Ekaterina Danilova

Few-shot object detection of bacteria colonies

Master's Thesis in Information Technology

May 16, 2024

University of Jyväskylä

Faculty of Information Technology

Author: Ekaterina Danilova

Contact information: daniley@jyu.fi

Supervisors: Pölönen Ilkka, and Ihalainen Janne

Title: Few-shot object detection of bacteria colonies

Työn nimi: Bakteeripesäkkeiden vähäotostunnistaminen

Project: Master's Thesis

Study line: Cognitive Computing and Collective Intelligence

Page count: 57+0

Abstract: This thesis aims to research and apply various approaches to few-shot object detection to address the real-life problem of detecting bacterial colonies. Bacteria detection is an important topic across multiple fields. Traditional methods include chemical analysis and various image processing algorithms. This thesis investigates the development of the deep learning approach to the selected problem in order to find simple and effective solution.

Since manual data collection is a laborious task, this work puts emphasis on few-shot object detection, focusing on developing an approach optimized for minimal training data with less than 10 labeled images.

Various techniques, both model- and data-centric, are researched and utilized to identify the most efficient solution. Methods such as data augmentations, meta-learning, and fine-tuning are discussed and evaluated. The main emphasis of the practical part is on exploring the fine-tuning approach, together with data enhancements such as various augmentations and the use of additional close-domain data, and eventually comparing and analyzing their performance.

The findings suggest that fine-tuning is an effective and simple strategy for few-shot training, particularly when combined with other methods. Incorporating close-domain data significantly enhances model performance, and data augmentations also show good results in expanding the dataset size.

This work advances the application of few-shot learning in data-constrained environments. Given the specific challenge of bacteria detection, the insights gained are potentially valuable across multiple fields and applications.

Keywords: object detection, few show detection, machine learning, bacteria colony

Suomenkielinen tiivistelmä: Tämä opinnäytetyö pyrkii tutkimaan ja soveltamaan erilaisia lähestymistapoja vähäotos-tunnistamiseen bakteeripesäkkeiden havaitsemisessa.

Bakteerien tunnistaminen on tärkeä aihe monilla aloilla. Perinteisiä menetelmiä ovat kemiallinen analyysi ja erilaiset kuvankäsittelyalgoritmit. Tämä opinnäytetyö tutkii syväoppimisen lähestymistapaa valittuun ongelmaan löytääkseen yksinkertaisen ja tehokkaan ratkaisun.

Koska manuaalinen datankeruu on työlästä, tässä opinnäytetyössä keskitytään kehittämään lähestymistapaa vähäotos-tunnistamiseen, joka on optimoitu vähäiselle koulutusdatalle, alle kymmenelle merkitylle kuvalle.

Opinnäytetyössä tutkitaan ja hyödynnetään erilaisia tekniikoita, sekä malli- että datakeskeisiä, tehokkaimman ratkaisun löytämiseksi. Siinä käsitellään ja arvioidaan menetelmiä, kuten datan parannusta, metaoppimista ja neuroverkon hienosäätämistä.

Käytännön osassa pääpaino on hienosäätömenetelmien tutkiminen yhdessä datan parannusten, kuten erilaisten hienosäätöjen ja vastaavanlaisen lisädatan kanssa ja lopulta niiden suorituskyvyn vertaamisessa ja analysoinnissa.

Tulokset osoittavat, että hienosäätö on tehokas ja yksinkertainen strategia vähäotoskoulutukseen, erityisesti kun sitä yhdistetään muihin menetelmiin. Vastaavanlaisen lisädatan sisällyttäminen parantaa merkittävästi mallin suorituskykyä, ja datan hienosäädöt osoittavat myös hyviä tuloksia tietojoukon laajentamisessa.

Tämä työ edistää vähäotosoppimisen soveltamista datarajoitteisissa ympäristöissä. Ottaen huomioon bakteerien havaitsemisen haasteellisuuden, saadut tulokset ovat potentiaalisesti arvokkaita useilla aloilla ja sovelluksissa.

Avainsanat: kohteentunnistus, harvan esityksen tunnistus, koneoppiminen, bakteeripesäke

List of Figures

List of Tables

Contents

1 Introduction

The objective of this thesis is to detect specific fluorescent bacteria in images of Petri dishes. The solution must be compatible with limited training data, given the challenging and time-consuming nature of generating a substantial number of annotated images.

The rationale behind this study is rooted in the real-world utility of the proposed solution. The problem arises from the need to detect bacteria identified in research by Nissinen et al. 2023. These bacteria (aerobic anoxygenic phototrophic bacteria, later referenced as AAPB) are unique because they perform photosynthesis in aerobic conditions without producing oxygen, using diverse organic compounds as electron donors. Although traditionally associated with aquatic ecosystems, AAPB have been discovered in various habitats, including the phyllospheres (leaf surfaces) and endospheres (internal tissues) of boreal and subarctic plants. This study found AAPB to be present in these plant microbiomes across different climate zones in Finland, suggesting their role in plant adaptation to various environments (Nissinen et al. 2023).

However, the task can be extended to multiple other fields and applications since object detection of bacteria colonies includes many uses. Although methods vary according to specific objectives, the concept remains consistent. The applications include tasks such as testing the performance of antibiotics (Kee et al. 2013) or analyzing the safety of drinking water (Vail et al. 2003). Therefore, this thesis has the potential to be useful across a broad array of similar problems. Typically, the task of classifying and counting bacterial colonies is performed manually by lab technicians, leading to potential subjectivity. Adopting an automated approach could enhance consistency in results and simplify their work.

There are multiple approaches to automating the task of bacteria detection. This thesis will concentrate on the image-based bacteria detection techniques that utilize Deep Learning ignoring the other approaches such as those that use chemical reactions.

Many image-based bacteria detection methods do not utilize Deep Learning. For instance, there are implementations that use tools like Support Vector machines (Zhang et al., n.d.). Such a solution includes two stages: classification and detection. The detection is achieved

by using structurally similar images, making separating the regions like "background" and "container" easier and utilizing multiple image processing algorithms like Otsu's method to detect the important parts of the images. However, methods like the one described above all share common issues: they are exceptionally complex and require extensive image preprocessing, such as noise removal and the application of thresholding algorithms. Another problem shared by such image analysis methods lies in how dependent it is on the input format and how methods must be adapted to it. For instance, the cases such as those mentioned by Kee et al. 2013, where spectral imaging is used, are unlikely to be successfully applied to other solutions without algorithmic changes.

With the development of neural networks, it is possible to transition this task to deep learning algorithms, which can potentially simplify the process and offer more universal applicability.

This thesis aims to find a realistic and useful approach to detecting bacteria colonies and, in the process, to answer the following research questions:

- What are the most effective methods for few shot detection in the selected domain?
- What role do data augmentation and synthetic data generation play in improving the performance of few-shot object detection models for bacteria colony identification?
- Can transfer learning from related domains enhance the accuracy and ability to generalize of few-shot object detection models in identifying bacteria colonies?

2 Background

The primary objective of this thesis is to address the challenge of detecting bacteria in few-shot datasets. Few-shot training is a subset of supervised machine learning that specializes in handling limited data. The problem of learning on limited data includes multiple variations, such as zero-shot, one-shot, and few-shot learning, with the latter being the main method used in this work. The need for such methods arises from the difficulty of creating labeled data; therefore, many real-life learning tasks suffer from a lack of sufficiently large training datasets. Few-shot learning in its principle emulates the human way of learning where having some previous knowledge allows it to quickly learn new things from limited examples (*What Is Few-Shot Learning? | IBM* 2024).

There is no strict definition of what exactly constitutes a few-shot dataset since there is no clear maximum limit on the number of samples. However, the data used in this work is much smaller than what one would normally expect for a task like this, consisting of fewer than ten labeled images, thus qualifying as an example of few-shot learning.

The use of a few-shot dataset introduces many problems, such as a low variety of data, which would normally lead to overfitting; therefore, it is challenging to simply apply few-shot data to any model and train it from scratch. Overfitting can be spotted by the training loss continuing to drop during the training by the validation loss growing like shown on Figure 1. This situation means that the model got too closely adapted to the training data and did not learn the generalized properties. There are multiple solutions to and causes of overfitting. It can be caused by an overly complicated model or by noisy training data, with the small size of the training dataset being a very likely cause of these two issues, for instance by using too complicated model for too little amount of data. There are multiple training techniques to prevent it, such as modifying the network, early stopping, dropout, increasing the training dataset size etc (Pothuganti 2018). Since this thesis focuses on small training datasets, it is important to prevent overfitting.

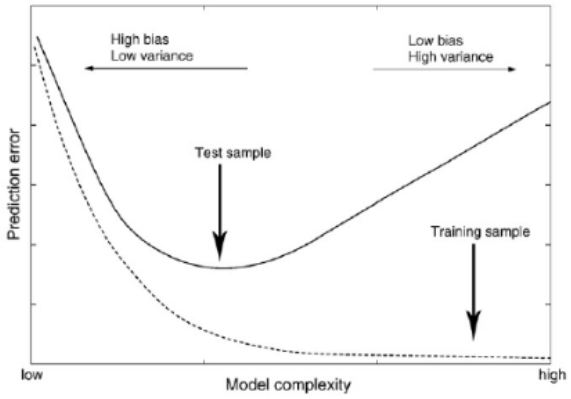


Figure 1: Overiftting by Pothuganti 2018.

Few-shot learning can be applied to various tasks, which vary in difficulty. While few-shot classification is a less complicated problem, few-shot object detection poses additional challenges, such as training the object detector with a small amount of data (Antonelli et al. 2022). The below mentioned methods apply to both object detection and classification tasks but the emphasis of the work is on object detection solutions.

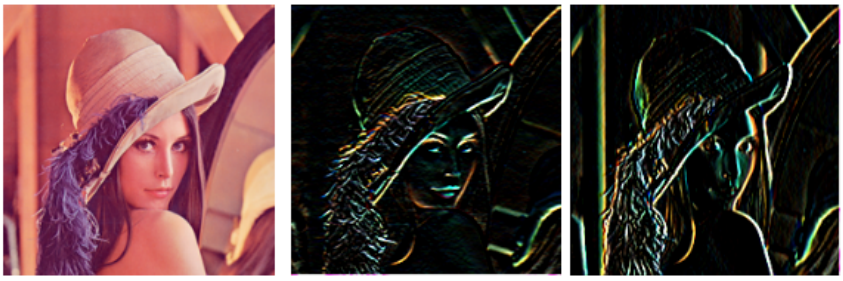
2.1 Convolutional neural networks

Convolutional neural networks were a breakthrough in the field of object detection since they allowed efficient processing spatial hierarchies in high-dimensional data, which makes them well suited for tasks like image or video processing.

CNNs introduce three new types of layers: convolutional, pooling, and fully connected ones.

The convolutional layer serves as the fundamental building block of a CNN and is used for constructing feature maps. It contains kernels that, while small in spatial dimensions, span the entire depth of the input image to capture all aspects of the input features. For example, in an RGB image, each kernel processes all three color channels, reflecting a depth of 3. These kernels, usually square (e.g., 3x3 or 5x5 pixels), slide over the input image to produce a 2D feature map that captures essential patterns and features (see Figure 2). The pooling layer is aimed at reducing dimensionality while retaining important features

by applying different pooling operations like max or average pooling in order to reduce computational load and overfitting. Fully connected layers, positioned towards the end of a CNN architecture, contain neurons that link to all activations from the previous layer, integrating global information from the feature maps. The layers can be arranged in various ways. The architecture that utilizes CNNs allows for working with high-dimensional data using fewer resources and helps avoid overfitting (O'Shea and Nash 2015).



(a) Lenna (b) Horizontal edge (c) Vertical edge

Figure 2: Convolutional kernels representation of the input image (example by Wu 2017).

All the object detection models mentioned in this thesis are based on CNNs.

2.2 Object detection

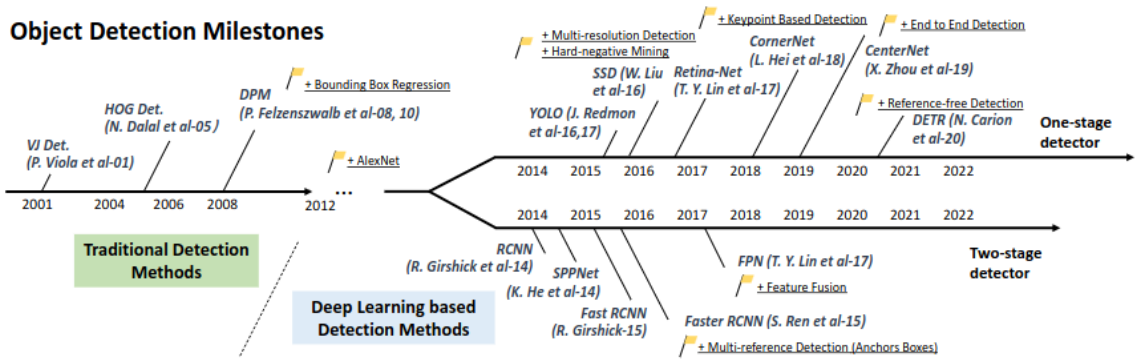


Figure 3: Object detection milestones by Zou et al. 2019.

The history of the object detection technologies on Figure 3 shows the breakthrough that happened with the adoption of CNNs described in section 2.1. After object detection methods

started to utilize Deep Learning they got separated in two groups: one-stage and two-stage. Two-stage detectors operate by executing two tasks: generating proposals for the objects and performing the classification task within the suggested bounding boxes. In contrast, one-stage detectors simplify this process by combining the two tasks, dividing the image into a grid, and simultaneously predicting bounding boxes and classes for each grid cell. Typically, the first group tends to offer faster performance, while the second group achieves higher accuracy. However, some one-stage detectors can combine the advantages of both types (Zou et al. 2019).

