Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Test-Time Adaptation for Segmentation

Master's Thesis

Bc. Klára Janoušková

Study program: Open Informatics
Specialisation: Computer Vision and Image Processing
Supervisor: Prof. Ing. Jiří Matas, Ph.D.
Co-supervisor: Dr. Chaim Baskin

Prague, May 2023

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Janoušková Klára** Personal ID number: **474386**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Computer Vision and Image Processing**

II. Master’s thesis details

Master’s thesis title in English:

Test-Time Adaptation for Segmentation

Master’s thesis title in Czech:

Adaptace během testování pro úlohu segmentace

Guidelines:

1. Get familiar with the literature on test-time adaptation, in particular via self-supervision. [1, 2, 3]
2. Propose and implement a novel method to improve the robustness of test-time adaptation to distribution shift for segmentation tasks. One possibility is to learn a deep neural network surrogate for a segmentation loss (such as IoU or boundary loss [4])) during the training phase.
3. Find a suitable benchmark and evaluate the method(s)
4. Compare to existing methods.

Bibliography / sources:

[1] Gandelsman, Yossi, et al. "Test-Time Training with Masked Autoencoders." Advances in Neural Information Processing Systems.
[2] Sun, Yu, et al. "Test-time training with self-supervision for generalization under distribution shifts." International conference on machine learning. PMLR, 2020.
[3] Liu, Yuejiang, et al. "TTT++: When does self-supervised test-time training fail or thrive?." Advances in Neural Information Processing Systems 34 (2021): 21808-21820.
[4] Kervadec, Hoel, et al. "Boundary loss for highly unbalanced segmentation." International conference on medical imaging with deep learning. PMLR, 2019.

Name and workplace of master’s thesis supervisor:

prof. Ing. Jiří Matas, Ph.D. Visual Recognition Group, FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **09.02.2023** Deadline for master's thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

prof. Ing. Jiří Matas, Ph.D. Supervisor’s signature	prof. Ing. Tomáš Svoboda, Ph.D. Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

. _____
Date of assignment receipt

_____
Student’s signature

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, May 2023

.....
Bc. Klára Janoušková

Abstract

The thesis proposes novel single-image test-time adaptation methods for image segmentation based on a deep neural network. Test-time adaptation improves the robustness of the deep neural network to data shift using a single unlabelled image with no other data available. The methods are evaluated on two different deep segmentation networks, including a recent segmentation model trained on 1 billion segmentation masks, the SegmentAnything Model. The best approach reduces the segmentation error of the two models caused by synthetic corruptions by 15.47 % and 25.12 %.

Keywords: Test-time adaptation, generalization, self-supervised learning, image segmentation.

Abstrakt

Tato práce navrhuje nové metody pro adaptaci segmentačních modelů založených na hlubokých sítích z jednoho obrázku během testování. Adaptace v době testování zvyšuje robustnost hluboké neuronové sítě vůči změně distribuce dat pomocí jednoho neanotovaného obrázku bez přístupu k jiným datům. Metody jsou vyhodnoceny na dvou různých hlubokých segmentačních sítích, včetně nedávno publikovaného segmentačního modelu SegmentAnything trénovaného na 1 miliardě segmentačních masek. Nejlepší navrhovaný přístup snižuje segmentační chybu vyhodnocovaných modelů způsobenou syntetickými poruchami o 15.47 % a 25.12 %.

Klíčová slova: Adaptace během testování, generalizace, učení bez učitele, segmentace obrazu.

Acknowledgements

First of all, I would like to express gratitude to my supervisors. To prof. Ing. Jiří Matas, Ph.D. for his guidance not only through the thesis but throughout the past 5 years I have spent in the lab. To dr. Chaim Baskin for his guidance and feedback during my internship at the Technion and subsequently, on the thesis.

I greatly appreciate the many valuable discussions about the thesis topic with Yash Patel and Moshe Kimhi. I would also like to thank Jan Dočekal and Martin Čejka for their feedback on the thesis content.

Finally, I would like to thank Martin for his infinite supply of cute animals and my family for their support throughout my academic journey.

