 ORGANIZATION AND STRUCTURE

STM is Turkish defence industry company working on various fields to produce quality hardware and software that can be used in the field. There were 4 offices in total for different purposes. The main building is for HR & business, NEP Office is for the administration stuff, OSTIM office is for producing tangible products and located very far away from the other 3 offices and the final office is the one that I have worked where mainly AI Engineers, System Engineers and other engineers are located. Here there are also hierarchy, the director is responsible for the whole building and under him there are other floors such as DevOps, Computer Vision etc. Every floor has a main manager who is connected to the director and every floor has a secondary manager who is connected to the main manager. Secondary manager is always in touch with the team and is reporting to the main manager and main manager is reporting to the director. The communication is bidirectional, meaning, the orders are also coming in the same fashion.

3.2 METHODOLOGIES AND STRATEGIES USED IN THE ORGANIZATION

In STM, there are many projects most of which is unknown by the many engineers. Here engineers work on specific parts of the problems and create solutions. These solutions are brought together on the upper levels of security. The engineers are abstracted from the top secret objectives and solely work on the engineering part of the problem. Therefore they employ a **"need-to-know principle"** on their works. As an intern I was only able observe the computer vision under the autonomous systems. There, people work mainly on object detection, object tracking and super resolution. There are weekly stand-up meetings where everyone is stood up around the cubicles and give briefing on what they have done, what are they going to be doing. These are generally unimportant things but whenever the team has very critical problem or topic to discuss, they quickly arrange a room to gather around a table and discuss for long hours.

4 CONCLUSION

In conclusion, during my internship period I learned how research is being followed on the industry side. The academy studies very specific and bounded problems; on the other hand, industry is more focused on solving real world problems. Therefore in addition to academic knowledge, the real world adjustment is necessary to solve these problems. From what I see, the industry is always few steps behind the academy because while academy has more resources and time to build something, industry needs to implement and make these solutions work in a timely manner by carefully controlling, monitoring, keeping a balance of the efficiency and performance.

5 APPENDICES

5.1 APPENDIX A

5.1.1 Training Shots

Category	#Instances	Category	#Instances	Category	#Instances
person	30	bicycle	30	car	30
motorcycle	30	airplane	30	bus	30
train	30	boat	30	bird	30
cat	30	dog	30	horse	30
sheep	30	cow	30	bottle	30
chair	30	couch	30	potted plant	30
dining table	30	tv	30		
total	600				

5.1.2 Testing

Category	#Instances	Category	#Instances	Category	#Instances
person	10538	bicycle	313	car	1641
motorcycle	388	airplane	131	bus	259
train	212	boat	458	bird	453
cat	195	dog	223	horse	305
sheep	306	cow	376	bottle	939
chair	1594	couch	236	potted plant	429
dining table	633	tv	257		
total	19886				

5.1.3 Average Precision (AP)

AP	**AP50**	**AP75**	**APs**	**APm**	**APl**	
11.694	20.506	11.659	4.819	10.948	18.225	

5.1.4 Per-Category Bbox AP

Category	AP	Category	AP	Category	AP
person	1.419	bicycle	0.302	car	26.515
motorcycle	9.805	airplane	16.313	bus	34.556
train	18.296	boat	4.690	bird	8.007
cat	7.628	dog	6.001	horse	8.739
sheep	10.392	cow	6.493	bottle	10.352
chair	5.201	couch	14.588	potted plant	1.699
dining table	5.347	tv	37.537		

5.1.5 Evaluation Results for coco_test_novel in CSV Format

Task	nAP	nAP50	nAP75	nAPs	nAPm	nAPl	AP
bbox	11.6939	20.5061	11.6585	4.8193	10.9483	18.2253	11.6939

5.2 APPENDIX B

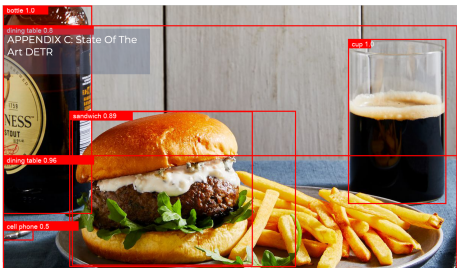
5.2.1 imTED Results

Class Name	**mAP**
person	0.2693
bicycle	0.5798
car	0.4096
motorcycle	0.6739
airplane	0.7379
bus	0.7217
train	0.5968
boat	0.5494
bird	0.5674
cat	0.5249
dog	0.5182
horse	0.386
sheep	0.9258
cow	0.9051
bottle	0.4242
chair	0.6791
couch	0.7813
potted plant	0.7838
dining table	0.499
tv	0.8037
All Categories mAP	0.5833
AP @[IoU=0.50, area=all, maxDets=100]	0.583

5.2.2 hANMCL Results

Class Name	**mAP**
person	0.0223
bicycle	0.2221
car	0.0801
motorcycle	0.4726
airplane	0.5172
bus	0.5115
train	0.5634
boat	0.3865
bird	0.5862
cat	0.6549
dog	0.3865
horse	0.1343
sheep	0.7924
cow	0.9107
bottle	0.0346
chair	0.2633
couch	0.7147
potted plant	0.2914
dining table	0.4997
tv	0.5627
All Categories mAP	0.4431
AP @[IoU=0.50, area=all, maxDets=100]	0.443