2.2.1 RetinaNet

For instance, Lin et al. 2017 introduced RetinaNet as an example of a one-stage detector that achieves both high performance speed and accuracy. RetinaNet employs a Focal Loss function during training to prevent easy negatives from overwhelming the training process and to improve performance in scenarios where the image predominantly contains background classes. It combines a Feature Pyramid Network backbone with two sub-networks used for classification and box regression tasks. The Feature Pyramid Network efficiently discovers objects at different scales by having each layer of the pyramid handle different dimensions of the input image. ResNet is used as the backbone that extracts feature maps of various scales (Lin et al. 2016).

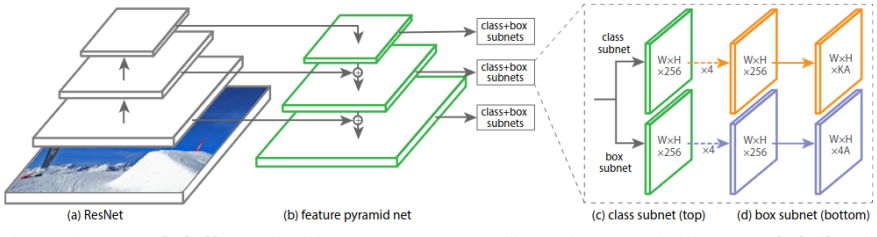


Figure 4: Image and caption by Lin et al. 2016: RetinaNet architecure **a**) producing the multi-scale convolutional feature pyramid using a residual network (ResNet) as an encoder (left) and a feature pyramid net (FPN) as a decoder (right). (**b**) Class subnet for classifying anchor boxes (top), and box subnet for regressing from anchors boxes to ground-truth object boxes (bottom).

The ways of approaching the problem of few-shot object detection was separated in two groups for clarity: data-centric and model-centric. The first approach was meant to improve the dataset and make it effectively not "few-shot" data anymore and the next approach focused on improving the model performance on the few-shot data.

The data-centric methods listed below can be used as a possible solution to improving the few-shot dataset.

2.3 Data-centric methods

2.3.1 Data augmentations

Applying various augmentations to data is the most straightforward way to increase the training set. Augmentations not only extend the training set but also address the generality issue of the small dataset by adding variety. For instance, Channel Dropout can help add variety to colors, and various resizing and cropping techniques can help to learn different sizes of objects.

However, simply applying multiple augmentations to the data is not always the right approach. Sometimes it can cause the issue of underfitting, making the training slow and inef-

ficient (Sikka 2021). The augmentations should be selected in such a way that they achieve two purposes: adding variety to the training data while keeping it close enough to the test data

2.3.2 Same-domain data

Another data-based approach to improving few-shot learning is re-using another similar dataset which has plenty of training data. Using close-domain data can solve the over-fitting issue and increase general accuracy. For instance, as described by Casado-García et al. 2019, fine-tuning the model with the closely-related data significantly improves performance and the model's ability to generalize.

2.3.3 Synthetic data

Generating synthetic data is another approach to expanding the training dataset. Various methods exist for achieving this. For example, the VAE (Variational Autoencoder) is a type of generative network that can create additional samples of training data, significantly enlarging the training dataset size (Chadebec et al. 2022). A basic variational autoencoder is composed of two parts: an encoder and a decoder. The encoder captures the input representation and encodes it into a latent representation, while the decoder reconstructs the data from this encoded representation. A key feature of variational autoencoders, as opposed to standard encoders, is their ability to produce different encodings from the same input. VAEs tend to generate diverse samples which is good for creating varied training data.

Variational autoencoders are widely used in few-shot learning in various ways and not only as the method of generating additional data. They can be used for encoding various types of data, so for instance they can be used as the part of meta-learning approach that will encode some learned class features via VAEs (Han et al. 2023). While VAEs often need a lot of training data, there are developments to apply it to few-shot setting (Chadebec and Allassonnière 2021).

Another approach involves GANs, which work by having two networks (a generator and a discriminator) compete with each other. GANs tend to generate less diverse data than VAEs.

They also require more training data, which might make them less suitable for few-shot learning, even though there are some adaptations of GANs for few-shot settings (Robb et al. 2020).

Other method involves diffusion models, which work by applying multiple transformations to the data in order to generate new samples. They often provide higher-quality results than VAEs and allow for the generation of high-resolution data. While they typically require a lot of training data, there are approaches to adapt them for few-shot learning (Giannone, Nielsen, and Winther 2022).

The main issue with the above solutions is that they provide unlabeled data, which does not address the problem of the expensive and labor-intensive task of manually labeling the samples. This does not apply to all the methods, for instance, the data generated using the style transfer retains the bounding boxes not requiring additional work as shown in bacteria object detector by Pawłowski, Majchrowska, and Golan 2022.

The subsequent methods primarily focus on the models and training themselves, unlike the previously described data-centric solutions.

2.4 Model-centric methods

2.4.1 Meta-learning

There are multiple approaches to meta-learning, but the main idea revolves around a network that is "learning to learn." Meta-learning includes two parts: a base learning algorithm that solves the objective and a meta-learning algorithm that updates the base learning algorithm. Various algorithms fall under the definition of meta-learning as long as they work on end-to-end optimization of the base learning algorithm (Hospedales et al. 2021).

Meta-learning can be divided into three types: metric-based, model-based, and optimization-based. Metric-based learning focuses on discovering a good feature space for use in new tasks, model-based learning adapts the model state to new tasks, and optimization-based meta-learning aims to provide the fastest learning possible, with the popular MAML network (Huisman, Rijn, and Plaat 2021) being an example of it. Prototypical networks (Snell,

Swersky, and Zemel 2017) and Siamese networks (Koch, Zemel, Salakhutdinov, et al. 2015) are examples of metric-based meta-learning and the model-based meta-learning approaches include networks such as described by Santoro et al. 2016.

Meta-learning is often used specifically for tasks concerning few-shot data since meta-learning algorithms allow improving the network with abundant data and later applying the improved algorithm to few-shot or even zero-shot data. For instance, the Meta-RCNN implementation by Yan et al. 2019 suggests using meta-learning on RoI (region of interest) features, providing state-of-the-art performance.

Many of the mentioned methods can be combined; for instance, it is possible to utilize VAEs as part of meta-learning models as described by Han et al. 2023.

2.4.2 Metric learning

Metric learning is a subset of Machine Learning that can be helpful for few-shot training. It is not a subtype of Meta-learning and is different from metric-based meta-learning.

Metric learning focuses on learning a distance function over objects tuned to a given task (Kulis et al. 2013). The goal is to learn a metric or distance function that measures how similar or dissimilar two objects are in a way that is useful for the specific task at hand, such as classification, clustering, or retrieval tasks. This means that metric learning is different from metric-based meta-learning in the sense that it adapts to the given task and dataset while meta-learning aims to generalize. Metric learning is used in applications like face recognition, where the aim is to learn a space in which distances directly correspond to a measure of facial similarity. However, metric learning has received some criticism, with many papers overestimating the efficiency of it and therefore causing doubts about whether these methods are as useful as they often claim to be (Musgrave, Belongie, and Lim 2020).

As Karlinsky et al. 2019 suggests, metric learning can be incorporated into few-shot object detection. The proposed solution includes using Distance Metric Learning (that learns backbone network parameters, embedding space, and distributions of different categories) classifier as a part of a standard object detection model.

2.4.3 Fine-tuning

Fine-tuning is one of the stages of transfer learning, where the first stage includes training the model on one dataset and then freezing and re-training only some layers on the other dataset. Fine-tuning can improve the performance of multiple networks even if the target dataset it not small and has enough data to be trained on the empty model. As Yosinski et al. 2014 discovered: "transferring features and then fine-tuning them results in networks that generalize better than those trained directly on the target dataset".

Fine-tuning is not immune to overfitting, if the fine-tuned dataset is too small or number of parameters is large and too many of the layers are re-trained the model can overfit on the new small dataset. Therefore it is important to understand which features can be re-trained and which should be left frozen. Generally keeping more of the layers frozen allows the network to keep its base features and be fine-tuned without overfitting (Yosinski et al. 2014). Such logic is used in the approach below where the fine-tuning does not affect the base features. Figure 5 illustrates the second stage of the fine-tuning approach used for few-shot object detection by Wang et al. 2020. This few-shot object detection method involves initially training the object detector on classes with abundant data, followed by fine-tuning only the last layer of existing detectors for rare classes. The implementation is based on the pre-trained Faster R-CNN model, where the box predictor is fine-tuned on both novel and base classes, while the feature extractor remains unchanged. This method shows significant improvements over existing meta-learning approaches. It not only simplifies the training process but also enhances the model's ability to generalize from limited data, addressing a key challenge in few-shot learning scenarios.

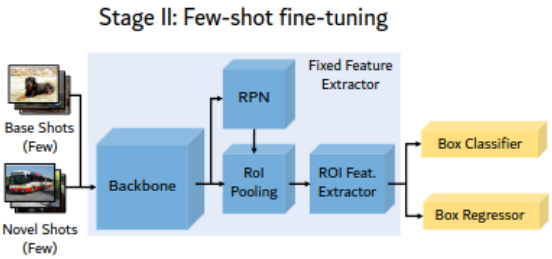


Figure 5: Fine-tuning by Wang et al. 2020

3 Data and methods

This section describes the types and amounts of data available and details on the experiments. It also includes short conclusions and notes about the acquired results, but the full numerical data for the results can be found in Section 4. The section includes all the attempted methods including the ones that were decided to not be viable. Those not viable methods are shortly listed first.

The hardware includes 2 NVIDIA RTX A4000 GPUs with 16 GB RAM each resulting in 32 GB RAM total.

3.1 Data selection

The pictures were taken with the same camera and background in order to be further analyzed for their fluorescent properties. The process of collecting the data for object detection is described on stage A of the Figure 6 where Petri dishes were illuminated with white light and the picture of the dish was captured with a Rasberry Pi PiNoir Camer V2 with ordinary RGB-settings. The process of construing and using the NIR (Near-infrared imaging system) is described by Franz et al. 2023.

