TU WIEN Informatics

Flower State Classification for Watering System

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Tobias Eidelpes, BSc
Matrikelnummer 01527193

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.-Prof. Dr. Horst Eidenberger

Wien, 20. Februar 2023

Tobias Eidelpes	Horst Eidenberger

TU WIEN Informatics

Flower State Classification for Watering System

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Tobias Eidelpes, BSc
Registration Number 01527193

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.-Prof. Dr. Horst Eidenberger

Vienna, 20th February, 2023

Tobias Eidelpes	Horst Eidenberger

Erklärung zur Verfassung der Arbeit

Tobias Eidelpes, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 20. Februar 2023

Tobias Eidelpes

Danksagung

Ihr Text hier.

Acknowledgements

Enter your text here.

Kurzfassung

Ihr Text hier.

Abstract

Enter your text here.

Contents

Kurzfassung **xi**

Abstract **xiii**

Contents **xv**

1 Evaluation **1**
1.1 Object Detection 1
1.2 Classification 7
1.3 Aggregate Model 14

List of Figures **19**

List of Tables **21**

List of Algorithms **23**

Acronyms **25**

Bibliography **27**

CHAPTER 1

Evaluation

The following sections contain a detailed evaluation of the model in various scenarios. First, we present metrics from the training phases of the constituent models. Second, we employ methods from the field of Explainable Artificial Intelligence (XAI) such as Gradient-weighted Class Activation Mapping (Grad-CAM) to get a better understanding of the models' abstractions. Finally, we turn to the models' aggregate performance on the test set and discuss whether the initial goals set by the problem description have been met or not.

1.1 Object Detection

The object detection model was pre-trained on the COCO [LMB+15] dataset and fine-tuned with data from the Open Images Dataset (OID) [KRA+20] in its sixth version. Since the full OID dataset contains considerably more classes and samples than would be feasibly trainable on a small cluster of GPUs, only images from the two classes *Plant* and *Houseplant* have been downloaded. The samples from the Houseplant class are merged into the Plant class because the distinction between the two is not necessary for our model. Furthermore, the OID contains not only bounding box annotations for object detection tasks, but also instance segmentations, classification labels and more. These are not needed for our purposes and are omitted as well. In total, the dataset consists of 91479 images with a roughly 85/5/10 split for training, validation and testing, respectively.

1.1.1 Training Phase

The object detection model was trained for 300 epochs on 79204 images with 284130 ground truth labels. The weights from the best-performing epoch were saved. The model's fitness for each epoch is calculated as the weighted average of mAP@0.5 and mAP@0.5:0.95:

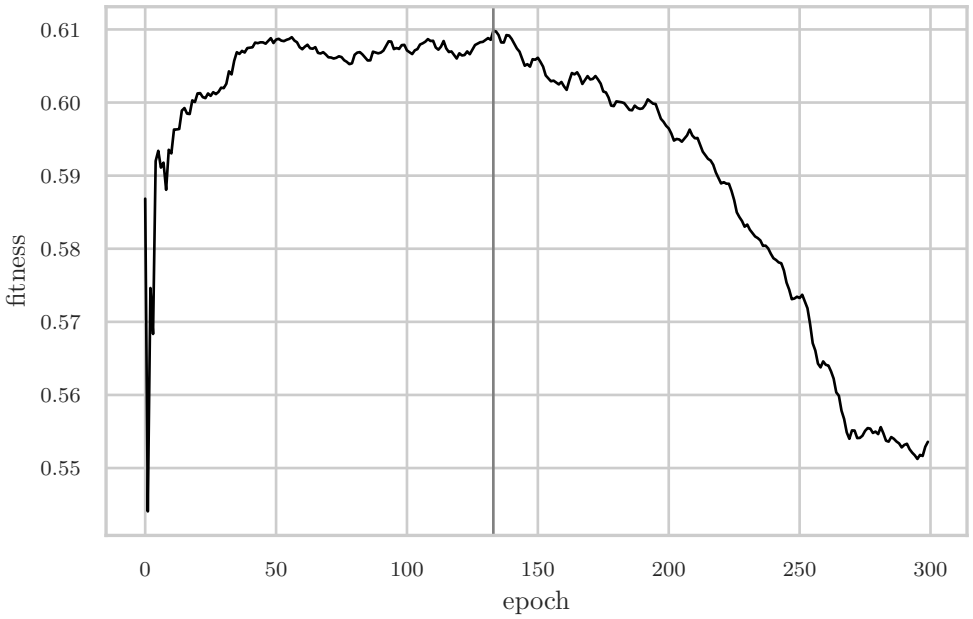

Figure 1.1: Object detection model fitness for each epoch calculated as in equation 1.1. The vertical gray line at 133 marks the epoch with the highest fitness.

$$f_{epoch} = 0.1 \cdot \mathsf{mAP@0.5} + 0.9 \cdot \mathsf{mAP@0.5:0.95} \tag{1.1}$$

Figure 1.1 shows the model's fitness over the training period of 300 epochs. The gray vertical line indicates the maximum fitness of 0.61 at epoch 133. The weights of that epoch were frozen to be the final model parameters. Since the fitness metric assigns the mAP at the higher range the overwhelming weight, the mAP@0.5 starts to decrease after epoch 30, but the mAP@0.5:0.95 picks up the slack until the maximum fitness at epoch 133. This is an indication that the model achieves good performance early on and continues to gain higher confidence values until performance deteriorates due to overfitting.

Overall precision and recall per epoch are shown in figure 1.2. The values indicate that neither precision nor recall change materially during training. In fact, precision starts to decrease from the beginning, while recall experiences a barely noticeable increase. Taken together with the box and object loss from figure 1.3, we speculate that the pre-trained model already generalizes well to plant detection because one of the categories in the COCO [LMB+15] dataset is *potted plant*. Any further training solely impacts the confidence of detection, but does not lead to higher detection rates. This conclusion is supported by the increasing mAP@0.5:0.95 until epoch 133.

Further culprits for the flat precision and recall values may be found in bad ground truth data. The labels from the OID are sometimes not fine-grained enough. Images