List of Tables

List of Figures

List of Acronyms

ADV Adversarial-Attack-Based Test-Time Training. viii, 28, 39, 40, 63, 64

BCE Binary Cross Entropy. 26

CASS Class-Agnostic Semantic Segmentation Model. viii, 23, 35, 36, 39, 49

dIoU Deep-Intersection-over-Union-Based Test-Time Training. viii, 28, 31, 33, 35, 38–43, 52, 58, 61, 64, 66

ENT Entropy-Minimization-Based Test-Time Adaptation. viii, 28, 29, 38–41, 48–50, 59, 64, 67, 68

FGSM Fast Gradient Sign Method. 28

IoU Intersection over Union. 5, 13, 26, 27, 31, 32, 35–39, 71

MAE Masked Autoencoders. ix, 4, 5, 12, 13, 23, 26, 27, 31, 35

PGD Projected Gradient Descent. 28

REC Reconstruction-Based Test-Time Training. viii, 28, 30, 37–48, 52, 53, 58, 60, 64–66

REF Mask-Refinement-Based Test-Time Training. viii, 28, 33, 35, 38–52, 58, 62, 64–67, 70

SAM the SegmentAnything Model. viii, ix, 2, 5, 13, 14, 18, 25, 32, 35, 48–51, 67

sIoU SAM-Intersection-over-Union-Based Test-Time Training. viii, 32, 48–50, 67, 69

TTA Test-Time Adaptation. viii, ix, 3–5, 8–10, 18, 19, 23, 27–29, 31, 33, 36–53, 64–67

TTT Test-Time Training. 3, 4, 8, 9, 23, 32, 51

ViT Vision Transformer. 13, 23, 24, 51

Contents

Chapter 1

Introduction

State-of-the-art solutions to computer vision tasks are currently dominated by methods based on deep neural networks. Fully-supervised learning works very well when both images (also referred to as features or covariates) and their corresponding labels are available in abundance. A standard way of fully-supervised learning of deep nets is to optimize the parameters of the network by minimizing a loss function between the predicted and target values on a large dataset also referred to as the training data.

But fully-annotated data is also scarce. Many approaches to learning with limited supervision such as semi-supervised, weakly-supervised or self-supervised learning have been proposed with impressive results, greatly narrowing the gap between learning with limited supervision and fully-supervised. Even when fully-supervised data is available, self-supervised pretraining or extending the training set with unlabelled data can improve performance.

When the test (source distribution samples not used for training) and deployment (target distribution) data samples are independent and identically distributed and the source and target distributions are the same, theoretical bounds on the true risk under that distribution based on test accuracy exist. But when the distribution and its relationship to the source distribution are unknown, there are no guarantees on the model performance on previously unseen data.

Imagine a camera in a car that drives through different areas that experiences a sudden change of weather. The lens may get soiled. Animals appear on the road. In real-world applications, the environment where the model is deployed often changes all the time. In fact, very few things do not change in time. The change of distribution between the training data and the data encountered at deployment is referred to as domain shift. An example of how domain shift may impact the performance of a state-of-the art segmentation model trained on a billion of images is shown in Figure 1.1.

Even a single image provides information about the distribution. In some applications,

Figure 1.1: The impact of domain shift on the the SegmentAnything Model (SAM) trained on a billion of images. The top row shows an image similar to those in the training set (left) and the high-quality predicted mask (right). The bottom row shows an image corrupted by rain most likely not seen during training (left) and a low-quality mask prediction (right).

the model can receive completely independent images, for example, in an image editing tool where users upload arbitrary pictures.

The idea of adapting a model to a single image can be observed across many fields. For example, UniTune [1] finetunes a large pretrained diffusion model to maintain fidelity to the input image while allowing for expressive manipulations. The performance of portrait NeRF on unseen images in [2] is improved by finetuning an MLP which models radiance field on a single image. In tracking, many methods such as D3S [3] initialize the model from the first frame. In other vision tasks such as image classification or segmentation, the optimization process stops at the end of the training procedure, discarding the information about distribution carried by test images.

There are multiple kinds of domain shifts, each of which requires a different treatment. The one focused on in most of the computer vision domain adaptation literature is the covariate shift. Covariate shift refers to the setting where the distribution of the input features changes but the labels remain the same. Examples of covariate shift are changes from sunny to rainy weather, mobile to DSLR camera or photographs to sketches.