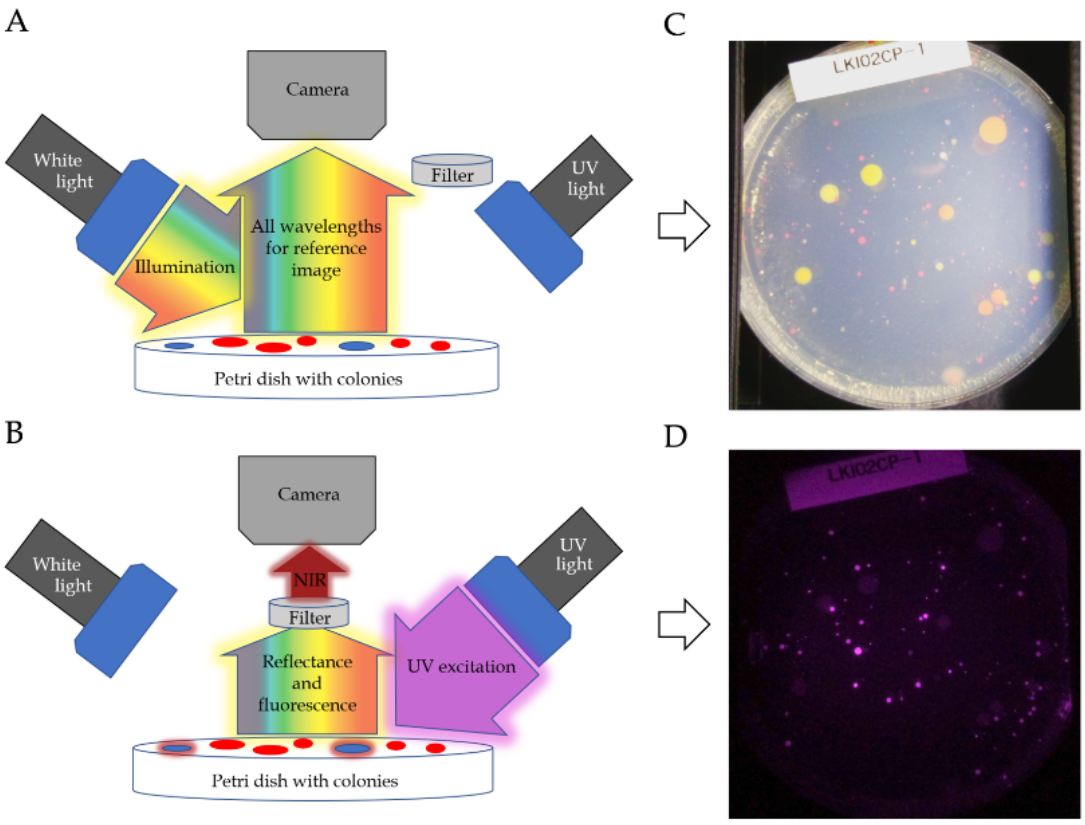


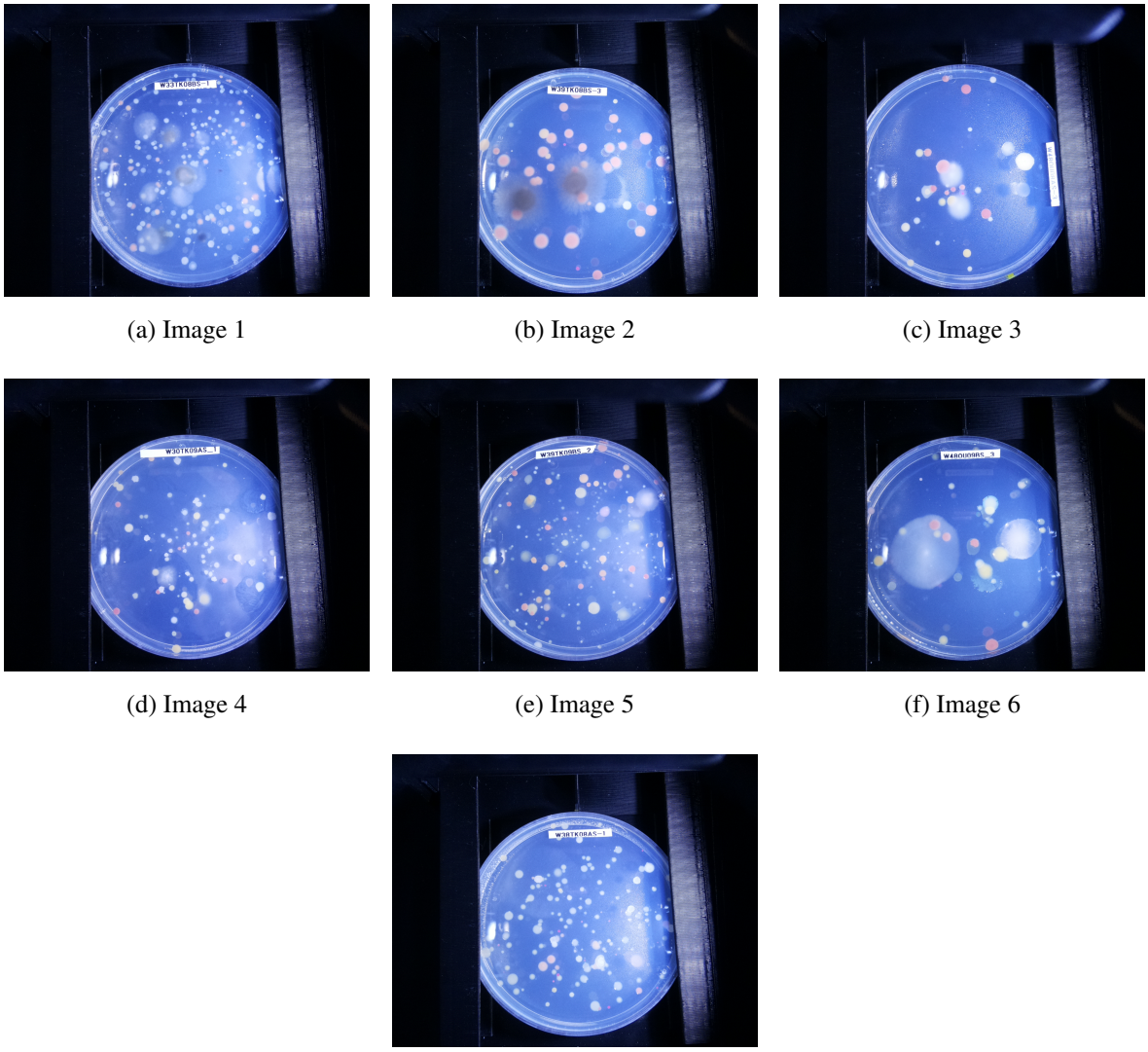
Figure 6: Process of collecting data with NIR, image by Nissinen et al. 2023.

A typical picture was colored, had the size of 1024x1280 px and included the view of a Petri dish, along with any bacteria and fungi colonies present. The total dataset contained nine labeled images, two of which were used for validation. The methods used included two approaches: using the maximum available training data (seven images) and using only two images. This was done with the intention of discovering how reducing the training data affected the performance of the method.

The original data provided contained 15 images, but not all of them were valid. The samples with excessive bacterial growth were invalid and were going to be excluded from the object detection training set. However, one more of the desired features included classifying valid and invalid samples, therefore all the images including invalid ones were used for the training of the classification network.

Another limitation was the various types of bacteria in a dish that occurred in different proportions. Since the classes were unevenly spread in already limited training data, it was decided to concentrate on general bacteria detector and detect a single class called "bacteria colony" instead of differentiating between various colony types. Usually various types of colonies could be recognized by different colors.

The selected data was used to apply various methods and search for optimal solution for object detection. The total available training data is shown in Figure 7. The images display varying quantities of bacterial colonies and levels of fungal contamination. Bacterial colonies are smaller and have more defined, round shapes, whereas large, unstructured white or gray shapes indicate fungi. This adds a layer of complexity to the task as it is crucial not to confuse bacterial colonies with fungal overgrowth. For instance, Image 7 presents clean data without any fungal contamination, while all other images include it to some extent, such as Image 2, which features two large gray shapes.



(a) Image 1 (b) Image 2 (c) Image 3

(d) Image 4 (e) Image 5 (f) Image 6

(g) Image 7

Figure 7: Maximum available object detection training set.

3.2 Ready-made solutions

For the sake of the experiment, it was decided to test the performance of an LLM on such a task. Given the extreme popularity of LLMs and their capability to solve various tasks, it was intriguing to determine whether they could work with this object detection task. The model used for this test was ChatGPT-4. The query "Mark all the bacteria on the image" produced the results displayed in Figure 8.

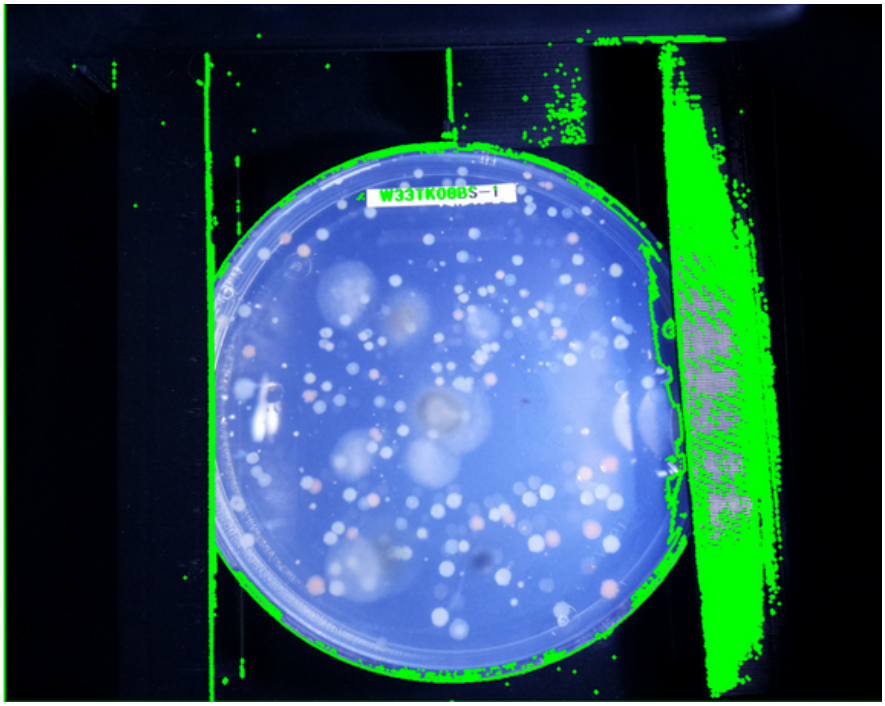


Figure 8: ChatGPT result of "Mark all the bacteria on the image" query.

As is visible from Figure 8, this approach was found to be unsuitable for such a use case; therefore, a more specialized solution had to be created and adapted to our scenario. According to the explanation provided by ChatGPT, the main steps involved were grayscaled, thresholded, and contours detected. This example has demonstrated why special methods must be used for this task and why general-purpose solutions like LLMs do not fit.

3.3 Generative networks

The selected approach was a Variational Autoencoder (VAE), chosen for its relative ease of implementation. However, due to its resource-intensive nature, it was unable to produce high-quality, high-resolution colored images. Another issue was that while this method could add variety to the training data, it did not address the fundamental challenge of annotating the data, which remained the primary difficulty. Even with additional resources, the production of more training data was found to be resource-heavy—as is characteristic of all VAEs—and the task of manually marking the bacteria continued to be a human responsibility. As the result, this straightforward approach to generating additional data was decided to be unsuitable

for the intended use-case.

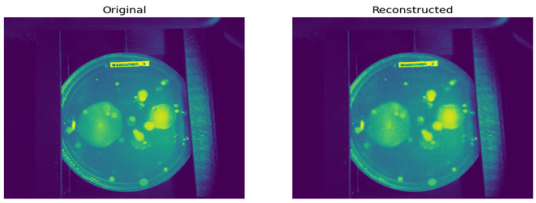


Figure 9: Maximum achieved resolution of 512x512 using VAE.

3.4 Meta-learning

Various implementations of meta-learning algorithms have been developed; however, difficulties were encountered when trying to apply them to real use-cases. Most of the proposed architectures were found too complicated and advanced to be written from scratch, and many of the existing implementations were often outdated, having been updated on average at least four years ago (Meta-R-CNN). This age led to numerous version incompatibilities, making them impossible to be used as-is.

Some of the still-maintained models were found (Meta-Faster-R-CNN, Meta-DETR), but the results obtained on the COCO few-shot dataset were unsatisfactory, leading to the decision to stop the experiments. For example, one of the better-maintained networks, Meta-DETR, required approximately 40 hours for few-shot training on the default training dataset. The general lack of continued development, updates and documentation is a shared problem of multiple meta-learining implementations.

It was concluded that although meta-learning algorithms could be useful, they were not easily adaptable to real-world scenarios. Due to limitations in resources, it was decided that exploring the topic further or implementing original code was not worthwhile.

3.5 Fine-tuning

The fine-tuning approaches described below were based on the TensorFlow Object Detection API. TensorFlow version used was 2.11.0. Although the TensorFlow Object Detection API

had recently been deprecated, it still supported the necessary TensorFlow versions and had extensive documentation, so it was decided to continue the experiments on it.

The selected architecture **SSD ResNet50 V1 FPN 1024x1024** (RetinaNet) (see architecture 4 and more details about RetinaNet at Section 2.2.1) was taken from its Model Garden; the code for it was manually written based on various TensforFlow tutorials and guides. As previously mentioned, Focal Loss is one of the main features of RetinaNet architecture due to its ability to balance positive and negative results and stop easy negative results from the background from overwhelming the training.

Lin et al. 2017 describes the creation of focal loss formula starting from cross entropy as follows:

$$\mathrm{CE}(p,y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise} \end{cases}$$

where $y \in \{\pm 1\}$ is the ground-truth class and $p \in [0,1]$ is the estimated probability for the class with label y = 1. The above formula can be redefined through p

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise} \end{cases}.$$

which allows the previous cross entropy formula to be simplified as $\mathrm{CE}(p,y) = \mathrm{CE}(p_t) = -\log(p_t)$ (since this notation applies to to any value of y). After introducing $\alpha \in [0,1]$ as the weighting factor in order to fix the class imbalance and defining α_t similarly to p_t the α balanced loss can be defined as:

$$\mathrm{CE}(p_t) = -\alpha_t \log(p_t)$$

Even though α balances the positives and negatives, it still gets overwhelmed by large class imbalance. RetinaNet is a dense detector which finds huge amount of potential bounding boxes therefore the huge class imbalance is inevitable and detections contain a lot of easy negative results. To address it, Lin et al. 2017 proposes a new loss function:

$$\mathrm{FL}(p_t) = -(1-p_t)^{\gamma} \log(p_t)$$

where $(1-p_t)^{\gamma}$ is a modulating factor and $\gamma \geq 0$ is the tunable focusing parameter. This formula allows those easy negative results to avoid affecting the total loss. The final version

is the α-balanced variant of the above formula that will be used with sigmoid function for additional stability:

$$\mathrm{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t)$$

ObjectDetection API base configuration describes loss functions used by the network: Sigmoid Focal Classification Loss (same as described above) for classification and WeightedSmoothL1LocalizationLoss for localization loss. All the training used the same configuration with the modifications made only to *post-processing* section where various thresholds and amount of detections were modified according to the test cases. For instance, IOU threshold was set to lower value of 0.1 since the data did not expect overlapping bacteria detections.

In the first stage, the network was pre-trained on the ImageNet dataset (the weights for which were provided by TensorFlow). The next task involved fine-tuning it on a few-shot dataset in various ways.

RetinaNet features two training heads (see Figure 10), where one is a subnet for classification and the other is a subnet for box regression. The experiments below involved fine-tuning these subnetworks.

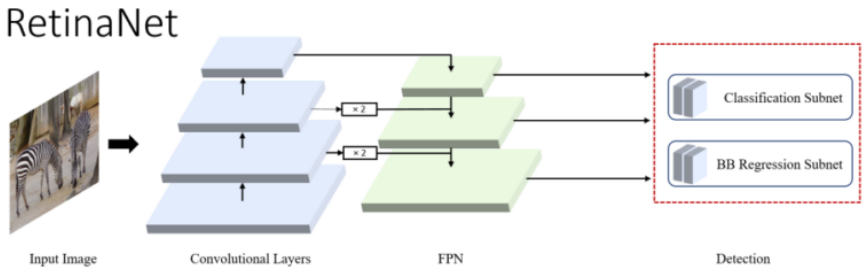


Figure 10: RetinaNet architecture by Zaidi et al. 2021.

Adam optimizer was selected through experimental evaluation for all subsequent fine-tuning processes since it was found to perform better than other optimizers (PMSProp was tested as well). The pre-trained RetinaNet was available in two configurations: 640x640 and 1024x1024 pixels. The version with larger dimension was tested to perform better (as it more closely approximated the original dimensions of the few-shot dataset) and was selected

for all the following experiments. Network trained with larger dimensions doesn't always perform better and it should be selected based on the dimensions of the training data.

The methods described below include various combinations of data-centric strategies used with fine-tuning techniques, aimed at determining the most effective method to perform few-shot bacteria detection. The model configurations described above were same for all the experiments (except for the next Section 3.5.1 which focused on classification task).