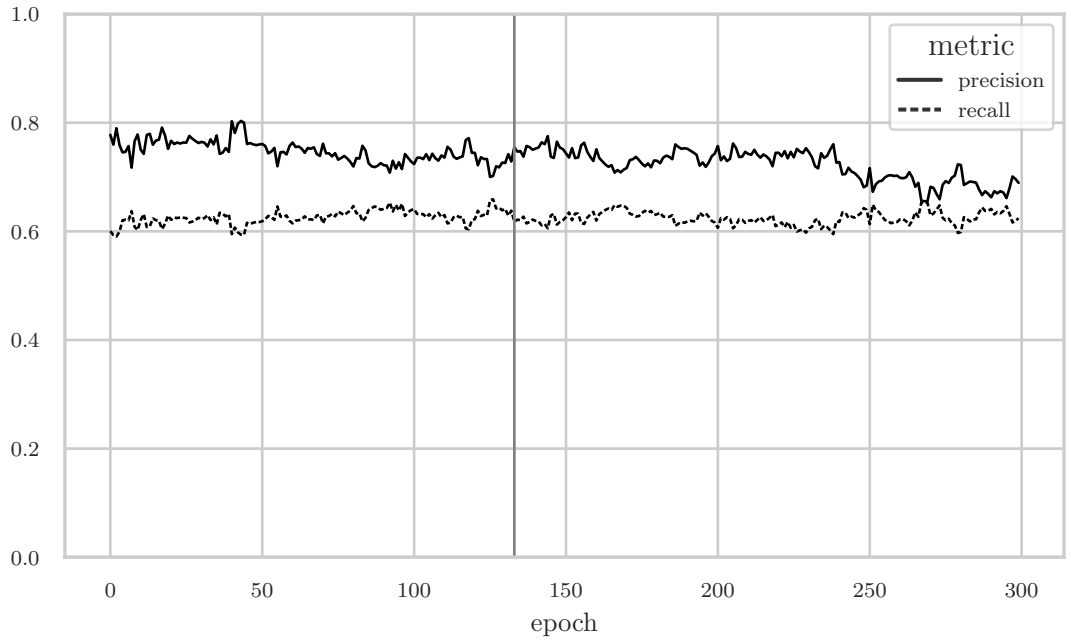

Figure 1.2: Overall precision and recall during training for each epoch. The vertical gray line at 133 marks the epoch with the highest fitness.

which contain multiple individual—often overlapping—plants are labeled with one large bounding box instead of multiple smaller ones. The model recognizes the individual plants and returns tighter bounding boxes even if that is not what is specified in the ground truth. Therefore, it is prudent to limit the training phase to relatively few epochs in order to not penalize the more accurate detections of the model. The smaller bounding boxes make more sense considering the fact that the cutout is passed to the classifier in a later stage. Smaller bounding boxes help the classifier to only focus on one plant at a time and to not get distracted by multiple plants in potentially different stages of wilting.

The box loss decreases slightly during training which indicates that the bounding boxes become tighter around objects of interest. With increasing training time, however, the object loss increases, indicating that less and less plants are present in the predicted bounding boxes. It is likely that overfitting is a cause for the increasing object loss from epoch 40 onward. Since the best weights as measured by fitness are found at epoch 133 and the object loss accelerates from that point, epoch 133 is probably the correct cutoff before overfitting occurs.

1.1.2 Test Phase

Of the 91479 images around 10% were used for the test phase. These images contain a total of 12238 ground truth labels. Table 1.1 shows precision, recall and the harmonic mean of both (F1-score). The results indicate that the model errs on the side of sensitivity

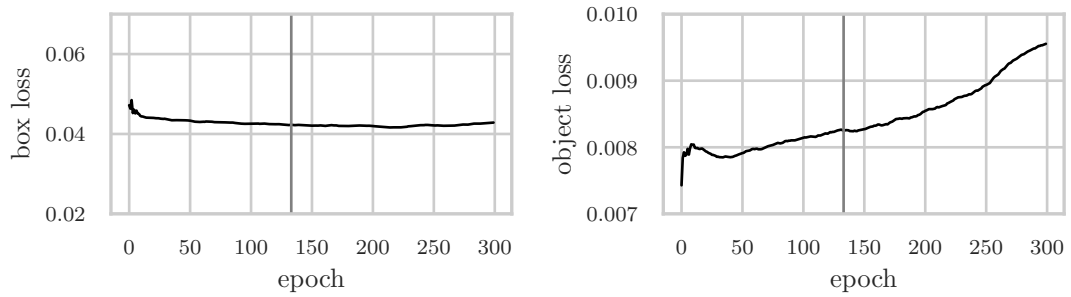

Figure 1.3: Box and object loss measured against the validation set of 3091 images and 4092 ground truth labels. The class loss is omitted because there is only one class in the dataset and the loss is therefore always zero.

because recall is higher than precision. Although some detections are not labeled as plants in the dataset, if there is a labeled plant in the ground truth data, the chance is high that it will be detected. This behavior is in line with how the model's detections are handled in practice. The detections are drawn on the original image and the user is able to check the bounding boxes visually. If there are wrong detections, the user can ignore them and focus on the relevant ones instead. A higher recall will thus serve the user's needs better than a high precision.

	Precision	Recall	F1-score	Support
Plant	0.547571	0.737866	0.628633	12238.0

Table 1.1: Precision, recall and F1-score for the object detection model.

Figure 1.4 shows the Average Precision (AP) for the Intersection over Union (IOU) thresholds of 0.5 and 0.95. Predicted bounding boxes with an IOU of less than 0.5 are not taken into account for the precision and recall values of table 1.1. The lower the detection threshold, the more plants are detected. Conversely, a higher detection threshold leaves potential plants undetected. The precision-recall curves confirm this behavior because the area under the curve for the threshold of 0.5 is higher than for the threshold of 0.95 (0.66 versus 0.41). These values are combined in COCO's [LMB+15] main evaluation metric which is the AP averaged across the IOU thresholds from 0.5 to 0.95 in 0.05 steps. This value is then averaged across all classes and called mean average precision (mAP). The object detection model achieves a state-of-the-art mAP of 0.5727 for the *Plant* class.

1.1.3 Hyper-parameter Optimization

To further improve the object detection performance, we perform hyper-parameter optimization using a genetic algorithm. Evolution of the hyper-parameters starts from the initial 30 default values provided by the authors of YOLO. Of those 30 values, 26 are allowed to mutate. During each generation, there is an 80% chance that a mutation

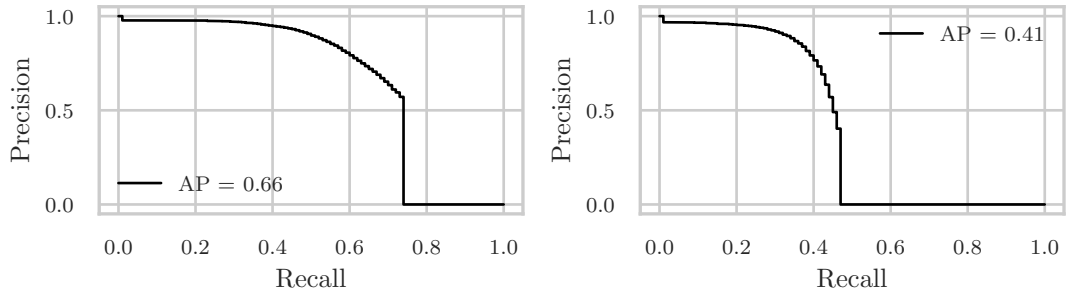

Figure 1.4: Precision-recall curves for IOU thresholds of 0.5 and 0.95. The AP of a specific threshold is defined as the area under the precision-recall curve of that threshold. The mAP across IOU thresholds from 0.5 to 0.95 in 0.05 steps mAP@0.5:0.95 is 0.5727.

occurs with a variance of 0.04. To determine which generation should be the parent of the new mutation, all previous generations are ordered by fitness in decreasing order. At most five top generations are selected and one of them is chosen at random. Better generations have a higher chance of being selected as the selection is weighted by fitness. The parameters of that chosen generation are then mutated with the aforementioned probability and variance. Each generation is trained for three epochs and the fitness of the best epoch is recorded.