Other kinds of domain shift are label shift and concept shift. Label shift can be seen

as the opposite of covariate shift - the distribution of the label marginal changes while the conditional remains unchanged. Finally, concept shift refers to the case when even the definition of labels may change, for instance, public opinion on what is fashionable differs both regionally and in time. In this thesis, it is assumed that the shift is only in the distribution of the covariates.

Even when limited to covariate shift, different works make different assumptions on data availability. In some cases, both source and target data are available while in others, only target data is available. The target distribution may be fixed or change in time. The data may be available at once, in small batches or an image at a time. The set-up when a model is being adapted to domain shift but no source data are available is referred to as Test-Time Adaptation (TTA) or Test-Time Training (TTT). The difference is that TTA does not make any assumptions about the training procedure of the model. In contrast, in TTT, the training process is altered for example by learning an auxiliary self-supervised loss alongside the main task. Unless the distinction is important, the methods will be referred to simply as TTA.

This work explores the possibilities of adapting a segmentation model at test-time to a single image under an arbitrary covariate shift. No assumption about previous or subsequent frames distribution is made.

Out of the many methods proposed for TTA for classification, only the test-time batch normalization or statistics adaptation [4], [5] and entropy minimization [6] have been evaluated and used as baselines for image segmentation. In [7], the performance of entropy minimization in a continual setup is explored. Segmentation-specific methods based on augmented view consistency to identify reliable predictions are proposed in [8], [9]. Even though most works include an evaluation on Cityscapes [10] (a dataset focused on street scenes) to ACDC [11] (street scenes in adverse weather conditions such as fog, nighttime, rain, and snow) domain shift, the results are not comparable due to different setups. All of these benchmarks are driving-specific. As a result, there are many unknowns in how the methods compare in different scenarios.

Close to this thesis, a line of work on TTT in medical imaging explores learning segmentation-specific self-supervised functions to optimize in the form of a deep neural network, as opposed to an arbitrary self-supervised task such as image reconstruction. These self-supervised tasks are based on predicting an improved mask by an autoencoder [12] or learning to discriminate between in-distribution and out-of-distribution masks [13] by exploiting the discriminator network of a Generative Adversarial Network (GAN). To the best of our knowledge, similar methods have not been explored on traditional image segmentation datasets.

Generally, the TTA literature does not compare to TTT methods with the argument

that these require a modification of the training process and methods that can be applied to any model are preferable. As a result, it is not known how these methods perform on image segmentation. Given the prevalence of self-supervised pretraining, how powerful it has been shown to be for dense-prediction tasks [14], [15] and the success of TTT with Masked Autoencoders (MAE) [16] on classification, continuing training on these tasks alongside the target task is often a straightforward extension and generally, the area of TTT for image segmentation is a direction worth exploring.

Considering the aforementioned limitations, in the thesis, we first propose to adopt a similar evaluation approach to TTA for segmentation as in classification [16]–[18], that is, an extension of common segmentation datasets with corruptions as in Imagenet-C [19]. Imagenet-C was created by applying a wide range of corruptions of different levels to the original images. The corruptions include various types of blur, noise and intensity transforms, as well as simulating weather conditions such as frost, snow or fog. This has the advantage that performance on different domain shifts and levels can be studied easily. Similar extensions can be easily applied to any of the existing datasets, independent of size, task or difficulty. Such a benchmark provides a diverse validation set for hyper-parameter search and gaining a better understanding of the methods in different scenarios.

Next, we propose a diverse set of methods including three Test-Time Training (TTT) methods for adaptation of image segmentation tasks. From TTA methods, the performance of the commonly used entropy minimization from [6] and the adversarial-transformation-based optimization of [20] which makes the segmentation network more robust to small changes in the input is studied. The application of entropy minimization to image segmentation is straightforward and was done by previous work, so it is considered as baseline method. In the proposed setup, entropy minimization is capable of reducing the error caused by the corruptions by 6.9 % on average with the same hyper-parameters across a wide range of corruption types and levels. Our adaptation of adversarial-transformation-based optimization performed well across multiple corruption settings but no hyper-parameters performing well across all of them were found.