3.5.1 Classification fine-tuning

Since one of the subtasks of the thesis involved classifying valid and invalid samples, a fine-tuning approach was used for this as well. The difference between valid and invalid samples lies in the amount of bacteria and fungi growth. Samples with excessively high numbers of either were considered invalid. As this classification task was not the main objective of the thesis, no comparative measurements were performed. The training set consisted of 4 valid and 4 invalid samples, which were used to fine-tune the MobileNetV2 network previously trained on the ImageNet dataset. The training included 100 epochs with the RMSProp optimizer (see 11). After training, the network displayed 100% accuracy on the 9-image test set. No data augmentations or changes were made or necessary.

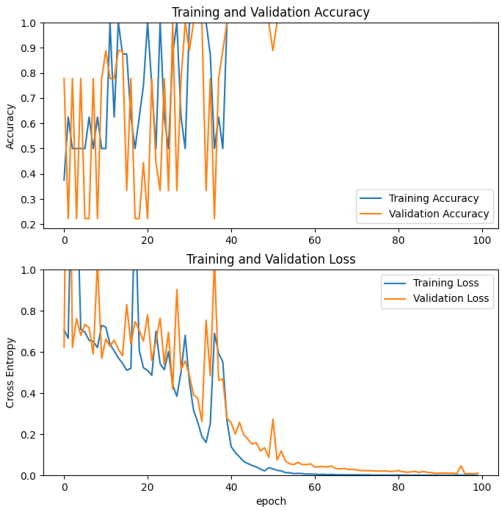


Figure 11: Accuracy and loss of classification task based on transfer learning.

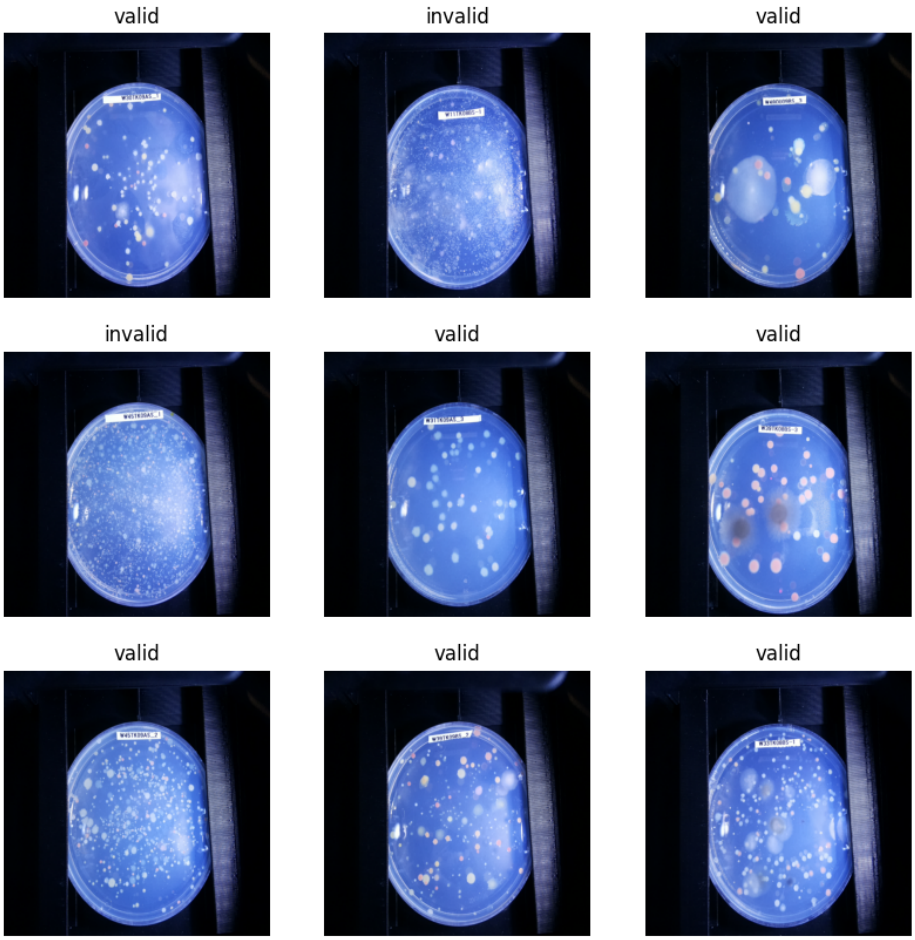


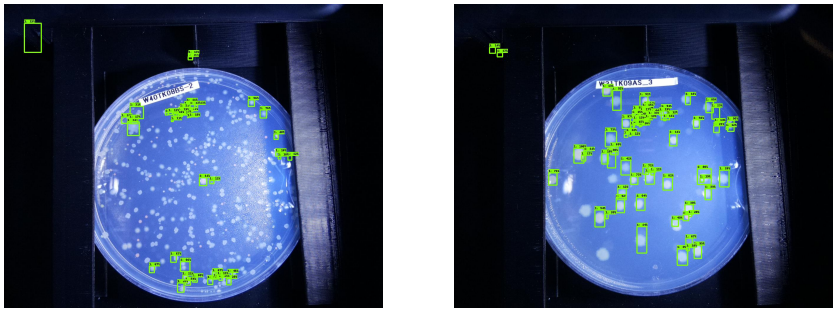
Figure 12: Test results of the classifier of valid/invalid samples

Classifying the images is challenging because the evaluation is highly subjective, and edge cases can be labeled differently by different people. However, the overall accuracy is high, and cases with high confidence scores are likely to be classified correctly. Cases with lower confidence can be flagged for additional human verification. This experiment was used to verify whether the fine-tuning approach is helpful for the classification tasks.

All the next experiments concentrated solely on object detection.

3.5.2 Bacteria detections after fine-tuning of the classification head only

The initial method of fine-tuning used was the simplest approach, which involved training solely the classification head. This training was found to bring surprisingly satisfactory results for simple cases. However, this solution could not be further extended or improved, as the box regression head remained fixed. Nonetheless, for simple cases, this approach could be a valid solution but due to the nature of the target dataset it did not provide sufficient results.



(a) Image 13 test (b) Image 10 test

Figure 13: Partial fine-tuning

3.5.3 Fine-tuning data as-is

The subsequent approach to fine-tuning involved fine-tuning both the classification and box regression heads of the network using the few-shot data without any modifications. Two versions of the dataset were used: a 7-image dataset and a 2-image dataset. According to the results presented in Figure 14 and 15, the model exhibited no signs of overfitting, neither with the 7-image set nor with the 2-image set. This lack of overfitting aligns with the expectations described in section 2.4.3 where fine-tuning is expected to provide some protection from overfitting.

3.5.4 Augmenting data

As previously discussed in Section 2.3.1, data augmentations offer a straightforward method to expand the dataset which is light on resources. This method combines the model-centric

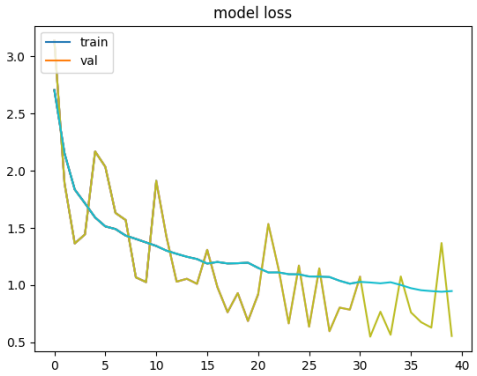


Figure 14: Loss of fine-tuning 7 images as-is.

Figure 15: Loss of fine-tuning 2 images as-is.

fine-tuning solution with a data-centric augmentations approach, re-using the method described in Section 3.5.3 but with additional augmented data.

For this purpose, the Albumentations library was used to generate augmented images (Buslaev et al. 2020). However, not all augmentation methods were equally effective; thus, multiple configurations were tested.

The initial approach involved experimentation with color transformations and image flipping while preserving the size and shape of the image. The augmentations applied included HorizontalFlip, VerticalFlip, and ChannelDropout, each with varying (but always less than 100%) probabilities of application. This method generated 15 augmented images for each original training image, thus expanding the dataset to a total of 105 images for the 7-image scenario and 30 images for the 2-image scenario.

The second approach used all the previously mentioned augmentationsa and added random scaling of the data, aiming to enhance the detection of objects at various sizes. The scaling and cropping parameters were not fixed; when applied, they adjusted the image dimensions to a minimum of 200 pixels and a maximum of the original size, thus making the resulting data much more diverse. The variety of the resulting data can be observed in Figure 16. Since cropping was included in the augmentations, more samples were generated per image to increase the chance of covering the entire area of the source image. Because of this, each training image produced 30 augmented samples, resulting in a training set size of 210.

Such size is typical for object detection training and extends beyond the scope of a few-shot dataset. In the 2-image scenario, a dataset of 60 images was generated.

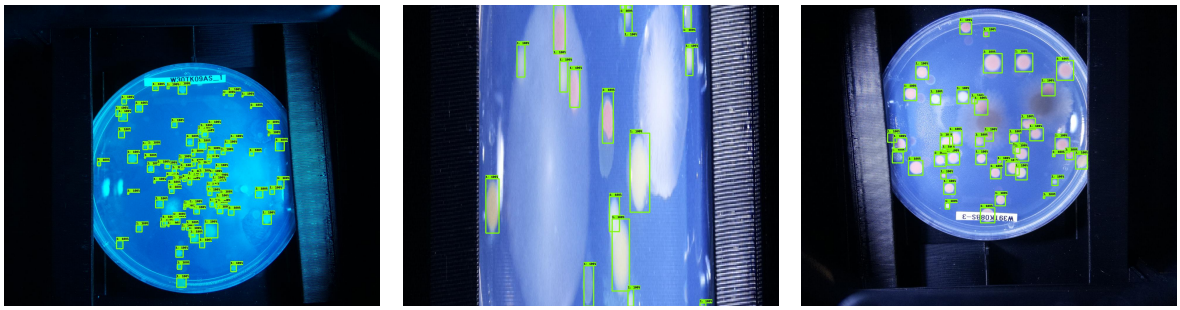


Figure 16: Samples of augmented data using various color, flipping and cropping augmentations.

3.5.5 Close domain data

A close-domain dataset containing 369 images was discovered, and the subsequent set of experiments used various configurations with this close-domain data. The dataset predominantly featured images of Petri dishes, with variations primarily in image quality and coloration. Most images showed minimal bacterial and fungi overgrowth. A representative image from the close-domain dataset is shown in Figure 17. The dataset was published by Solymosi and Nagy 2023.

The initial method involved fine-tuning the network exclusively with these 369 training images, while using the original few-shot validation images to assess performance on the target dataset.

Given the high volume of data, this scenario no longer represented an example of few-shot learning, theoretically permitting the training of an object detector from scratch without prior fine-tuning. However, the objective was to compare the efficacy of the fine-tuning method against various data enhancement strategies, thus, fine-tuning was used for consistency. In addition to that, transfer learning often brings better performance and generalization therefore it's often a good addition even when it's not strictly required (Yosinski et al. 2014).

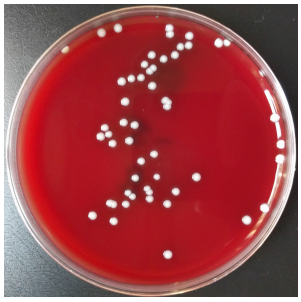


Figure 17: Sample image of domain data.

3.5.6 Mixing the close domain and original data

The next set of experiments used mix of data, combining images from the few-shot and close-domain datasets in each batch in equal proportions. The intention behind this balanced approach was to investigate whether it helps the model in learning more effectively and performing more accurately on few-shot data. Through this method, it was hoped to enhance the model's capacity to adapt to the few-shot dataset. However, this method suffered from the lack of data since the same few-shot images were re-used in every batch. This issue especially strongly affected the 2-image few-shot dataset scenario since the few-shot data within the batch always had duplicates. However, it was decided to accept this limitation in order to keep the same proportion of few-shot and same-domain data for both experiments.

It was hypothesized that such a combination could outperform the methods described in the previous section by better adapting the model to the few-shot dataset.

3.5.7 Augmenting the close domain and original data

The next set of experiments replicated the earlier ones described in Section 3.5.6, with a single modification: images from both datasets were subject to augmentations. The augmentations applied were limited to color modifications and flips, excluding actions like cropping. The list of selected augmentations, each with a probability of less than 1 of being applied, included the following:

- HorizontalFlip
- VerticalFlip

- ChannelDropout
- RandomSunFlare

The intended effect was that by combining close domain and original data, the differences between the few-shot and close domain datasets could be minimized. It also solved the issue of repetitive data within the same batch (for 2-image dataset) and across multiple batches (for both datasets) mentioned in previous Section 3.5.6.

4 Results

The chapter below presents comparative results of the validation data. The results are divided into two categories: those achieved with a training size of 7 images versus 2 images. Some experiments were not repeated as the size of the few-shot dataset was irrelevant for them. The validation images can be viewed in Figure 18.

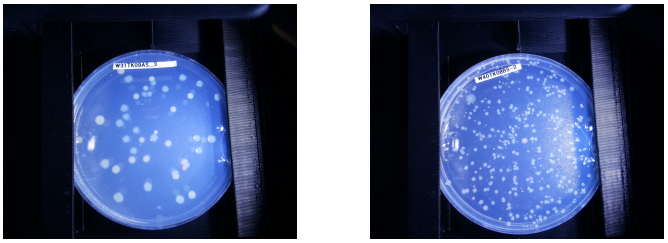


Figure 18: Images 10 and 13 used for validation.

The results in the experiments were measured separately for the two validation images from the Figure 18 above. This distinction was made due to differences in the difficulty of the validation images, with one (Image 10) representing a simpler case and the other (Image 13) a more complicated case. Separate statistics were compiled to allow for a comparison of performance on more and less advanced cases. The training was conducted on the training dataset, which is presented in full size in Figure 7. However, it must be noted that the full 7-image dataset was used only in half of experiments, other experiments used 2 images.