In total, we ran 87 iterations of which the 34^{th} generation provides the best fitness of 0.6076. Due to time constraints, it was not possible to train each generation for more epochs or to run more iterations in total. We assume that the performance of the first few epochs is a reasonable proxy for model performance overall. The optimized version of the object detection model is then trained for 70 epochs using the parameters of the 34^{th} generation.

Figure 1.5 shows the model's fitness during training for each epoch. After the highest fitness of 0.6172 at epoch 27, the performance quickly declines and shows that further training would likely not yield improved results. The model converges to its highest fitness much earlier than the non-optimized version discussed in section 1.1.1, which indicates that the adjusted parameters provide a better starting point in general. Furthermore, the maximum fitness is 0.74% higher than in the non-optimized version.

Figure 1.6 shows precision and recall for the optimized model during training. Similarly to the non-optimized model from figure 1.2, both metrics do not change materially during training. Precision is slightly higher than in the non-optimized version and recall hovers at the same levels.

The box and object loss during training is pictured in figure 1.7. Both losses start from a lower level which suggests that the initial optimized parameters allow the model to converge quicker. The object loss exhibits a similar slope to the non-optimized model in figure 1.3. The vertical gray line again marks epoch 27 with the highest fitness. The box

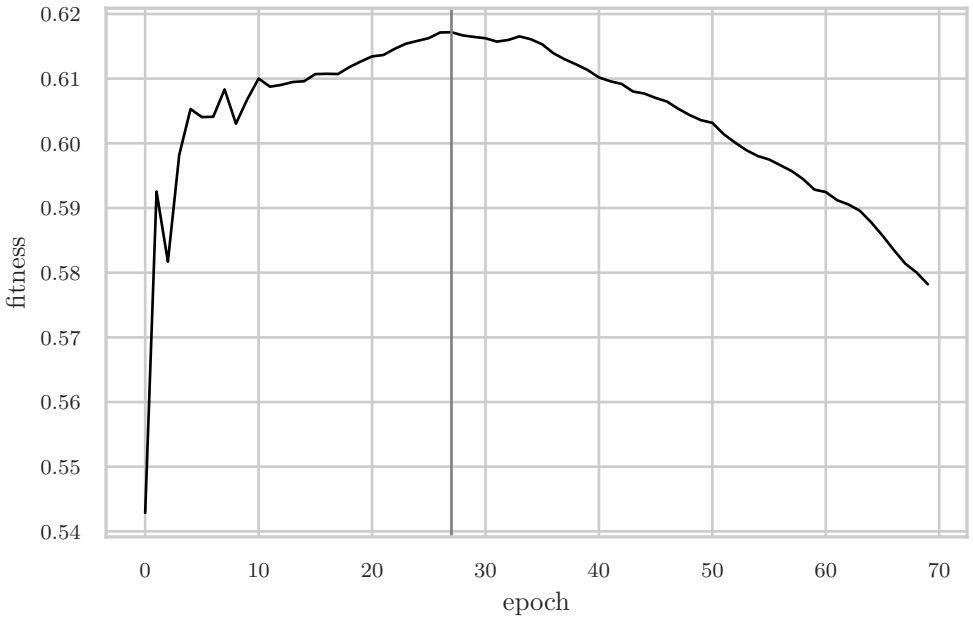

Figure 1.5: Object detection model fitness for each epoch calculated as in equation 1.1. The vertical gray line at 27 marks the epoch with the highest fitness of 0.6172.

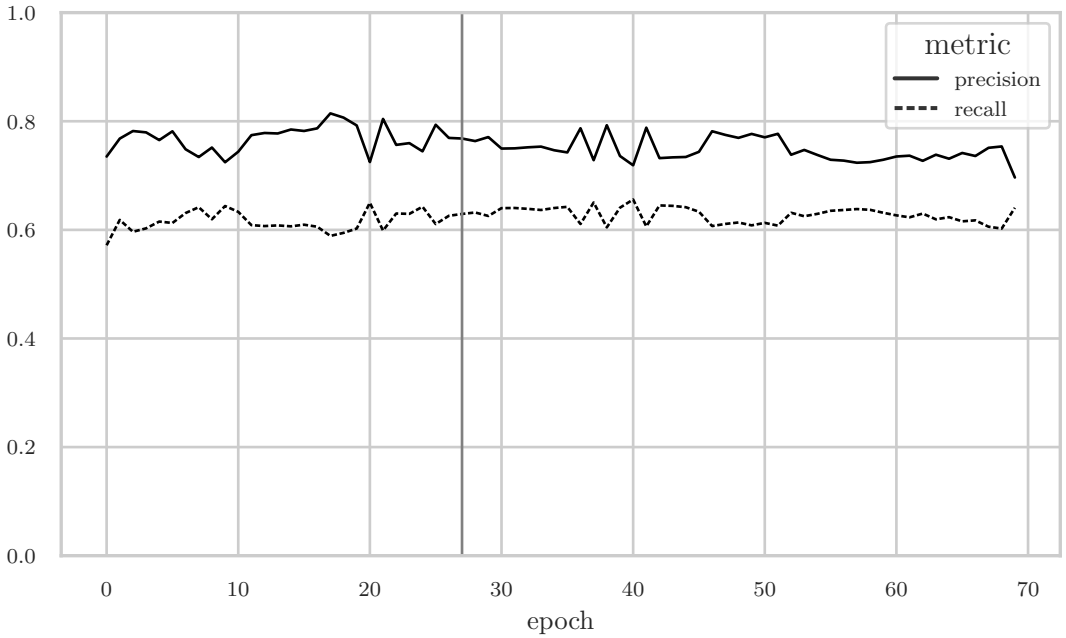

Figure 1.6: Overall precision and recall during training for each epoch of the optimized model. The vertical gray line at 27 marks the epoch with the highest fitness.

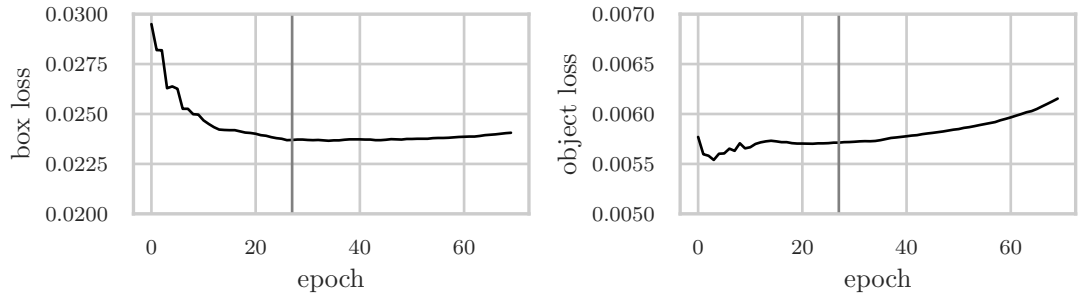

Figure 1.7: Box and object loss measured against the validation set of 3091 images and 4092 ground truth labels. The class loss is omitted because there is only one class in the dataset and the loss is therefore always zero.

loss reaches its lower limit at that point and the object loss starts to increase again after epoch 27.