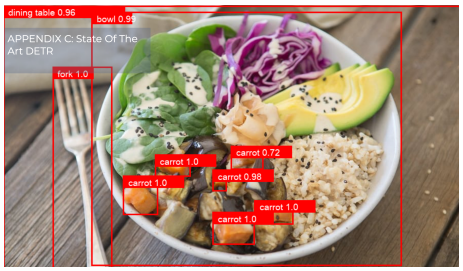
5.3 APPENDIX C



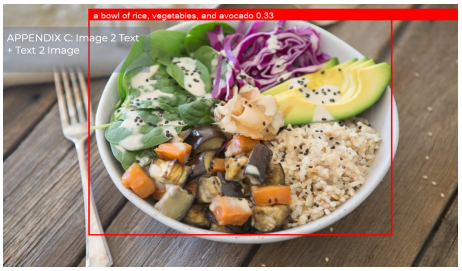
(a) DETR

(b) My Approach

Figure 1: Comparison of *SOTA* with my approach



(a) DETR

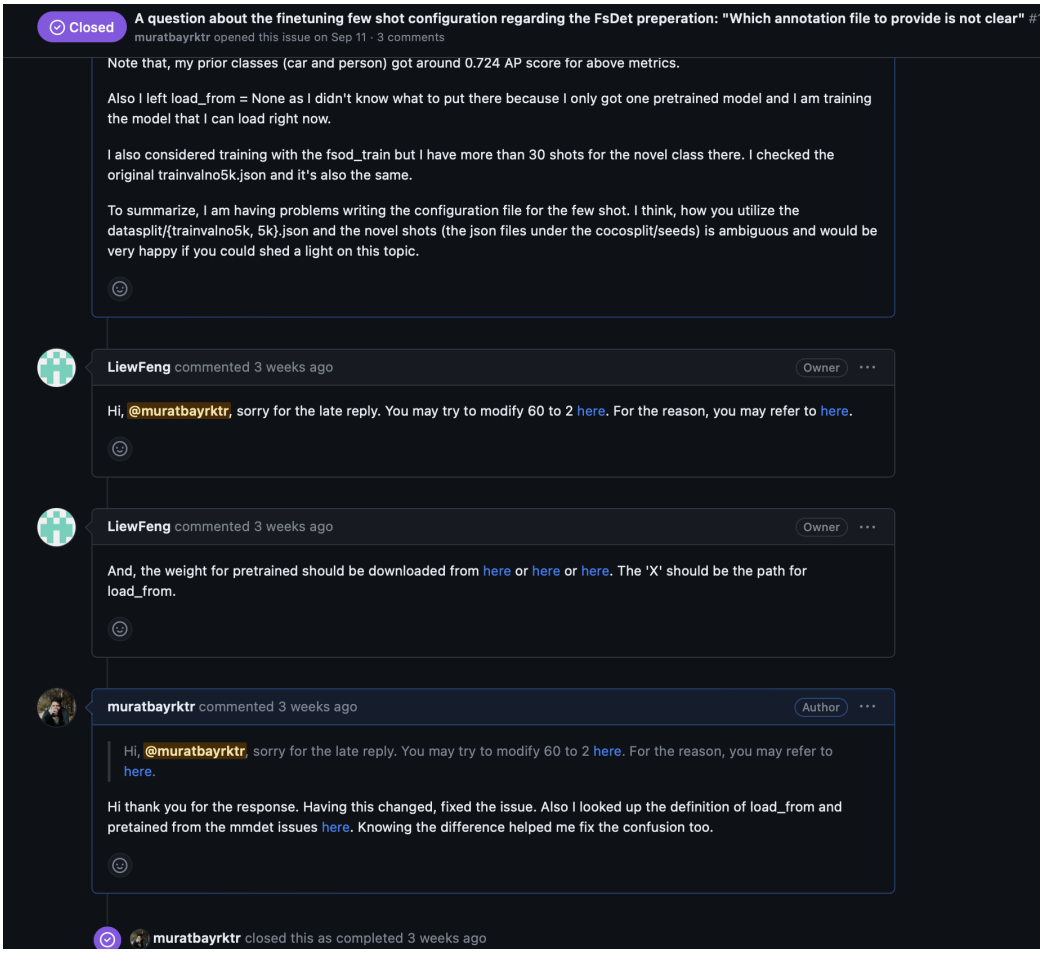


(b) My Approach

Figure 2: Comparison of *SOTA* with my approach

5.4 APPENDIX D

Figure 3: Github issue that I seeked help for and resolved later.