4.1 Metrics interpretation

The selected metrics are standard in the field of object detection and used for comparing performance in various object detection competitions, such as Open Images RVC, COCO Detection Challenge, VOC Challenge, and others (Padilla et al. 2021).

Intersection over Union (IoU) is used to calculate the accuracy of box predictions—specifically, it measures the percentage of the area overlap between the predicted and actual bounding boxes relative to their combined area. It is assumed in this work that two bounding boxes with the highest IoU (with a minimum threshold of 0.5) must be associated with the same

object. While this measurement does not show the total number of correct detections, it illustrates the precision of detecting the location of the object. In addition to that, he post-processing setup for the Object Detection API is configured with an IoU threshold of 0.1 to automatically eliminate overlapping results (this is adapted to the target few-shot data where objects are not supposed to overlap). This configuration influences the overall number of images detected and, consequently, affects the Precision and Recall values, which are standardized across all experiments.

Recall and Precision are calculated based on the identified measurements: True Positives (correctly predicted objects), False Positives (predictions of non-existent objects), and False Negatives (objects that were not detected). Given that there is only one class ("bacteria"), class-specific metrics are not applicable. Finding the optimal performance typically involves optimizing the relationship between recall and precision. An increase in the minimum threshold for detecting an object typically results in a decrease in precision while recall may increase. Missing many positives results in a high recall, as noted by Hicks et al. 2022.

The formulas used to calculate the above values were taken from Padilla et al. 2021:

$$\text{Pr} = \frac{\sum_{n=1}^{S} TP_n}{\sum_{n=1}^{S} TP_n + \sum_{n=1}^{N-S} FP_n} = \frac{\sum_{n=1}^{S} TP_n}{\text{all detections}},$$

$$Rc = \frac{\sum_{n=1}^{S} TP_n}{\sum_{n=1}^{S} TP_n + \sum_{n=1}^{G-S} FN_n} = \frac{\sum_{n=1}^{S} TP_n}{\text{all ground truths}},$$

$$IOU = \frac{\text{area}(B_p \cap B_{\text{gt}})}{\text{area}(B_p \cup B_{\text{gt}})},$$

where the dataset contains G ground-truths and the model that outputs N detections, of which S are correct ($S \leq G$).

The significance of these metrics varies depending on the application. For example, research in the medical field often requires high sensitivity to missing positive results due to the potential dangerous consequences of overlooking them, as mentioned by Hicks et al. 2022. Therefore, high recall is crucial in these contexts.

This thesis is aimed at developing a solution to assist in the detection of bacterial growth

without entirely eliminating manual work, as human verification of the results remains necessary. Therefore, a few missed detections are considered acceptable, and a higher precision is preferred. While some bacteria may be missed, they can still be identified manually, which is more acceptable than a high volume of false predictions since adding missed predictions is expected to be easier job for human than removing the false positives.

4.1.1 Numerical results and table interpretation

The table structure is as following:

1. Setup: stands for the experiment setup, the short descriptions of which are available in Figures 1 and 4.
2. IoU(13): IoU for Image 13 from Figure 18.
3. IoU(10): Same but for Image 10.
4. Precision(13): Precision for Image 13.
5. Precision(10): Precision for Image 10.
6. Recall(13): Recall for Image 13.
7. Recall(10): Recall for Image 10.
8. Other(13): Other statistics for Image 13, specifically the numbers used for Precision/Recall calculation where TP = true positives, FP = false positives and FN = false negatives.
9. Other(10): Statistics for Image 10.
10. Confidence score: Minimum confidence score of the detections.

The cases in Tables 1 and 4 follow the experiments described in the previous Section:

1. Case 1 is described by Section 3.5.3
2. Cases 2 and 3 are described by Section 3.5.4, each representing its own setup of the augmentations, the first case including Color/Noise/Flipping augmentations and the latter one including all of the above together with random cropping
3. Case 4 displays the results for Section 3.5.5 where only the close domain data is trained without including the few-shot dataset into the training.
4. Case 5 is described by the setup in section 3.5.6 where the close domain and few-shot

data is mixed in equal proportions

5. Case 6 is the variation of Case 5 with the addition of augmentations described by Section 3.5.7
6. Case 7 is the fine-tuning of only the classification head described by Section 3.5.2

4.1.2 Tests of 7 image dataset

Test case	Dataset size/ Batch size	Epoch and learning rate	Description
Case 1	7, BS = 2	500, 0.01 LR	No augmentations, training as-is
Case 2	7*15=105, BS = 16	2500, 0.01LR	Color, shape augmentations
Case 3	7*30=210, BS = 16	2500, 0.01 LR	Color, shape augmentations, random size cropping and scaling
Case 4	369, BS = 16	2500, 0.01 LR	Close domain training
Case 5	369+training ds, BS = 16	2500, 0.01 LR	Close domain + original few shot dataset with color augmentations
Case 6	369+training ds, BS = 16	1500, 0.01 LR	Close domain + original few shot dataset
Case 7		400, 0.01LR	Only classifier retrained

Table 1: Description of training configurations of 7-image dataset.

Setup	IoU(13)	IoU(10)	Precision(13)	Precision(10)	Recall(13)	Recall(10)	Other(13)	Other(10)
Case 1	64%	71.8%	18%	8.7%	30.8%	67.4%	TP=118, FP=537, FN=265	TP=31, FP=325,FN=15
Case 2	19%	20.7%	15.1%	13%	30%	80.4%	TP=115,FP=645,FN=268	TP=37,FP=246,FN=9
Case 3	17.8%	22.9%	27.8%	20%	30%	80.4%	TP=117, FP=303, FN=266	TP=37,FP=148, FN=9
Case 4	65.7%	71.6%	28.8%	30.6%	27.9%	89.1%	TP=107,FP=265,FN=276	TP=41, FP=93, FN = 5
Case 5	63.2%	74.7%	32.5%	25.5%	24.5%	80.4%	TP=94, FP=195, FN=289	TP=37, FP=108, FN=9
Case 6	65.2%	78.4%	23.7%	1.10	37.3%	84.8%	TP=143 FP=460 FN=240	TP=39 FP=258 FN=7
Case 7	56%	63.2%	13%	32%	1.6%	54.3%	Tp = 6, FP = 40, FN = 377	TP = 28, FP = 53, FN = 21

Table 2: Validation metrics for 7 image dataset with minimum confidence score 10%.

Setup	IoU(13)	IoU(10)	Precision(13)	Precision(10)	Recall(13)	Recall(10)	Other(13)	Other(10)
Case 1	64.9%	71.8%	50%	53.4%	16.7%	67.4%	TP=64, FP=65, FN=319	TP=31,FP=27, FN=15
Case 2	64.1%	49.6%	44.3%	58%	24.8%	78.2%	TP=95,FP=119,FN=288	TP=36,FP=26,FN=10
Case 3	30.1%	44.3%	61.9%	58.3%	18.2%	76%	TP=70, FP=43, FN=313	TP=35, FP=25, FN=11
Case 4	66.9%	71.6%	53.5%	78.7%	18%	80.4%	TP=69, FP=60,FN=314	TP=37,FP=10,FN=9
Case 5	64.4%	74.8%	66.7%	69.7%	13%	69.6%	TP=50 FP=25 FN=333	TP=32 FP=14 FN=14
Case 6	66.9%	79.3%	56.8%	53.6%	26.1%	80.4%	TP=100 FP=76 FN=283	TP=37 Fp=32 FN=9
Case 7	54.6%	62.6%	20%	42.3%	1.3%	47.8%	TP = 5, FP = 20, FN = 378	TP = 22, FP = 30, FN = 24

Table 3: Validation metrics for 7 image dataset with minimum confidence score 20%

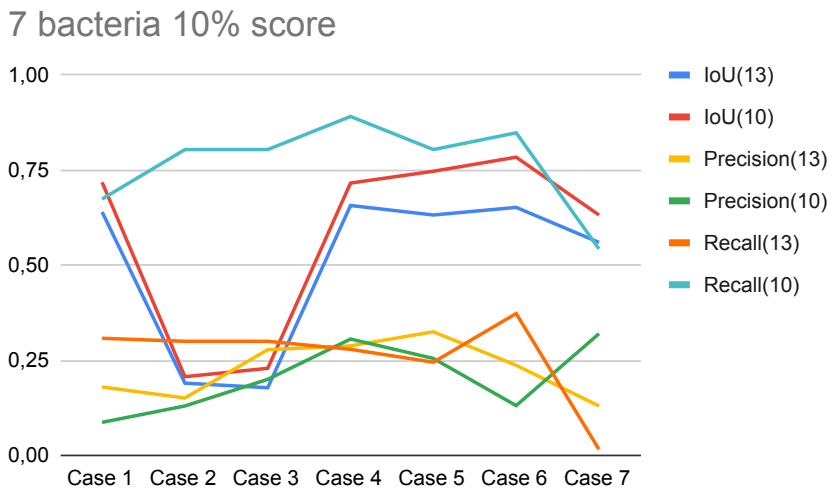


Figure 19: 7 bacteria statistics with 10% threshold

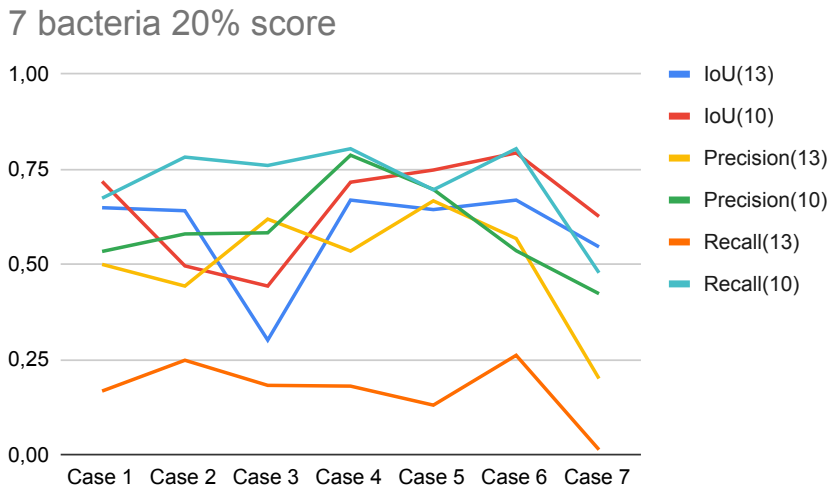


Figure 20: 7 bacteria statistics with 20% threshold

Figure 21: Visualizations of metrics for 7-image dataset.

4.1.3 Tests of 2 image dataset

The following experiments were conducted on a smaller dataset consisting of 2 images; the selected images were 2 and 7, as depicted in Figure 22. All experiments were identical except for the dataset size. Cases 4 and 7, as visible in Table 4, were not included since they were independent of the few-shot dataset size. Case 4 did not utilize the few-shot data at all,

and Case 7 involved re-training only the classifier head.

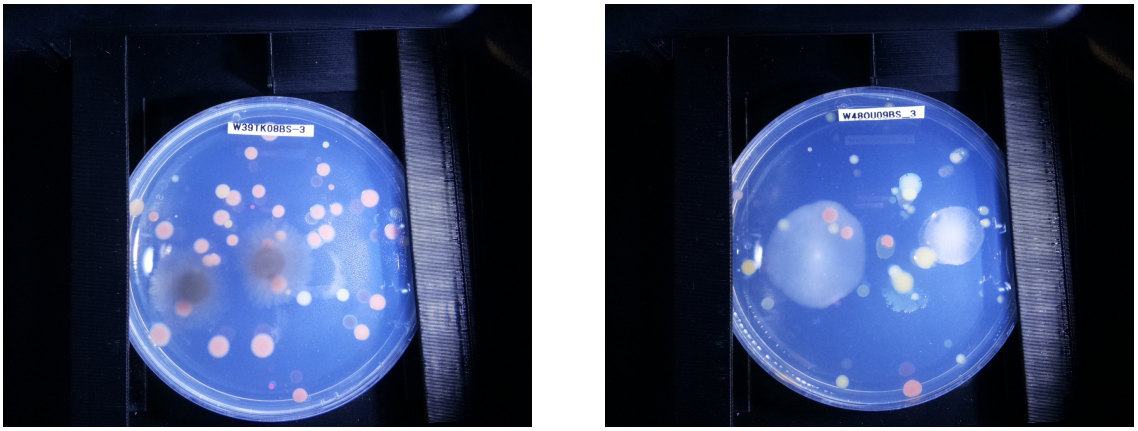


Figure 22: 2-image training set: Image 2 and Image 7.

Test case	Dataset size/ Batch size	Epoch and learning rate	Description
Case 1	2, BS = 2	2500, 0.01 LR	No augmentations, training as-is
Case 2	2*15 = 30, BS = 16	2500, 0.01 LR	Color, shape augmentations
Case 3	2*30 = 60	2500, 0.01 LR	Color, shape augmentations, random size cropping and sclaing
Case 4	-	-	-
Case 5	369+training ds, BS = 16	2500, 0.01 LR	Close domain + original few shot dataset with color augmentations
Case 6	369+training ds, BS = 16	1500, 0.01 LR	Close domain + original few shot dataset
Case 7	-	-	-

Table 4: Description of training configurations of 2-image dataset.