	Precision	Recall	F1-score	Support
Plant	0.633358	0.702811	0.666279	12238.0

Table 1.2: Precision, recall and F1-score for the optimized object detection model.

Turning to the evaluation of the optimized model on the test dataset, table 1.2 shows precision, recall and the F1-score for the optimized model. Comparing these metrics with the non-optimized version from table 1.1, precision is significantly higher by more than 8.5%. Recall, however, is 3.5% lower. The F1-score is higher by more than 3.7% which indicates that the optimized model is better overall despite the lower recall. We feel that the lower recall value is a suitable trade off for the substantially higher precision considering that the non-optimized model's precision is quite low at 0.55.

The precision-recall curves in figure 1.8 for the optimized model show that the model draws looser bounding boxes than the optimized model. The AP for both IOU thresholds of 0.5 and 0.95 is lower indicating worse performance. It is likely that more iterations during evolution would help increase the AP values as well. Even though the precision and recall values from table 1.2 are better, the mAP@0.5:0.95 is lower by 1.8%.

1.2 Classification

The classifier receives cutouts from the object detection model and determines whether the image shows a stressed plant or not. To achieve this goal, we trained a Residual Neural Network (ResNet) [HZR+16] on a dataset of 452 images of healthy and 452 stressed plants. We chose the ResNet architecture due to its popularity and ease of implementation as well as its consistently high performance on various classification tasks. While its classification speed in comparison with networks optimized for mobile and edge

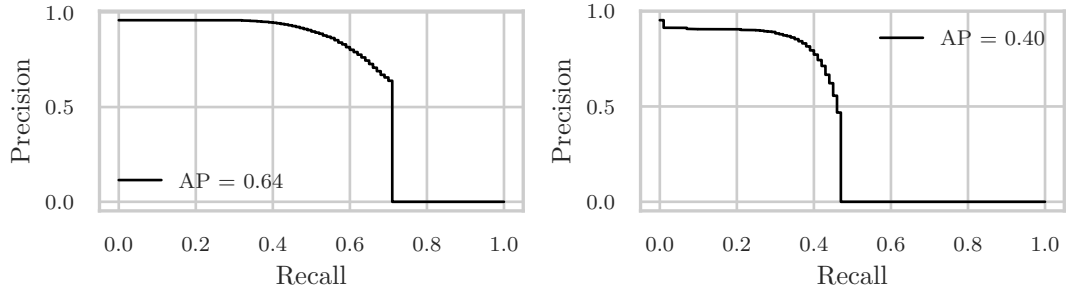

Figure 1.8: Precision-recall curves for IOU thresholds of 0.5 and 0.95. The AP of a specific threshold is defined as the area under the precision-recall curve of that threshold. The mAP across IOU thresholds from 0.5 to 0.95 in 0.05 steps mAP@0.5:0.95 is 0.5546.

devices (e.g. MobileNet) is significantly lower, the deeper structure and the additional parameters are necessary for the fairly complex task at hand. Furthermore, the generous time budget for object detection *and* classification allows for more accurate results at the expense of speed. The architecture allows for multiple different structures, depending on the amount of layers. The smallest one has 18 and the largest 152 layers with 34, 50 and 101 in-between. The larger networks have better accuracy in general, but come with trade-offs regarding training and inference time as well as required space. The 50 layer architecture (ResNet50) is adequate for our use case.

1.2.1 Training Phase

The dataset was split 85/15 into training and validation sets. The images in the training set were augmented with a random crop to arrive at the expected image dimensions of 224 pixels. Additionally, the training images were modified with a random horizontal flip to increase the variation in the set and to train a rotation invariant classifier. All images, regardless of their membership in the training or validation set, were normalized with the mean and standard deviation of the ImageNet [DDS+09] dataset, which the original ResNet model was pre-trained with. Training was done for 50 epochs and the best-performing model as measured by validation accuracy was selected as the final version.

Figure 1.9 shows accuracy and loss on the training and validation sets. There is a clear upwards trend until epoch 20 when validation accuracy and loss stabilize at around 0.84 and 0.3, respectively. The quick convergence and resistance to overfitting can be attributed to the model already having robust feature extraction capabilities.

1.2.2 Hyper-parameter Optimization

In order to improve the aforementioned accuracy values, we perform hyper-parameter optimization across a wide range of parameters. Table 1.3 lists the hyper-parameters and their possible values. Since the number of all combinations of values is 11520 and

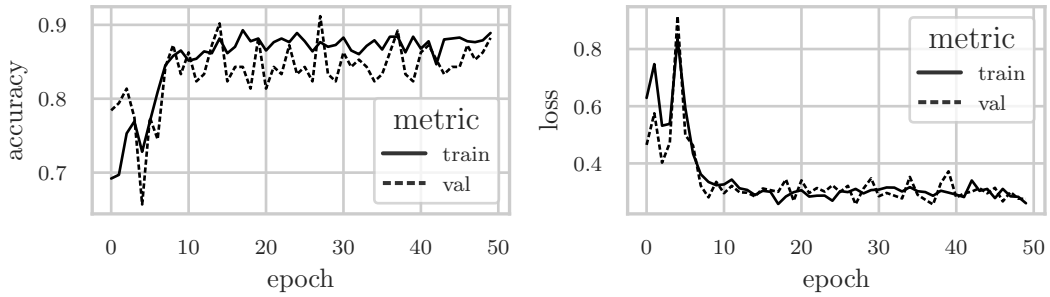

Figure 1.9: Accuracy and loss during training of the classifier. The model converges quickly, but additional epochs do not cause validation loss to increase, which would indicate overfitting. The maximum validation accuracy of 0.9118 is achieved at epoch 27.

each combination is trained for 10 epochs with a training time of approximately six minutes per combination, exhausting the search space would take 48 days. Due to time limitations, we have chosen to not search exhaustively but to pick random combinations instead. Random search works surprisingly well—especially compared to grid search—in a number of domains, one of which is hyper-parameter optimization [BB12].

Parameter	Values
optimizer	adam, sgd
batch size	4, 8, 16, 32, 64
learning rate	0.0001, 0.0003, 0.001, 0.003, 0.01, 0.1
step size	2, 3, 5, 7
gamma	0.1, 0.5
beta one	0.9, 0.99
beta two	0.5, 0.9, 0.99, 0.999
eps	0.00000001, 0.1, 1

Table 1.3: Hyper-parameters and their possible values during optimization.