The proposed TTT methods include self-supervised reconstruction with masked autoencoders which aligns the representation of the encoder with the given image, deep intersection-over-union surrogate which minimizes the estimated error caused by the domain shift and a deep mask refinement network which is trained to eliminate noise from the predicted mask. Since the domain shifts are not known in advance, they are simulated by adversarial transformations. The results show that the TTT approaches like self-supervised reconstruction or mask refinement greatly outperform the entropy minimization baseline when only a small number of images is available, with corruption error reduction of 11.86 % by the reconstruction-based method and 11.33 % by the mask-

refinement-based method. The deep IoU-estimation-based method performs similarly to entropy minimization in terms of best IoU achieved (6.54 % corruption error reduction %) but is more stable across different hyper-parameter settings. Notably, it rarely diverges within the 10 optimization steps evaluated, which is desirable since there is no stopping rule for the test-time optimization. The combination of the best-performing methods, reconstruction and refinement, reduces the corruption error by 15.47 %.

The majority of the experiments are conducted on top of the reconstruction-pretrained Masked Autoencoders (MAE) model proposed in [15] finetuned on a small segmentation dataset on a simple binary-segmentation task. This allows for fast experimentation with different test-time adaptation methods, including image reconstruction. The most promising methods are further evaluated on the recently released the SegmentAnything Model (SAM) [21]. SAM is a model trained on 1 billion images and has been shown to perform well across a wide range of benchmarks. It was published only a month before the thesis, making it one of the most recent and most general segmentation models available.

The mask-refinement module is also explored as a post-processing method, as opposed to TTA. These experiments show promising results even on non-corrupted images, improving the Intersection over Union (IoU) of the binary segmentation model by 3 %.

Link to code repository: `https://github.com/klarajanouskova/TTA-SEG`

In summary, the contribution of the thesis are the following

- Bridging the TTA methods for segmentation on natural images with methods proposed in medical imaging domain.
- A general validation framework for hyper-parameter tuning inspired by Imagenet-C is proposed.
- Establishing a diverse set of TTA baselines for segmentation.
- Test-time training methods are proposed for the segmentation task for the first time and are shown to outperform the test-time adaptation methods.
- The methods are evaluated on two different models, showing both a custom model trained on a small segmentation dataset and a very recent general model trained on 1 billion masks can benefit from test-time adaptation methods.
- The adversarially-trained mask-refinement module is shown to perform well even as a post-processing step.

Chapter 2

Related Work

This chapter first introduces the problem of domain shift and its different variations in Section 2.1, with a focus on covariate shift which is studied in the thesis. Test-time adaptation and test-time training are then described in Section 2.2 where its relationship to the many similar tasks is explained. Section 2.2 provides an overview of the existing test-time adaptation methods. Section 2.3 is about self-supervised learning since it is the basis of test-time training. The focus will be on masked-autoencoders of [15] since these are the basis of the models used in the experiments and optimizing the reconstruction loss of the masked autoencoder is one of the proposed test-time training methods. Finally, in Section 2.4, the Segment Anything [21] model used later in experiments is described.

2.1 Domain Shift

Let us denote the input space (features, covariates) as $\mathcal{X}$, the output space as $\mathcal{Y}$ and the source and target distributions over $\mathcal{X} \times \mathcal{Y}$ as $P_{\mathcal{S}}$ and $P_{\mathcal{T}}$, respectively. In the case of image segmentation, $\mathcal{X} = R^{\mathrm{H}\times\mathrm{W}\times 3}$ are images and $\mathcal{Y} = R^{\mathrm{H}\times\mathrm{W}\times\mathrm{C}}$ are the segmentation masks of C classes. It is assumed that a model $f : \mathcal{X} \to \mathcal{Y}$ is trained on a training set $\mathcal{S} = \{(x_i, y_i) \in (\mathcal{X} \times \mathcal{Y})\}_{i=1}^{n}$ drawn i.i.d. from $P_{\mathcal{S}}$.