Setup	IoU(13)	IoU(10)	Precision(13)	Precision(10)	Recall(13)	Recall(10)	Other(13)	Other(10)
Case 1	58%	75.9%	10%	29.8%	3.6%	78.3%	TP = 14, FP = 126, FN = 369	TP = 36, FP = 85, FN = 10
Case 2	63%	73.6%	31.3%	28%	14%	78.2%	TP = 52, FP = 114, FN = 331	TP = 36, FP = 94, FN = 10
Case 3	62.6%	75.5%	18.1%	27%	7%	82.6%	TP = 27, FP = 122, FN = 356	TP = 38. FP = 104, FN = 8
Case 5	65.8%	72%	42.2%	32.8%	24%	87%	TP=92 FP=126 FN=291	TP=40 FP=82 FN=6
Case 6	64.7%	77.8%	26.4%	18.6%	26.6%	84.8%	TP=102 FP=285 FN=281	TP=39 FP=171 FN=7

Table 5: Validation metrics for 2 image dataset with minimum confidence score 10%.

Setup	IoU(13)	IoU(10)	Precision(13)	Precision(10)	Recall(13)	Recall(10)	Other(13)	Other(10)
Case 1	62.5%	76.5%	33%	77.8%	1.6%	76%	TP = 6, FP =12, FN = 377	TP = 35, FP = 10, FN = 11
Case 2	62.1%	73.8%	43.5%	69%	4.4%	76%	TP = 17, FP = 22, FN = 366	TP = 35, FP = 16, FN = 11
Case 3	67.2%	75.6%	27.6%	71.2%	2%	80%	TP = 8, FP = 21, FN = 375	TP = 37, FP = 15, FN = 9
Case 5	67.4%	73%	67.2%	80.5%	10%	71.7%	TP=39 FP=19 FN=344	TP=33 FP=8 FN=13
Case 6	67.2%	77.9%	65.1%	70.3%	14.6%	82.6%	TP=56 FP=30 FN=327	TP=38 FP=16 FN=8

Table 6: Validation metrics for 2 image dataset with minimum confidence score 20%.

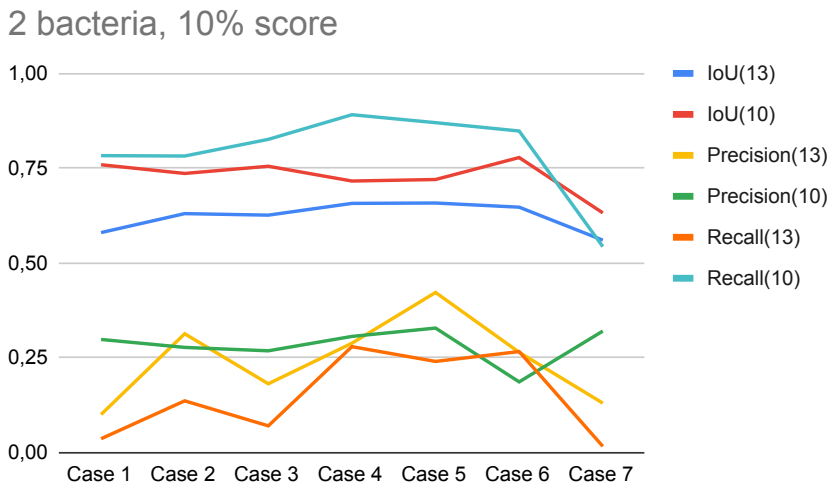


Figure 23: 2 bacteria statistics with 10% threshold.

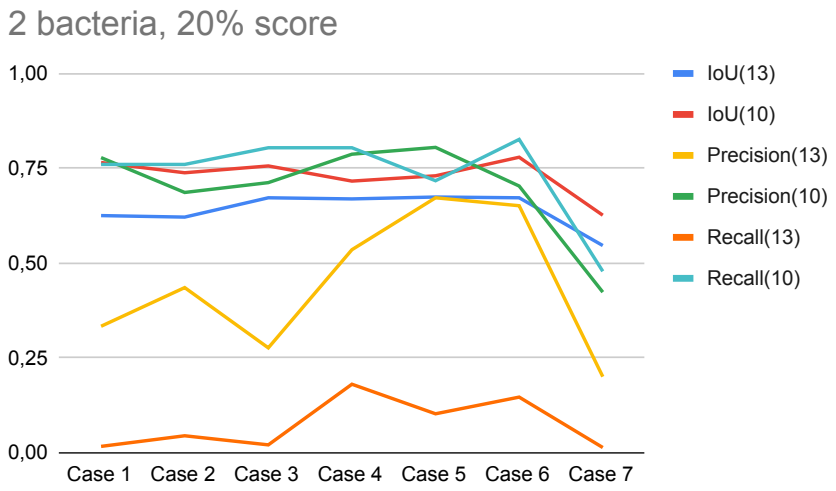


Figure 24: 2 bacteria statistics with 20% threshold.

Figure 25: Visualizations of metrics for 2-image dataset.

4.2 Test dataset results

The tables above present the validation statistics collected based on two images of varying difficulty. To clarify the comparison, solutions with better evaluation measurements were selected to demonstrate their performance on the test data. Both the 7-image and 2-image datasets delivered higher precision results when a minimum threshold of 20% was utilized;

therefore, the data below is shown only for that threshold. The test set contains more images than those displayed, the selected images are made to represent various difficulty levels of the data.

4.2.1 Seven image dataset results

According to the validation set metrics from Table 3, Case 4 was found to have the highest precision, followed by Cases 5, 3, and 2. However, a significant drop in the IoU was observed for Case 3. Given the small size of the validation set, it is reasonable to compare all four potential best candidates against the test set. An empirical check of the results indicated that Case 2 was the best candidate, followed by Cases 4, 3, and 5.

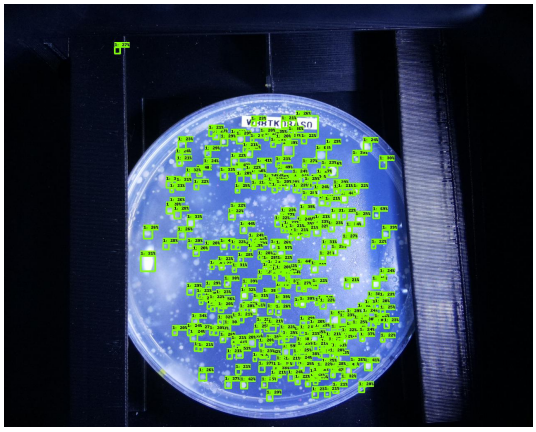


Figure 26: Case 2 Image 11

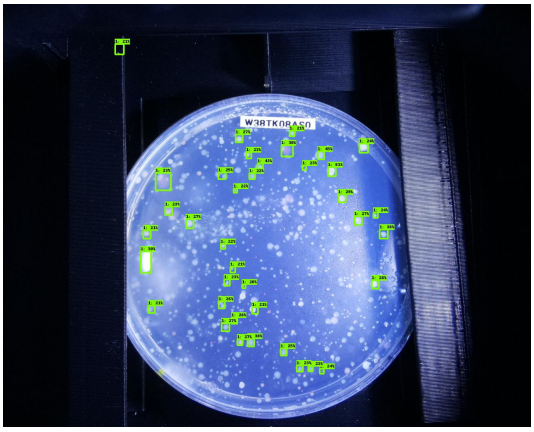


Figure 27: Case 3 Image 11

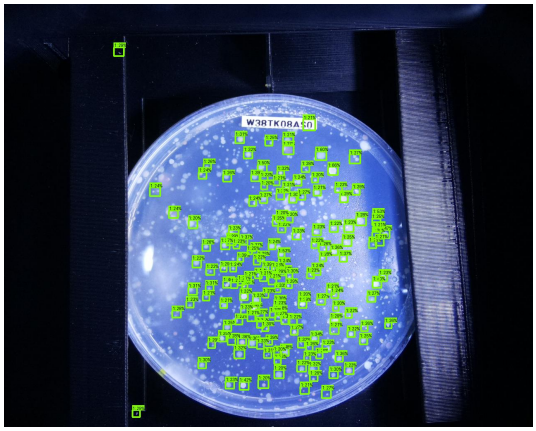


Figure 28: Case 4 Image 11

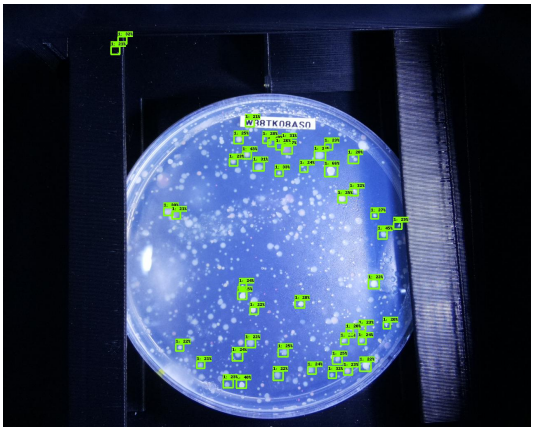


Figure 29: Case 5 Image 11

Figure 30: Comparisons of the predictions on Image 11 for 7-image dataset.

The general performance on the simple cases was found to be satisfactory across all training scenarios:

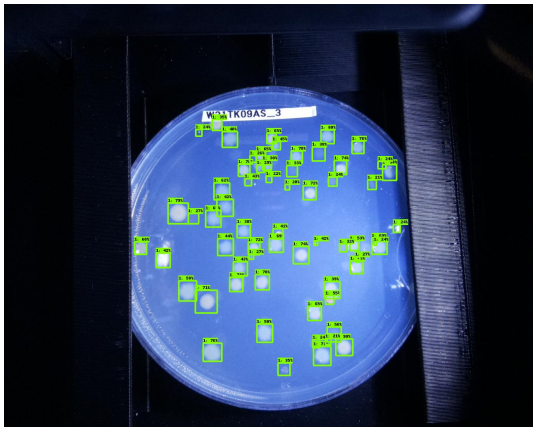


Figure 31: Case 2 Image 10

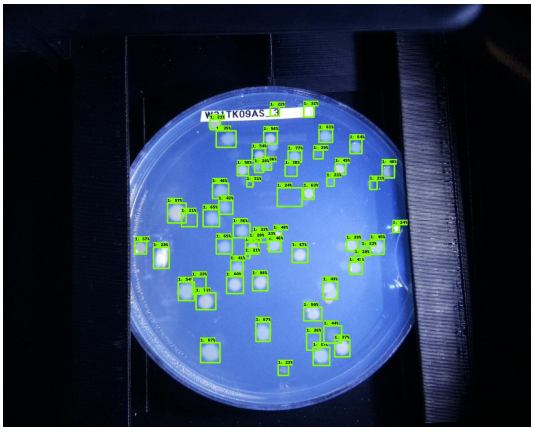


Figure 32: Case 3 Image 10

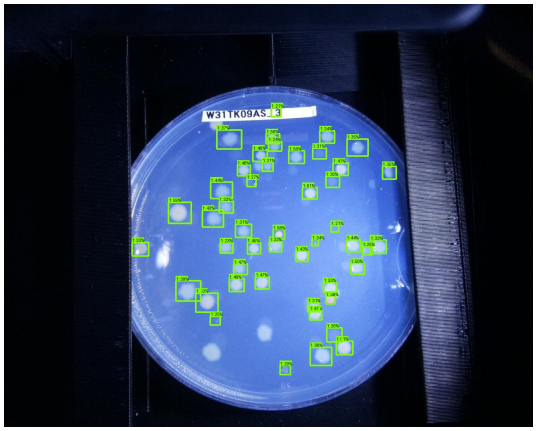


Figure 33: Case 4 Image 10

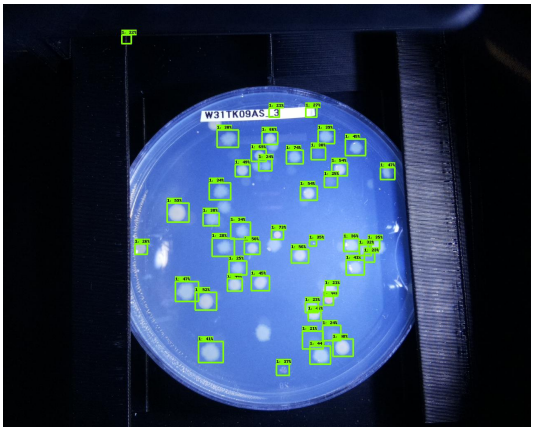


Figure 34: Case 5 Image 10

Figure 35: Comparisons of the predictions on Image 10 (which is also part of validation set) for 7-image dataset.

4.2.2 Two image dataset results

The best performing results for the case of the 2-image dataset, based on the data in Figure 25, was observed as follows: Case 5, Case 6, Case 4, Case 2, etc.