The random search was run for 138 iterations which equates to a 75% probability that the best solution lies within 1% of the theoretical maximum (1.2). Figure 1.10 shows three of the eight parameters and their impact on a high F1-score. Stochastic Gradient Descent (SGD) has less variation in its results than Adam [KB17] and manages to provide eight out of the ten best results. The number of epochs to train for was chosen based on the observation that almost all configurations converge well before reaching the tenth epoch. The assumption that a training run with ten epochs provides a good proxy for final performance is supported by the quick convergence of validation accuracy and loss in figure 1.9.

$$1 - (1 - 0.01)^{138} \approx 0.75 \tag{1.2}$$

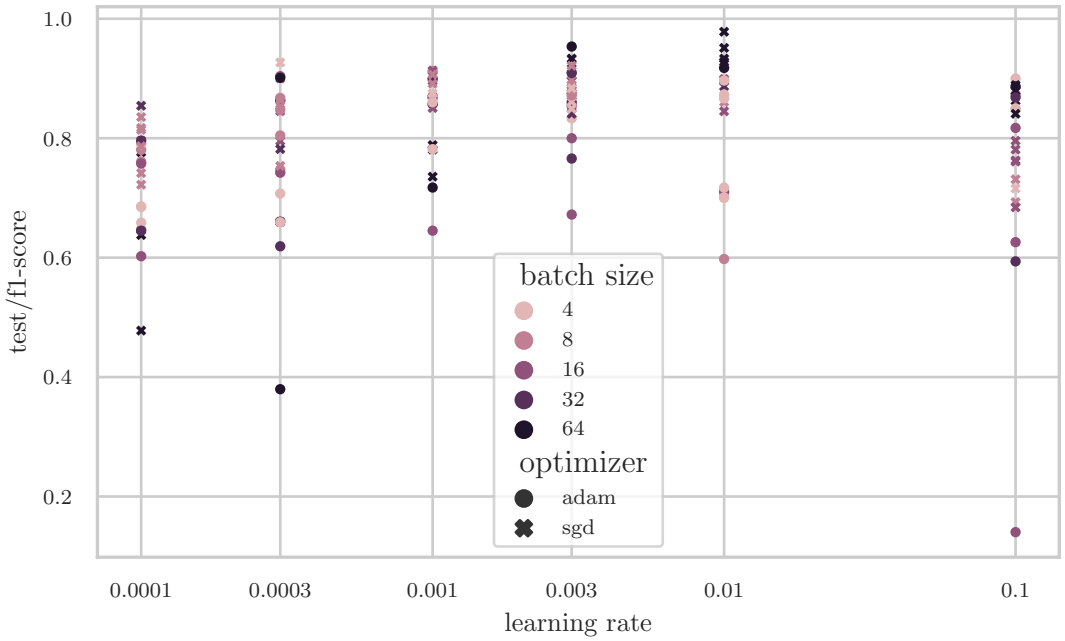

Figure 1.10: This figure shows three of the eight hyper-parameters and their performance measured by the F1-score during 138 trials. Differently colored markers show the batch size with darker colors representing a larger batch size. The type of marker (circle or cross) shows which optimizer was used. The x-axis shows the learning rate on a logarithmic scale. In general, a learning rate between 0.003 and 0.01 results in more robust and better F1-scores. Larger batch sizes more often lead to better performance as well. As for the type of optimizer, SGD produced the best iteration with an F1-score of 0.9783. Adam tends to require more customization of its parameters than SGD to achieve good results.

Table 1.4 lists the final hyper-parameters which were chosen to train the improved model. In order to confirm that the model does not suffer from overfitting or is a product of chance due to a coincidentally advantageous train/test split, we perform stratified 10-fold cross validation on the dataset. Each fold contains 90% training and 10% test data and was trained for 25 epochs. Figure 1.11 shows the performance of the epoch with the highest F1-score of each fold as measured against the test split. The mean Receiver Operating Characteristic (ROC) curve provides a robust metric for a classifier's performance because it averages out the variability of the evaluation. Each fold manages to achieve at least an Area Under the Curve (AUC) of 0.94, while the best fold reaches 0.98. The mean ROC has an AUC of 0.96 with a standard deviation of 0.02. These results indicate that the model is accurately predicting the correct class and is robust against variations in the training set.

The classifier shows good performance so far, but care has to be taken to not overfit the model to the training set. Comparing the F1-score during training with the F1-score