After deployment, the model encounters samples $(x, y) \in \mathcal{X} \times \mathcal{Y}$ drawn from $P_{\mathcal{T}}$. In the optimistic case, $P_{\mathcal{S}} = P_{\mathcal{T}}$ and there is no domain shift. Otherwise, the following cases of domain shift are considered:

Covariate shift refers to the case where

$$P_{\mathcal{S}}(y|x) = P_{\mathcal{T}}(y|x) \tag{2.1}$$

but

$$P_{\mathcal{S}}(x) \neq P_{\mathcal{T}}(x) \tag{2.2}$$

Sometimes called sample selection bias, this shift kind has received the most attention in the computer vision community. Covariate shift naturally occurs in settings when labels are caused by the input features. Covariate shift may occur because different sensors were used to collect the features during training and deployment, such as mobile and DSLR cameras. When a classifier distinguishing cats and dogs is trained, it is unlikely that the training data contains all the possible dog and cat breeds and their combinations. But these breeds will be encountered after deployment. A self-driving system may be very good at detecting obstacles in European cities, but it may see kangaroos for the first time only after it was deployed in Australia. Other examples of covariate shifts are changes of weather, sketches to photographs, synthetic to real images or differences in light conditions such as day to night.

Label shift, also referred to as prior or target shift, is the reverse of the covariate shift where

$$P_{\mathcal{S}}(x|y) = P_{\mathcal{T}}(x|y) \tag{2.3}$$

but

$$P_{\mathcal{S}}(y) \neq P_{\mathcal{T}}(y) \tag{2.4}$$

Label shift is often encountered in diagnosis, where diseases (labels) cause the symptoms (features). For instance, the prevalence of different diseases varies seasonally. Similarly, a dataset collected before a cycling infrastructure was built is going to contain much more cars than bicycles compared to the current reality.

Concept shift happens when

$$P_{\mathcal{S}}(y|x) \neq P_{\mathcal{T}}(y|x) \tag{2.5}$$

that is, the relationship between inputs x and outputs y changes. Some concepts have different meanings regionally. For instance, in most countries, a nod of the head means agreement while in some, it indicates refusal. A female height of 170 cm is considered as average height in Sweden but tall in Spain. Other concepts, such as what is considered fashionable, change over time.

The techniques that aim to transfer the knowledge obtained from the training (source) data distribution to the test (target) data are referred to as domain adaptation, a special case of transfer learning where the task remains the same. A comprehensive overview of domain adaptation in computer vision can be found in [22].

2.2 Test-Time Adaptation

Domain adaptation methods share a lot in common with semi-supervised learning [23], [24] and may exploit strategies such as generative models [25], [26]. In practice, this is often infeasible since source data may not be available for example for privacy reasons, or we may only have a small number of target domain images available. In continually evolving environments, the distribution may have already changed by the time adaptation on a whole target dataset is completed. Various modifications of the traditional domain adaptation scenario tackling the aforementioned limitations have recently emerged, for example by considering no access to source data or a continual domain shift [7], [9], [18], [27]. In particular, test-time adaptation methods assume no source data is available and aim to exploit the information from as little as a single target domain image.

Both Test-Time Adaptation (TTA) and Test-Time Training (TTT) could be described as source-free unsupervised domain adaptation. A model trained on a source domain has to adapt to a target domain without access to the source data and without any labels. Let us first describe related terms and also the difference between similar but different setups that may be easily confused:

Continuous domain adaptation assumes the target data change continually, as opposed to a static target distribution. Furthermore, it is important to avoid catastrophic forgetting (previous knowledge). In this work, we make no assumption about the relationship between the distribution of subsequent samples, each sample could come from a different distribution. For this reason, our models are initialized to the pretrained weights before adapting to a new image and catastrophic forgetting is not a concern. While single-image adaptation methods can be extended to continual learning, it is not clear that methods performing better in the single-image setup will also perform better in batch or stream of data mode.

Domain generalization aims for a strong model that would generalize to unseen domains without any adaptation. Common approaches are domain-invariant representation learning, data augmentation or data generation. In contrast, this work focuses on adapting a pre-trained model to become a specialist in the current domain. It was shown in [7] that a stronger, more general model can lead to better adaptation results, making these directions complementary. However, without adaptation, training a very general model on a large set of distributions may harm the model's performance on the individual distributions when compared to specialized models with carefully optimized data augmentation [28], [29].

Online domain adaptation expects a stream of data as input, possibly a single one, as opposed to a whole dataset. Test-time adaptation methods can be sued for online adaptation but generally, online adaptation techniques do not assume the source data are

not available.

Zero-shot segmentation requires the model to directly perform predictions for previously unseen classes without any adaptation to these classes.