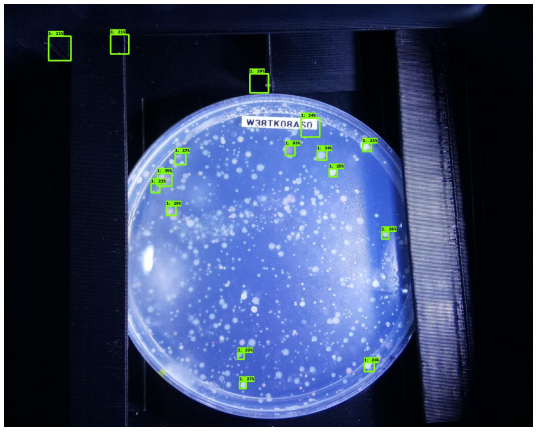


Figure 36: Case 2 Image 11

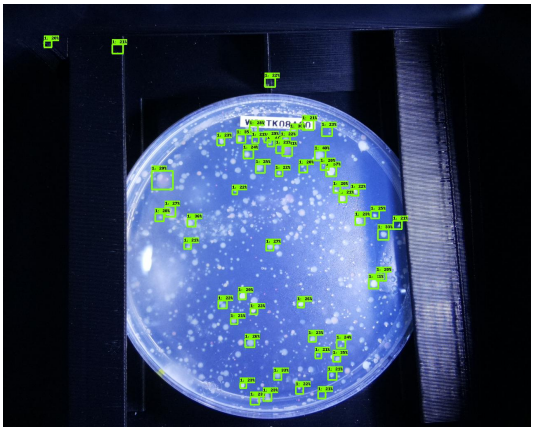


Figure 37: Case 6 Image 11

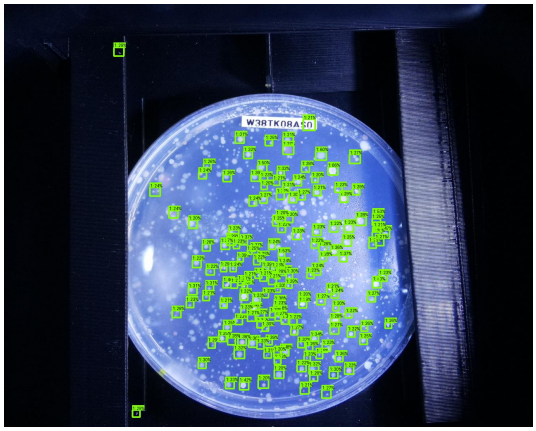


Figure 38: Case 4 Image 11

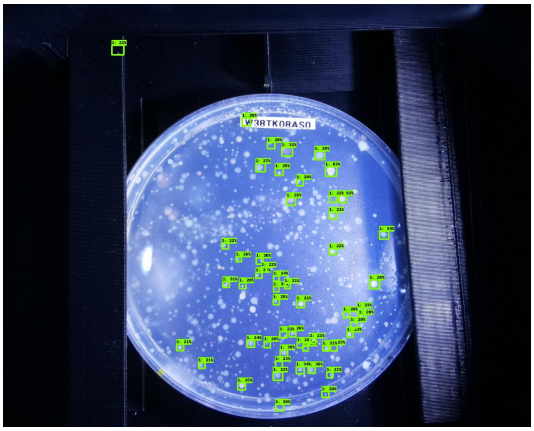


Figure 39: Case 5 Image 11

Figure 40: Comparisons of the predictions on Image 11 for 2-image dataset.

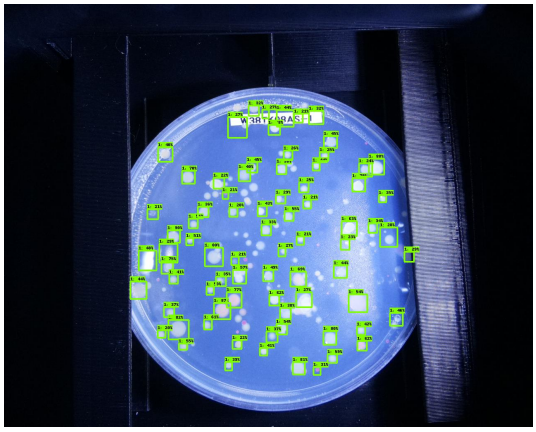


Figure 41: Case 2

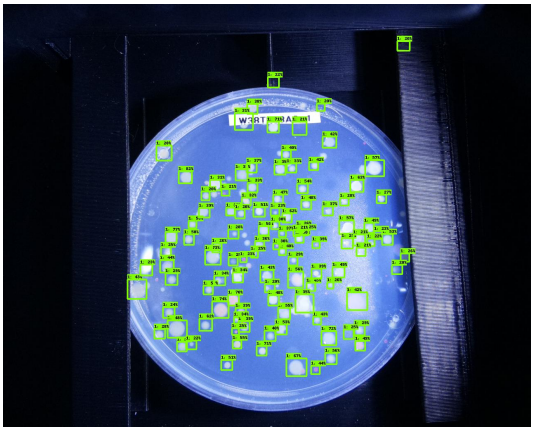


Figure 42: Case 6

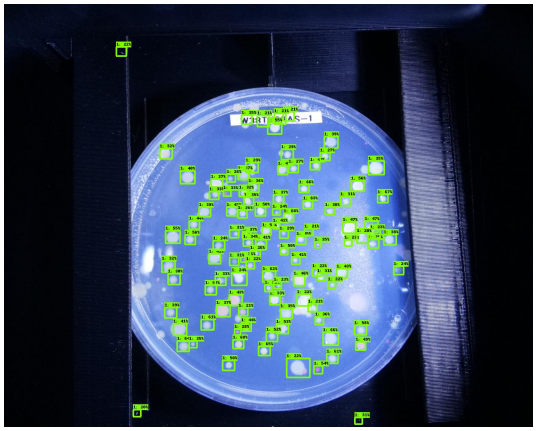


Figure 43: Case 4 Image 12

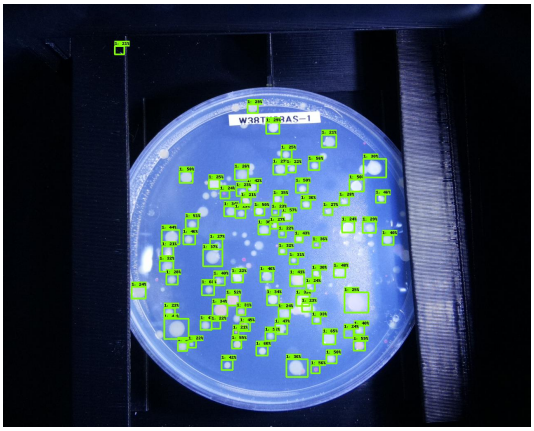


Figure 44: Case 5 Image 12

Figure 45: Comparisons of the predictions on Image 12 for 2-image dataset.

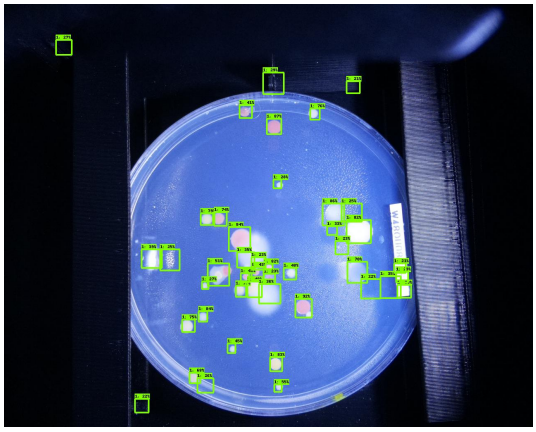


Figure 46: Case 2 Image 3

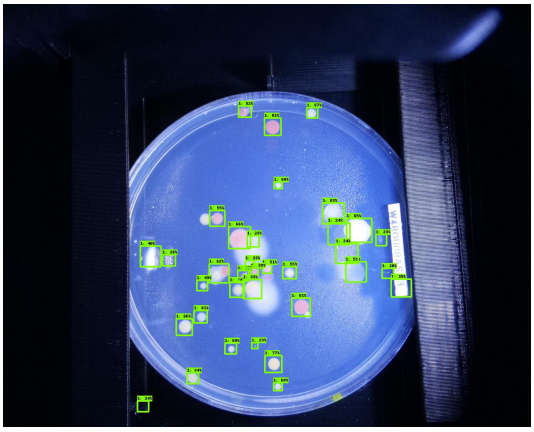


Figure 47: Case 6 Image 3

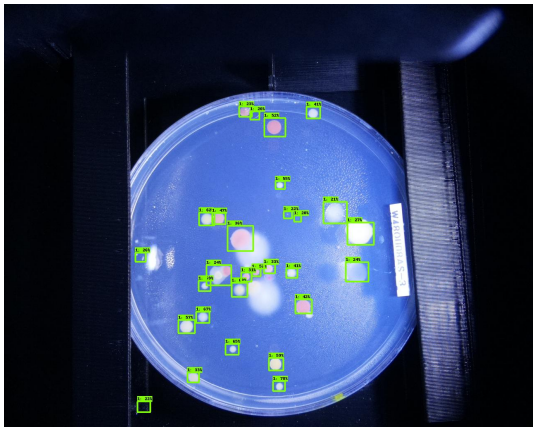


Figure 48: Case 4 Image 3

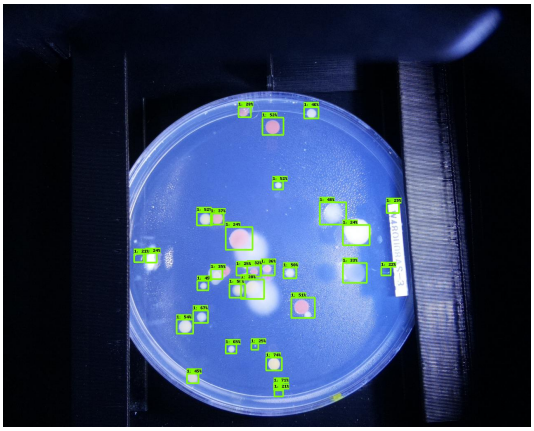


Figure 49: Case 5 Image 3

Figure 50: Comparisons of the predictions on Image 3 on 2-image dataset.

5 Discussion

5.1 Results discussion

The findings outlined in Sections 4.2.1 and 4.2.2, along with the validation metrics from Figures 25 and 21, have led to several conclusions. As anticipated, training with 7 images was generally observed to bring higher and more efficient results.

Several facts were revealed by the comparison between the results in 7-image and 2-image validation measurements:

- The recall for 2-image training was found to be lower than for 7-image training, likely due to fewer true positives being detected, which indicates a generally less effective detector.
- No sudden IoU drops were observed in 2-image training for Cases 2-3, unlike in 7-image training.
- The benefits of incorporating close-domain data were evident in 2-image training, while not being as critical in the case of 7-image dataset.
- Cropping augmentations did not improve the performance of detectors in both 2- and 7-image cases, despite the fact that they significantly increased the size of the datasets.
- The larger 7-image dataset was observed to receive more benefits from augmentations than the smaller 2-image dataset.
- While the 2-image dataset struggled to perform well using only its own few-shot data in more complicated images, it still managed to achieve good detections for simpler cases, as depicted in 50.

A notable observation from the close-domain training was that the integration of close-domain data with a few-shot dataset did not yield improvements. It was initially hypothesized that this combination would enhance adaptation to the few-shot dataset; however, training was actually slowed due to the high frequency of repetitions in each batch, given the relatively small size of the few-shot dataset. This suggests that special adaptations to the few-shot dataset are unnecessary, as the performance was improved by the close-domain

data alone. Adding the augmentations into few-shot and close domain data like in Case 5 did not bring the improvements either.

The differing behaviors observed between the 7-image and 2-image dataset training might prompt the question: Can the 7-image dataset still be considered a few-shot dataset, given its noticeably higher performance? I would argue that it still qualifies as a few-shot dataset, since the amount of data remains lower than what is typically required for object detection models. For instance, the same-domain dataset used in scenarios 4-6 contains over 200 images.

5.2 Limitations and Future research

This thesis explored several methods of few-shot object detection but did not include a comprehensive analysis of all potential approaches. For instance, it excluded older and more sophisticated techniques such as meta-learning, primarily due to the high computational resources most meta-learning algorithms require. Additionally, many existing meta-learning implementations suffer from outdated code and dependencies, further complicating their use. For this reason, most approaches used in this thesis are data-centric. Future research could expand this work by conducting a deeper review and comparison of model-centric solutions, specifically incorporating meta-learning techniques.

This thesis does not implement any few-shot detection method with perfect accuracy; rather, it seeks to develop its own approach based on the task at hand. For example, although the approach described by Wang et al. 2020 allows for detecting both novel and base classes, the implementation in this thesis did not prioritize base classes since they are not relevant to the application. Therefore, this solution may lack reusability since it is tailored to the practical task where preserving the detection of base classes is not critical.

The selected model-centric fine-tuning approach could be further improved by experimenting with various backbones and training configurations, such as optimizers and learning rates.

Since this thesis included object detection of only one class ("bacteria") for the sake of simplicity, future work could include multi-class bacteria detection and classification. In the case

of the methods used in this thesis it would mean that classifier part of the network would be fine-tuned on multiple classes of bacteria colonies instead of one.

When applied in a practical setting, this approach could also benefit from the use of active learning, where users would annotate missed detections. These annotations could then be incorporated into a new training dataset, gradually improving data scarcity and eventually constructing a more adequate dataset, as suggested (Brust, Käding, and Denzler 2018). This strategy is particularly applicable to the current use-case, where objects will be eventually marked manually in the event of inaccurate predictions.

The performance of the selected approach was evaluated using a specialized few-shot dataset rather than standard datasets like COCO or Pascal VOC, making direct comparisons with other solutions challenging. Nevertheless, as most studies do not report results from real-life data, the metrics gathered here are considered sufficiently informative.

This work can be compared with other research aimed at the same task of bacteria detection. Multiple approaches include using larger datasets such as AGAR (Majchrowska, Pawłowski, et al. 2021b), which contains around 18,000 images. Since the close-domain training worked well in the experiments, it is safe to assume that utilizing an even larger dataset for this thesis would bring even better performance. Much research utilizes large datasets such as AGAR with various deep learning approaches using CNNs (Majchrowska, Pawłowski, et al. 2021a; Yang et al. 2023).

Some research employs similar methods; for instance, El-Melegy, Mohamed, and Elmelegy 2019 used transfer learning to train a detector for tuberculosis bacteria. Their research utilized a dataset of 500 images with augmentations to increase the amount of training data, resulting in a total of 1,500 images. This approach achieved a recall and precision of 98.3% and 82.6%, respectively, for detecting TB bacilli.

Generally, most approaches to this problem involve using large datasets. However, the fact that data augmentations provided comparatively satisfactory results indicates that it is a valid alternative approach. Not all areas have access to large datasets; therefore, it is valuable that this thesis included experiments based on the assumption that no additional data could be added. While object detection can be effectively performed using large datasets, the few-

shot methods researched here still provided valuable insights for scenarios where additional data cannot be collected.

6 Conclusion

The problem of few-shot object detection has been studied previously, but most of the research and statistics have been conducted on generated few-shot datasets instead of real-life training data. This thesis tests various approaches in real-world scenario in order to identify the most helpful methods and provide recommendations for approaching similar tasks.