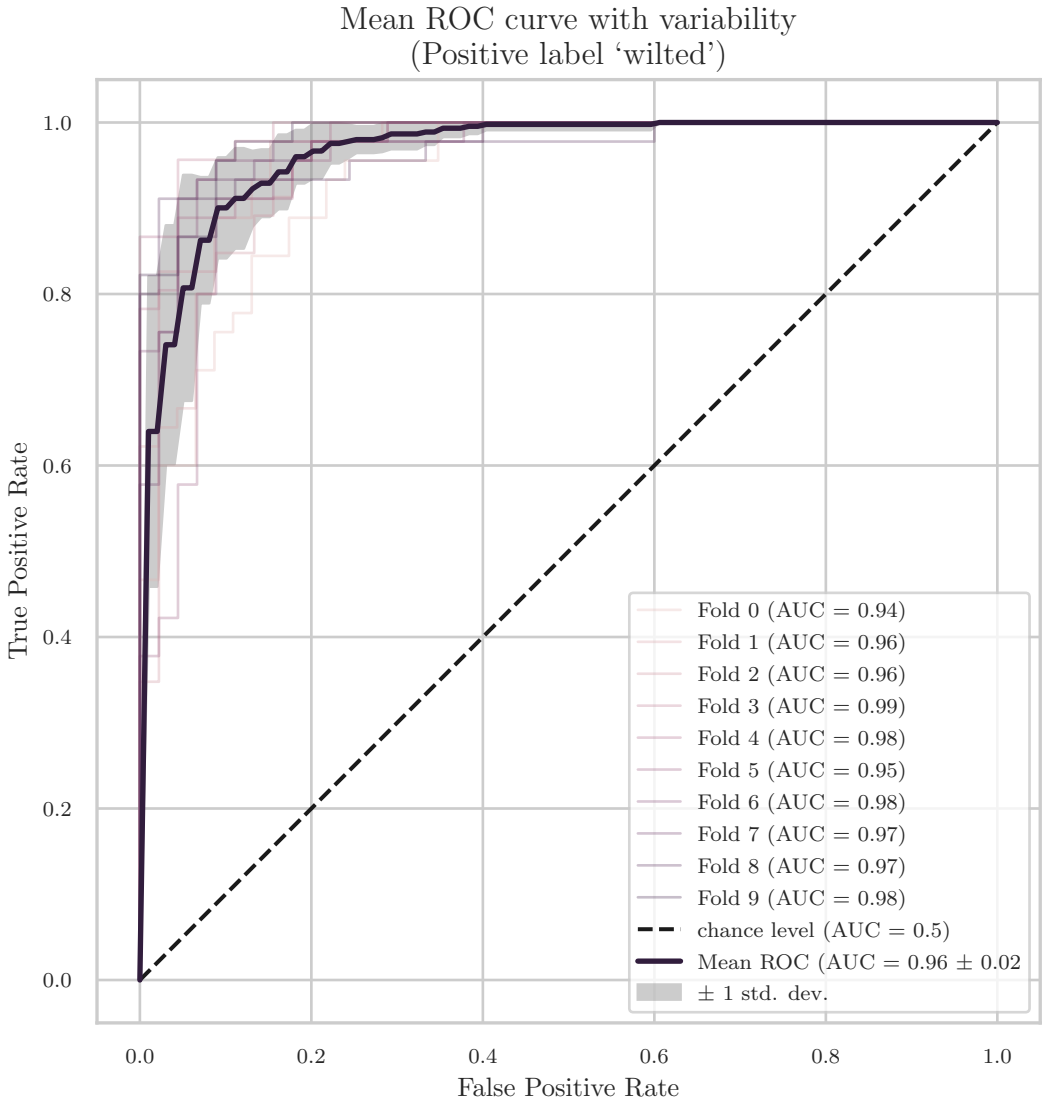

Figure 1.11: This plot shows the ROC curve for the epoch with the highest F1-score of each fold as well as the AUC. To get a less variable performance metric of the classifier, the mean ROC curve is shown as a thick line and the variability is shown in gray. The overall mean AUC is 0.96 with a standard deviation of 0.02. The best-performing fold reaches an AUC of 0.99 and the worst an AUC of 0.94. The black dashed line indicates the performance of a classifier which picks classes at random (AUC = 0.5). The shapes of the ROC curves show that the classifier performs well and is robust against variations in the training set.

Optimizer	Batch Size	Learning Rate	Step Size
SGD	64	0.01	5

Table 1.4: Chosen hyper-parameters for the final, improved model. The difference to the parameters listed in Table 1.3 comes as a result of choosing SGD over Adam. The missing four parameters are only required for Adam and not SGD.

during testing gives insight into when the model tries to increase its performance during training at the expense of generalizability. Figure 1.12 shows the F1-scores of each epoch and fold. The classifier converges quickly to 1 for the training set at which point it experiences a slight drop in generalizability. Training the model for at most five epochs is sufficient because there are generally no improvements afterwards. The best-performing epoch for each fold is between the second and fourth epoch which is just before the model achieves an F1-score of 1 on the training set.

1.2.3 Class Activation Maps

Neural networks are notorious for their black-box behavior, where it is possible to observe the inputs and the corresponding outputs, but the stage in-between stays hidden from view. Models are continuously developed and deployed to aid in human decision-making and sometimes supplant it. It is, therefore, crucial to obtain some amount of interpretability of what the model does *inside* to be able to explain why a decision was made in a certain way. The research field of XAI gained significance during the last few years because of the development of new methods to peek inside these black boxes.

One such method, Class Activation Mapping (CAM) [ZKL+15], is a popular tool to produce visual explanations for decisions made by Convolutional Neural Networks (CNNs). Convolutional layers essentially function as object detectors as long as no fully-connected layers perform the classification. This ability to localize regions of interest, which play a significant role in the type of class the model predicts, can be retained until the last layer and used to generate activation maps for the predictions.

A more recent approach to generating a CAM via gradients is proposed by Selvaraju et al. [SCD+20]. Their Grad-CAM approach works by computing the gradient of the feature maps of the last convolutional layer with respect to the specified class. The last layer is chosen because the authors find that "[...] Grad-CAM maps become progressively worse as we move to earlier convolutional layers as they have smaller receptive fields and only focus on less semantic local features." [SCD+20, p.5]

Turning to our classifier, figure 1.13 shows the CAMs for *healthy* and *stressed*. While the regions of interest for the *healthy* class lie on the healthy plant, the *stressed* plant is barely considered and mostly rendered as background information (blue). Conversely, when asked to explain the inputs to the *stressed* classification, the regions of interest predominantly stay on the thirsty as opposed to the healthy plant. In fact, the large

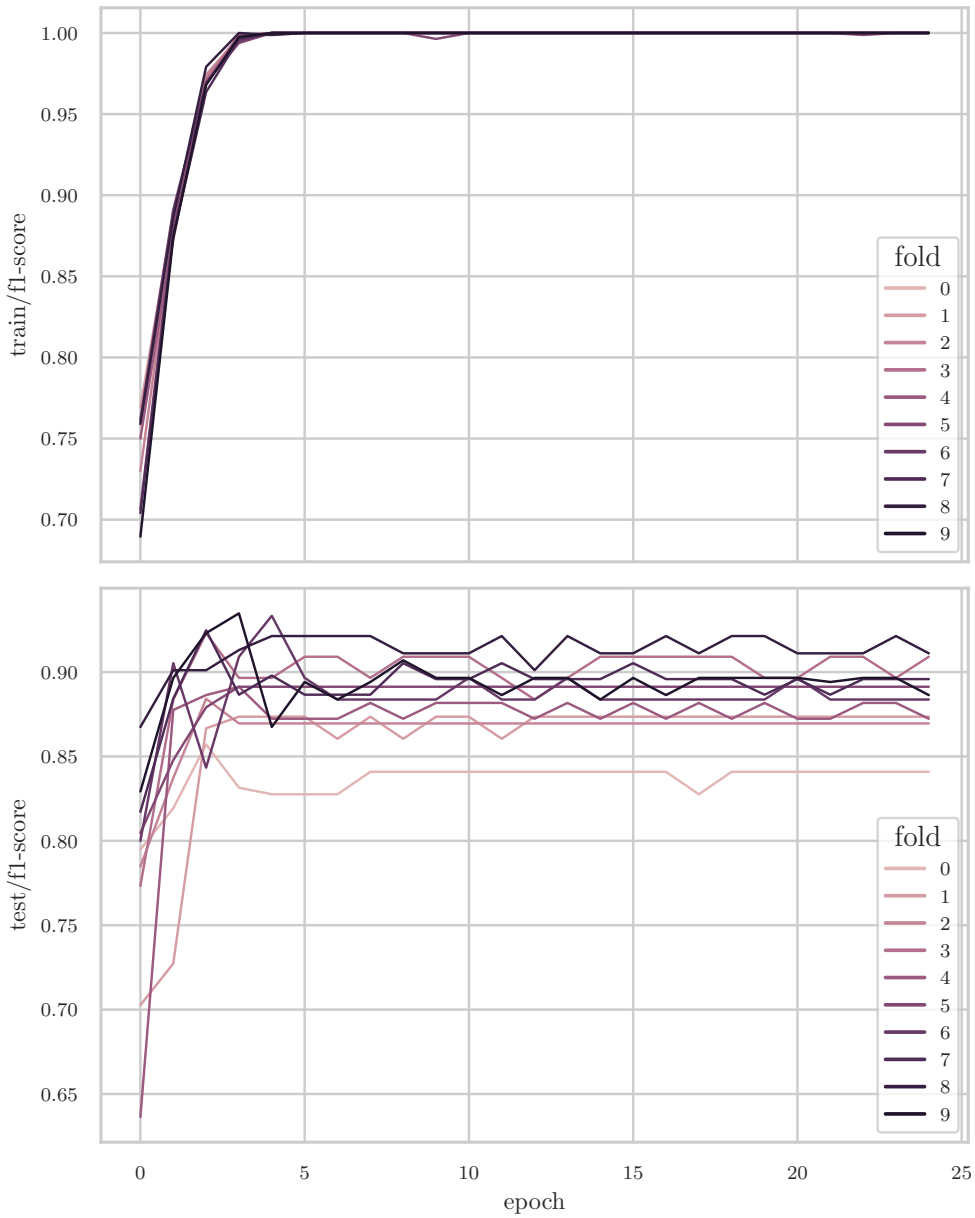

Figure 1.12: These plots show the F1-score during training as well as testing for each of the folds. The classifier converges to 1 by the third epoch during the training phase, which might indicate overfitting. However, the performance during testing increases until epoch three in most cases and then stabilizes at approximately 2-3% lower than the best epoch. We believe that the third, or in some cases fourth, epoch is detrimental to performance and results in overfitting, because the model achieves an F1-score of 1 for the training set, but that gain does not transfer to the test set. Early stopping during training alleviates this problem.