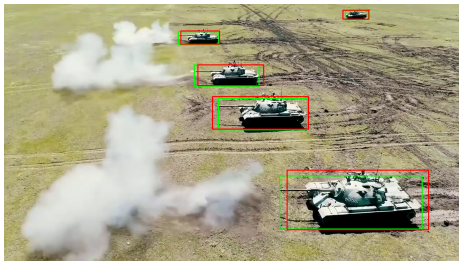


5.5 APPENDIX E

5.5.1 Few-Shot Results

Metric	Value
AP @[IoU=0.50:0.95, area=all, maxDets=100]	0.148
AP @[IoU=0.50, area=all, maxDets=1000]	0.411
AP @[IoU=0.75, area=all, maxDets=1000]	0.057
AP @[IoU=0.50:0.95, area=small, maxDets=1000]	0.044
AP @[IoU=0.50:0.95, area=medium, maxDets=1000]	0.235
AP @[IoU=0.50:0.95, area=large, maxDets=1000]	0.287
AR @[IoU=0.50:0.95, area=all, maxDets=100]	0.252
AR @[IoU=0.50:0.95, area=all, maxDets=300]	0.252
AR @[IoU=0.50:0.95, area=all, maxDets=1000]	0.252
AR @[IoU=0.50:0.95, area=small, maxDets=1000]	0.084
AR @[IoU=0.50:0.95, area=medium, maxDets=1000]	0.389
AR @[IoU=0.50:0.95, area=large, maxDets=1000]	0.439

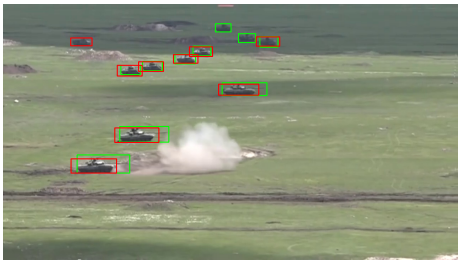
5.5.2 Some examples of detections with Few-Shot



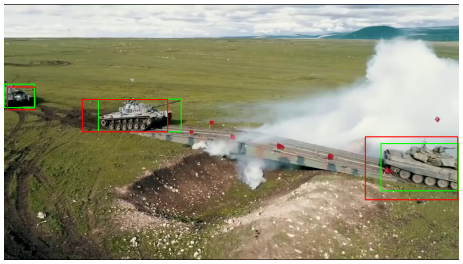
(a)

(b)

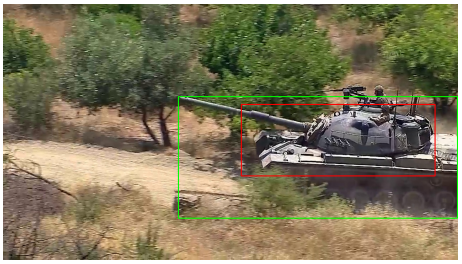
(a)



(b)

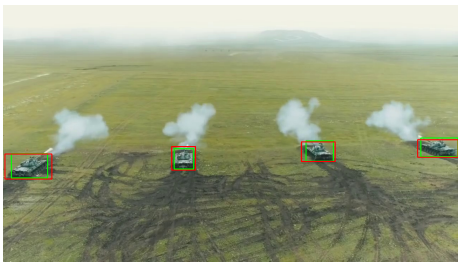


(a)



(b)

(a)



(b)

References

[1]

[2] Amir Bar, Xin Wang, Vadim Kantorov, Colorado J Reed, Roei Herzig, Gal Chechik, Anna Rohrbach, Trevor Darrell, and Amir Globerson. Detreg: Unsupervised pretraining with region priors for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14605–14615, 2022.

[3] Davide Chicco. Siamese neural networks: An overview. In *Methods in Molecular Biology*, pages 73–94. Springer US, August 2020.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[5] Guangxing Han, Jiawei Ma, Shiyuan Huang, Long Chen, and Shih-Fu Chang. Few-shot object detection with fully cross-transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5321–5330, 2022.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] Feng Liu, Xiaosong Zhang, Zhiliang Peng, Zonghao Guo, Fang Wan, Xiangyang Ji, and Qixiang Ye. Integrally migrating pre-trained transformer encoder-decoders for visual object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6825–6834, 2023.

[8] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5(64-67):2, 2001.

[9] Dongwoo Park and Jong-Min Lee. Hierarchical attention network for few-shot object detection via meta-contrastive learning. *arXiv preprint arXiv:2208.07039*, 2022.

[10] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[11] Joaquin Vanschoren. Meta-learning. *Automated machine learning: methods, systems, challenges*, pages 35–61, 2019.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[13] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

[14] Xin Wang, Thomas E Huang, Trevor Darrell, Joseph E Gonzalez, and Fisher Yu. Frustratingly simple few-shot object detection. *arXiv preprint arXiv:2003.06957*, 2020.

[15] Gongjie Zhang, Zhipeng Luo, Kaiwen Cui, and Shijian Lu. Meta-detr: Few-shot object detection via unified image-level meta-learning. *arXiv preprint arXiv:2103.11731*, 2(6), 2021.

[16] Xinyu Zhang, Yuting Wang, and Abdeslam Boularias. Detect every thing with few examples. *arXiv preprint arXiv:2309.12969*, 2023.