As discussed in the Results chapter (see 5.1), Precision is the more important metric for the selected task since it is preferable to avoid false detections. Many conclusions are built on this assumption. It is also impossible to compare these values straightforwardly to other solutions since their metrics are based on different datasets. Therefore, the results of various experiments completed in this thesis will be compared only among themselves.

The thesis provides the following answers to the research questions:

1. What are the most effective methods for few shot detection in the selected domain? The simplest approach to training on few-shot data and avoiding overfitting is to use fine-tuning techniques. To achieve better results, the next step should include improving the data itself. The general guidance is to always include close-domain data if available. If it is not available, extending the few-shot dataset using data augmentations can help. However, the augmentations should not be overdone, and the resulting data should not be too different from the expected test data. Fine-tuning the pre-trained model with improved data brings sufficient performance. Another approach that can help mostly in simple cases is fine-tuning only classification head of the network.
2. What role do data augmentation and synthetic data generation play in improving the performance of few-shot object detection models for bacteria colony identification? Data augmentations play a significant role and can noticeably improve the quality and variety of the dataset, significantly improving the quality of the model predictions. Data generation techniques are harder to implement, require more resources, and do not eliminate the manual labor of labeling the data. Therefore data augmentations are the preferable approach.
3. Can transfer learning from related domains enhance the accuracy and ability to gener-

alize of few-shot object detection models in identifying bacteria colonies? Yes, using close-domain data significantly enhances object detection performance and might even eliminate the need for using the few-shot dataset entirely.

The recommended approach is to use fine-tuning rather than meta-learning because of the method's simplicity and its modest resource requirements, coupled with some data improvement methods.

The practical purpose of this thesis is to classify approaches for improving few-shot object detection and to help researchers and developers choose the optimal solution. This thesis does not develop a novel theory; instead, it aims to apply the existing theories to the practical problem of object detection with few-shot data. The field of few-shot learning includes various approaches that differ significantly in difficulty, some of which are extremely challenging to implement or require substantial resources. Therefore, this comparative analysis can help users select the most practical solution for their specific use case.

Bibliography

Antonelli, Simone, D. Avola, L. Cinque, Donato Crisostomi, G. Foresti, Fabio Galasso, M. Marini, Alessio Mecca, and D. Pannone. 2022. "Few-Shot Object Detection: A Survey." *ACM Comput. Surv.,* https://www.semanticscholar.org/paper/Few-Shot-Object-Detection%3A-A-Survey-Antonelli-Avola/ec6905bae8024ceb586ec322b4f1e1f8193cf810.

Brust, Clemens-Alexander, Christoph Käding, and Joachim Denzler. 2018. "Active learning for deep object detection." *arXiv preprint arXiv:1809.09875.*

Buslaev, Alexander, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. 2020. "Albumentations: Fast and Flexible Image Augmentations." *Information* 11 (2). ISSN: 2078-2489. https://doi.org/10.3390/info11020125. https://www.mdpi.com/2078-2489/11/2/125.

Casado-García, Ángela, César Domínguez, Jónathan Heras, Eloy Mata, and Vico Pascual. 2019. "The Benefits of Close-Domain Fine-Tuning for Table Detection in Document Images." *arXiv* (December). https://doi.org/10.48550/arXiv.1912.05846. eprint: 1912.05846.

Chadebec, Clément, and Stéphanie Allassonnière. 2021. "Data Generation in Low Sample Size Setting Using Manifold Sampling and a Geometry-Aware VAE." *CoRR.*

Chadebec, Clément, Elina Thibeau-Sutre, Ninon Burgos, and Stéphanie Allassonnière. 2022. "Data Augmentation in High Dimensional Low Sample Size Setting Using a Geometry-Based Variational Autoencoder." *IEEE Trans. Pattern Anal. Mach. Intell.* 45, no. 3 (June): 2879–2896. https://doi.org/10.1109/TPAMI.2022.3185773.

Franz, Ole, Heikki Häkkänen, Salla Kovanen, Kati Heikkilä-Huhta, Riitta Nissinen, and Janne A. Ihalainen. 2023. "NIRis: A low-cost, versatile imaging system for NIR fluorescence detection of phototrophic cell colonies used in science and education." *bioRxiv* (June): 2023.05.31.543100. eprint: 2023.05.31.543100. https://doi.org/10.1101/2023.05.31.543100.

Giannone, Giorgio, Didrik Nielsen, and Ole Winther. 2022. "Few-Shot Diffusion Models." *arXiv* (May). https://doi.org/10.48550/arXiv.2205.15463. eprint: 2205.15463.

Han, Jiaming, Yuqiang Ren, Jian Ding, Ke Yan, and Gui-Song Xia. 2023. “Few-Shot Object Detection via Variational Feature Aggregation.” *arXiv* (January). https://doi.org/10.48550/arXiv.2301.13411. eprint: 2301.13411.

Hicks, Steven A., Inga Strümke, Vajira Thambawita, Malek Hammou, Michael A. Riegler, Pål Halvorsen, and Sravanthi Parasa. 2022. “On evaluation metrics for medical applications of artificial intelligence.” *Sci. Rep.* 12. https://doi.org/10.1038/s41598-022-09954-8.

Hospedales, Timothy, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2021. “Meta-Learning in Neural Networks: A Survey.” *IEEE Trans. Pattern Anal. Mach. Intell.* 44, no. 9 (May): 5149–5169. https://doi.org/10.1109/TPAMI.2021.3079209.

Huisman, Mike, Jan N. van Rijn, and Aske Plaat. 2021. “A survey of deep meta-learning.” *Artif. Intell. Rev.* 54, no. 6 (August): 4483–4541. ISSN: 1573-7462. https://doi.org/10.1007/s10462-021-10004-4.

Karlinsky, Leonid, Joseph Shtok, Sivan Harary, Eli Schwartz, Amit Aides, Rogerio Feris, Raja Giryes, and Alex M. Bronstein. 2019. “RepMet: Representative-Based Metric Learning for Classification and Few-Shot Object Detection.” In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* June.

Kee, Jack Sheng, Swee Yin Lim, Agampodi Promoda Perera, Yong Zhang, and Mi Kyoung Park. 2013. “Plasmonic nanohole arrays for monitoring growth of bacteria and antibiotic susceptibility test.” *Sens. Actuators, B* 182 (June): 576–583. ISSN: 0925-4005. https://doi.org/10.1016/j.snb.2013.03.053.

Koch, Gregory, Richard Zemel, Ruslan Salakhutdinov, et al. 2015. “Siamese neural networks for one-shot image recognition.” In *ICML deep learning workshop,* vol. 2. 1. Lille.

Kulis, Brian, et al. 2013. “Metric learning: A survey.” *Foundations and Trends® in Machine Learning* 5 (4): 287–364.

Lin, Tsung-Yi, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2016. “Feature Pyramid Networks for Object Detection.” *arXiv* (December). https://doi.org/10.48550/arXiv.1612.03144. eprint: 1612.03144.

Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. "Focal Loss for Dense Object Detection." *arXiv* (August). https://doi.org/10.48550/arXiv.1708.02002. eprint: 1708.02002.

Majchrowska, Sylwia, Jarosław Pawłowski, Natalia Czerep, Aleksander Górecki, Jakub Kuciński, and Tomasz Golan. 2021a. "Deep neural networks approach to microbial colony detection – a comparative analysis." *arXiv* (August). https://doi.org/10.1007/978-3-031-11432-8_9. eprint: 2108.10103.

Majchrowska, Sylwia, Jarosław Pawłowski, Grzegorz Guła, Tomasz Bonus, Agata Hanas, Adam Loch, Agnieszka Pawlak, Justyna Roszkowiak, Tomasz Golan, and Zuzanna Drulis-Kawa. 2021b. *AGAR a microbial colony dataset for deep learning detection.* arXiv: 2108.01234 [cs.CV].

El-Melegy, Moumen, Doaa Mohamed, and Tarek Elmelegy. 2019. "Automatic Detection of Tuberculosis Bacilli from Microscopic Sputum Smear Images Using Faster R-CNN, Transfer Learning and Augmentation," 270–278. September. ISBN: 978-3-030-31331-9. https://doi.org/10.1007/978-3-030-31332-6_24.

Musgrave, Kevin, Serge Belongie, and Ser-Nam Lim. 2020. "A metric learning reality check." In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16,* 681–699. Springer.

Nissinen, Riitta, Ole Franz, Salla Kovanen, Meri Mäkelä, Venla Kraft, Katri Ketola, Alli Liukkonen, Kati Heikkilä-Huhta, Heikki Häkkänen, and Janne A. Ihalainen. 2023. "Aerobic anoxygenic phototrophic bacteria are ubiquitous in phyllo- and endosphere microbiomes of boreal and subarctic plants." *bioRxiv* (February): 2023.02.19.529139. eprint: 2023.02.19.529139. https://doi.org/10.1101/2023.02.19.529139.

O'Shea, Keiron, and Ryan Nash. 2015. "An Introduction to Convolutional Neural Networks." *arXiv* (November). https://doi.org/10.48550/arXiv.1511.08458. eprint: 1511.08458.

Padilla, Rafael, Wesley L. Passos, Thadeu L. B. Dias, Sergio L. Netto, and Eduardo A. B. da Silva. 2021. "A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit." *Electronics* 10, no. 3 (January): 279. ISSN: 2079-9292. https://doi.org/10.3390/electronics10030279.

Pawłowski, Jarosław, Sylwia Majchrowska, and Tomasz Golan. 2022. "Generation of microbial colonies dataset with deep learning style transfer." *Sci. Rep.* 12, no. 5212 (March): 1–12. ISSN: 2045-2322. https://doi.org/10.1038/s41598-022-09264-z.

Pothuganti, Swathi. 2018. "Review on over-fitting and under-fitting problems in Machine Learning and solutions." *Int. J. Adv. Res. Electr. Electron. Instrum. Eng* 7:3692–3695.

Robb, Esther, Wen-Sheng Chu, Abhishek Kumar, and Jia-Bin Huang. 2020. "Few-Shot Adaptation of Generative Adversarial Networks." *arXiv* (October). https://doi.org/10.48550/arXiv.2010.11943. eprint: 2010.11943.

Santoro, Adam, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. "Meta-learning with memory-augmented neural networks." In *International conference on machine learning,* 1842–1850. PMLR.

Sikka, Metika. 2021. "Balancing the Regularization Effect of Data Augmentation." *Medium* (December). ISSN: 5514-8374. https://towardsdatascience.com/balancing-the-regularization-effect-of-data-augmentation-eb551be48374.

Snell, Jake, Kevin Swersky, and Richard S. Zemel. 2017. "Prototypical Networks for Few-shot Learning." *arXiv* (March). https://doi.org/10.48550/arXiv.1703.05175. eprint: 1703.05175.

Solymosi, Norbert, and Sára Nagy. 2023. "Annotated dataset for deep-learning-based bacterial colony detection" (April). https://doi.org/10.6084/m9.figshare.22022540.v3. https://figshare.com/articles/dataset/Annotated_dataset_for_deep-learning-based_bacterial_colony_detection/22022540.

Vail, J. H., R. Morgan, C. R. Merino, F. Gonzales, R. Miller, and J. L. Ram. 2003. "Enumeration of Waterborne Escherichia coli with Petrifilm Plates." *J. Environ. Qual.* 32, no. 1 (January): 368–373. ISSN: 0047-2425. https://doi.org/10.2134/jeq2003.3680.

Wang, Xin, Thomas E. Huang, Trevor Darrell, Joseph E. Gonzalez, and Fisher Yu. 2020. "Frustratingly Simple Few-Shot Object Detection." *arXiv* (March). https://doi.org/10.48550/arXiv.2003.06957. eprint: 2003.06957.

What Is Few-Shot Learning? | IBM. 2024. [Online; accessed 19. Apr. 2024], April. https://www.ibm.com/topics/few-shot-learning.

Wu, Jianxin. 2017. "Introduction to convolutional neural networks."

Yan, Xiaopeng, Ziliang Chen, Anni Xu, Xiaoxi Wang, Xiaodan Liang, and Liang Lin. 2019. "Meta r-cnn: Towards general solver for instance-level low-shot learning." In *Proceedings of the IEEE International Conference on Computer Vision,* 9577–9586.

Yang, Fan, Yongjie Zhong, Hui Yang, Yi Wan, Zhuhua Hu, and Shengsen Peng. 2023. "Microbial Colony Detection Based on Deep Learning." *Applied Sciences* 13 (19). ISSN: 2076-3417. https://doi.org/10.3390/app131910568. https://www.mdpi.com/2076-3417/13/19/10568.

Yosinski, Jason, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. "How transferable are features in deep neural networks?" *Advances in Neural Information Processing Systems* 27. https://proceedings.neurips.cc/paper_files/paper/2014/hash/375c71349b295fbe2dcdca9206f20a06-Abstract.html.

Zaidi, Syed Sahil Abbas, Mohammad Samar Ansari, Asra Aslam, Nadia Kanwal, Mamoona Asghar, and Brian Lee. 2021. "A Survey of Modern Deep Learning based Object Detection Models." *arXiv* (April). https://doi.org/10.48550/arXiv.2104.11892. eprint: 2104.11892.

Zhang, Chengcui, Wei-Bang Chen, Wen-Lin Liu, and Chi-Bang Chen. n.d. "An Automated Bacterial Colony Counting System." In *2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (sutc 2008),* 11–13. IEEE. ISBN: 978-0-7695-3158-8. https://doi.org/10.1109/SUTC.2008.50.

Zou, Zhengxia, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. 2019. "Object Detection in 20 Years: A Survey." *arXiv* (May). https://doi.org/10.48550/arXiv.1905.05055. eprint: 1905.05055.