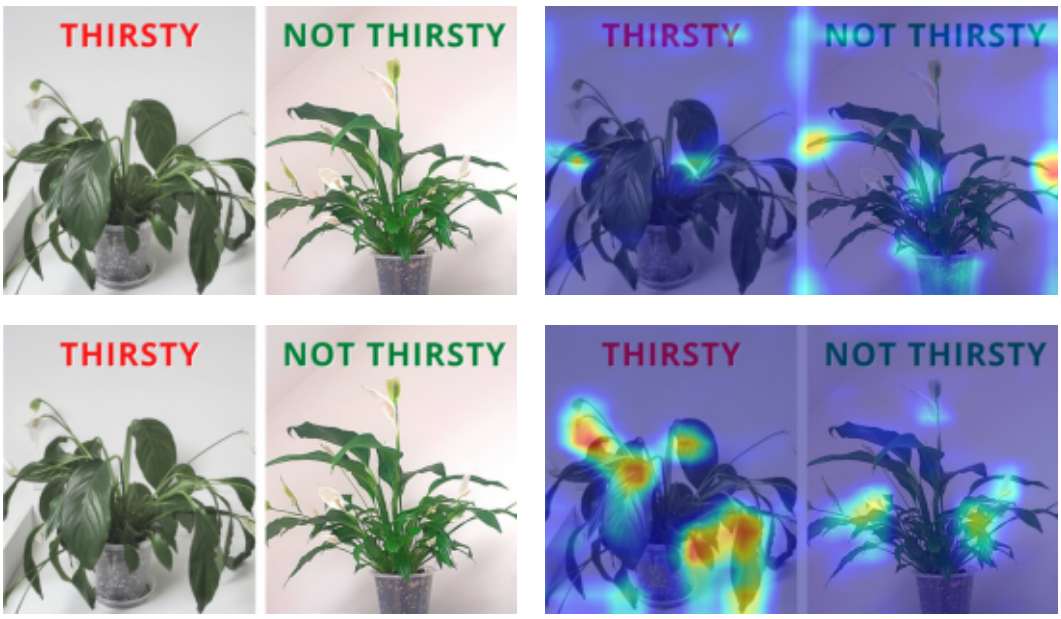

Figure 1.13: The top left image shows the original image of the same plant in a stressed (left) and healthy (right) state. In the top right image, the CAM for the class *healthy* is laid over the original image. The classifier draws its conclusion mainly from the healthy plant, which is indicated by the red hot spots around the tips of the plant. The bottom right image shows the CAM for the *stressed* class. The classifier focuses on the hanging leaves of the thirsty plant. The image was classified as *stressed* with a confidence of 70%.

hanging leaves play a significant role in determining the class the image belongs to. This is an additional data point confirming that the model focuses on the *right* parts of the image during classification.

1.3 Aggregate Model

In this section we turn to the evaluation of the aggregate model. We have confirmed the performance of the constituent models: the object detection and the classification model. It remains to evaluate the complete pipeline from gathering detections of potential plants in an image and forwarding them to the classifier to obtaining the results as either healthy or stressed with their associated confidence scores.

The test set contains 640 images which were obtained from a google search using the terms *thirsty plant*, *wilted plant* and *stressed plant*. Images which clearly show one or multiple plants with some amount of visible stress were added to the dataset. Care was taken to include plants with various degrees of stress and in various locations and lighting conditions. The search not only provided images of stressed plants, but also of healthy plants due to articles, which describe how to care for plants, having a banner image of

	Precision	Recall	F1-score	Support
Healthy	0.824	0.745	0.783	662.0
Stressed	0.707	0.783	0.743	488.0
micro avg	0.769	0.761	0.765	1150.0
macro avg	0.766	0.764	0.763	1150.0
weighted avg	0.775	0.761	0.766	1150.0

Table 1.5: Precision, recall and F1-score for the aggregate model.

healthy plants. The dataset is biased towards potted plants which are commonly put on display in western households. Furthermore, many plants, such as succulents, are sought after for home environments because of their ease of maintenance. Due to their inclusion in the dataset and how they exhibit water stress, the test set nevertheless contains a wide variety of scenarios.

After collecting the images, the aggregate model was run on them to obtain initial bounding boxes and classifications for ground truth labeling. Letting the model do the work beforehand and then correcting the labels allowed to include more images in the test set because they could be labeled more easily. Additionally, going over the detections and classifications provided a comprehensive view on how the models work and what their weaknesses and strengths are. After the labels have been corrected, the ground truth of the test set contains 662 bounding boxes of healthy plants and 488 of stressed plants.

Table 1.5 shows precision, recall and the F1-score for both classes *Healthy* and *Stressed*. Both precision and recall are balanced and the F1-score is high. Unfortunately, these values do not take the accuracy of bounding boxes into account and thus have only limited expressive power.

Figure 1.14 shows the precision and recall curves for both classes at different IOU thresholds. The left plot shows the AP for each class at the threshold of 0.5 and the right one at 0.95. The mAP is 0.6226 and calculated across all classes as the median of the IOU thresholds from 0.5 to 0.95 in 0.05 steps. The difference between mAP@0.5 and mAP@0.95 is fairly small which indicates that the bounding boxes encapsulate the objects of interest well. The cliffs at around 0.77 (left) and 0.7 (right) happen at a detection threshold of 0.5. The classifier's last layer is a softmax layer which necessarily transforms the input into a probability of showing either a healthy or stressed plant. If the probability of an image showing a healthy plant is below 0.5, it is no longer classified as healthy but as stressed. The threshold for discriminating the two classes lies at the 0.5 value and is therefore the cutoff for either class.

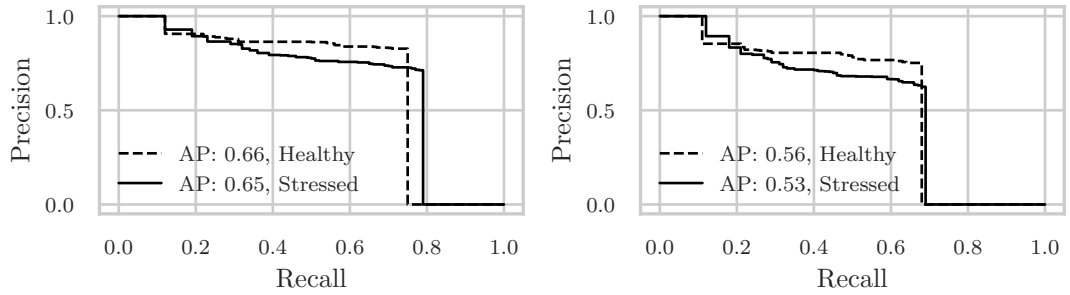

Figure 1.14: Precision-recall curves for IOU thresholds of 0.5 and 0.95. The AP of a specific threshold is defined as the area under the precision-recall curve of that threshold. The mAP across IOU thresholds from 0.5 to 0.95 in 0.05 steps mAP@0.5:0.95 is 0.6226.

	precision	recall	f1-score	support
Healthy	0.664	0.640	0.652	662.0
Stressed	0.680	0.539	0.601	488.0
micro avg	0.670	0.597	0.631	1150.0
macro avg	0.672	0.590	0.626	1150.0
weighted avg	0.670	0.597	0.630	1150.0

Table 1.6: Precision, recall and F1-score for the optimized aggregate model.

1.3.1 Hyper-parameter Optimization

So far the metrics shown in table 1.5 are obtained with the non-optimized versions of both the object detection and classification model. Hyper-parameter optimization of the classifier led to significant model improvements, while the object detector has improved precision but lower recall and slightly lower mAP values. To evaluate the final aggregate model which consists of the individual optimized models, we run the same test as in section 1.3.

Table 1.6 shows precision, recall and F1-score for the optimized model on the same test dataset of 640 images. All of the metrics are significantly worse than for the non-optimized model. Considering that the optimized classifier performs better than the non-optimized version this is a surprising result. There are multiple possible explanations for this behavior:

1. The optimized classifier has worse generalizability than the non-optimized version.

2. The small difference in the mAP values for the object detection model result in significantly higher error rates overall. This might be the case because a large number of plants is not detected in the first place and/or those which are detected are more often not classified correctly by the classifier. As mentioned in section 1.1.3,

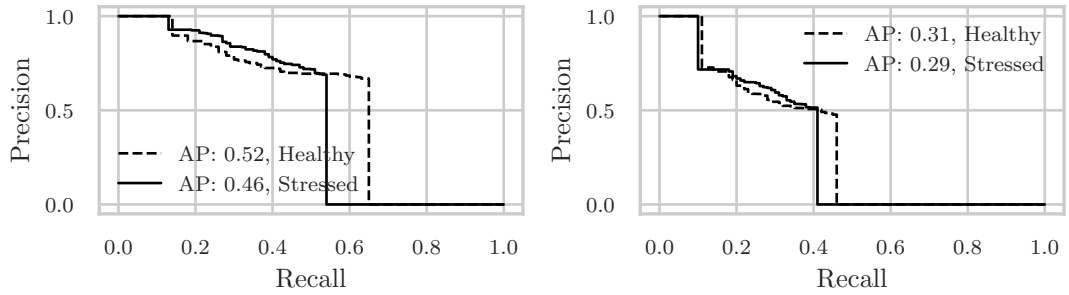

Figure 1.15: Precision-recall curves for IOU thresholds of 0.5 and 0.95. The AP of a specific threshold is defined as the area under the precision-recall curve of that threshold. The mAP across IOU thresholds from 0.5 to 0.95 in 0.05 steps mAP@0.5:0.95 is 0.4426.

running the evolution of the hyper-parameters for more generations could better the performance overall.

3. The test dataset is tailored to the non-optimized version and does not provide an accurate measure of real-world performance. The test dataset was labeled by running the individual models on the images and taking the predicted bounding boxes and labels as a starting point for the labeling process. If the labels were not rigorously corrected, the dataset will allow the non-optimized model to achieve high scores because the labels are already in line with what it predicts. Conversely, the optimized model might get closer to the actual ground truth, but that truth is not what is specified by the labels to begin with. If that is the case, the evaluation of the non-optimized model is too favorably and should be corrected down.

Of these three possibilities, the second and third points are the most likely culprits. The first scenario is unlikely because the optimized classifier has been evaluated in a cross validation setting and the results do not lend themselves easily to such an interpretation. Dealing with the second scenario could allow the object detection model to perform better on its own, but would probably not explain the big difference in performance. Scenario three is the most likely one because the process of creating the test dataset can lead to favorable labels for the non-optimized model.

Figure 1.15 confirms the suspicions raised by the lower metrics from table 1.6. More iterations for the evolution of the object detection model would likely have a significant effect on IOU and the confidence values associated with the bounding boxes.

List of Figures

1.1 Object detection fitness per epoch. 2
1.2 Object detection precision and recall during training. 3
1.3 Object detection box and object loss. 4
1.4 Object detection AP@0.5 and AP@0.95. 5
1.5 Optimized object detection fitness per epoch. 6
1.6 Hyper-parameter optimized object detection precision and recall during training. 6
1.7 Hyper-parameter optimized object detection box and object loss. 7
1.8 Hyper-parameter optimized object detection AP@0.5 and AP@0.95. . . . 8
1.9 Classifier accuracy and loss during training. 9
1.10 Classifier hyper-parameter optimization results. 10
1.11 Mean ROC and variability of hyper-parameter-optimized model. 11
1.12 F1-score of stratified 10-fold cross validation. 13
1.13 Classifier CAMs. 14
1.14 Aggregate model AP@0.5 and AP@0.95. 16
1.15 Optimized aggregate model AP@0.5 and AP@0.95. 17

List of Tables

1.1 Precision, recall and F1-score for the object detection model. 4
1.2 Precision, recall and F1-score for the optimized object detection model. . 7
1.3 Hyper-parameters and their possible values during optimization. 9
1.4 Hyper-parameters for the optimized classifier. 12
1.5 Precision, recall and F1-score for the aggregate model. 15
1.6 Precision, recall and F1-score for the optimized aggregate model. 16

List of Algorithms

Acronyms

AP Average Precision. 4, 5, 8

CAM Class Activation Mapping. 6, 9

CNN Convolutional Neural Network. 6

Grad-CAM Gradient-weighted Class Activation Mapping. 1, 6

IOU Intersection over Union. 4, 5, 8

mAP mean average precision. 4, 5, 8

OID Open Images Dataset. 1, 2

ResNet Residual Neural Network. 4, 5

XAI Explainable Artificial Intelligence. 1, 6

Bibliography

[BB12] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13:281–305, null, February 1, 2012.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 248–255, June 2009. DOI: 10.1109/CVPR.2009.5206848.

[HZR+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, June 2016. DOI: 10.1109/CVPR.2016.90.

[KB17] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. January 29, 2017. DOI: 10.48550/arXiv.1412.6980. preprint.

[KRA+20] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. *International Journal of Computer Vision*, 128(7):1956–1981, July 2020. DOI: 10.1007/s11263-020-01316-z.

[LMB+15] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. February 20, 2015. DOI: 10.48550/arXiv.1405.0312. preprint.

[SCD+20] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. *International Journal of Computer Vision*, 128(2):336–359, February 2020. DOI: 10.1007/s11263-019-01228-7.

[ZKL+15] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning Deep Features for Discriminative Localization. December 13, 2015. DOI: 10.48550/arXiv.1512.04150. preprint.