The difference between TTA and TTT is that TTA methods such as [12], [20] can be applied to arbitrary pre-trained models without any additional constraints while TTT methods like [16]–[18] require modifications to the training process. However, not all works make this distinction and both will often be jointly referred to as test-time adaptation throughout the thesis for simplicity.

2.2.1 Test-Time Adaptation Methods

The test-time training methods are based on optimizing a self-supervised task loss alongside a fully-supervised target task loss at training time. At test time, only the self-supervised task is optimized. Little is known about what makes a good self-supervised task for test-time training but a strong relationship between gradient correlation and improvement from test-time training is shown in [30]. First, [30] propose rotation prediction as the self-supervised task. Lately, it was shown that reconstruction with masked autoencoders is a very strong self-supervised task for test-time adaptation of classifiers by [16]. Further, the authors justify the success of the method under distribution shift in terms of the bias-variance trade-off. While training on large amounts of data reduces the variance compared to training on a single image, however, the model is biased by the training sample and self-supervised fine-tunning on the new sample helps to reduce this bias. Self-supervised learning and, in particular, reconstruction with masked autoencoders, will be further discussed in Section 2.3. The most common approach to test-time adaptation is to update normalization layers only. For instance, instead of keeping the statistics of the batch-normalization layer from the training set, these can be adapted on the test images [4], [5]. Another option is to update the parameters of the normalization layers via entropy minimization [6]. Both of these approaches have the advantage of not requiring too much compute, but are mostly outperformed by other methods. On classification tasks, entropy minimization is known to lead to poor results when the number of available images is small. A combination of self-supervised contrastive learning to refine the features and online label refinement with a memory bank is proposed in [31]. Recently, a method based on updating the parameters of the normalization layers of the network by optimizing it for robustness against adversarial perturbation as a representative of domain shift was proposed in [20], outperforming similar test-time adaptation approaches.

While more realistic domain shifts such as synth-to-real, day-night or different weather conditions are important, it is convenient to study the performance of different methods in a principled manner under a wide range of distribution shifts. Many TTA works adopt

the Imagenet-C [19] benchmark. The Imagenet-C dataset is an extension of Imagenet [32] where different corruptions such as noise, blur, fog or intensity transforms are used to modify the image, with different corruption levels. The TTA methods are evaluated on images with the highest level of corruption applied. While these showcase the power of the methods on images where performance degradation is the greatest, it is not known how the method performs on clean data or milder corruptions. For instance, it is possible that different parameters are optimal for different levels.

Test-Time Adaptation Methods for Segmentation: Both [8] and [9] exploit augmented views of the input images to identify reliable predictions to train with. The recently proposed method [8] is based on the consistency of predictions between augmented views, which replaces prediction confidence for selecting reliable pixels. Cross entropy loss is then minimized on such reliable predictions, together with a regularization based on information entropy [33] to prevent trivial solutions. The evaluation assumed a full test set available at once. In [7], the performance of entropy minimization in a continual setup is explored, proposing parameter restart to tackle weight drift, significantly improving performance. Multiple evaluation scenarios are proposed: synthetic to real domain shift, as well as a synthetic dataset with varying weather conditions. Similarly, [9] also focus on continual adaptation. Again, augmentations of the images are generated to obtain more reliable predictions. Further, the network parameters are stochastically reset to their initial values to prevent forgetting of the source domain knowledge.

Test-Time Adaptation Methods for Medical Imaging: Segmentation-focused TTA seems to be a more active research area in the medical domain. For example, [12] propose an autoencoder with a predicted mask as an input and an enhanced mask as output. At test time, the segmenter is optimized to produce masks closer to the enhanced ones. However, this work assumes the whole test dataset is available at once. The work of [13] is similar to [12] but instead of a masked-autoencoder, a GAN-like discriminator trained end-to-end together with the segmenter is used, as well as an auxiliary reconstruction loss.

These works assume domain shifts specific to the medical imaging domain such as the use of a different scanner and thus make the assumption that only low-level features are affected. Under this assumption, these works typically optimize a small adapter only, typically the first few convolutional layers of the segmenter. This may not always be enough if we consider a more general class of domain shifts, such as previously unseen classes for class-agnostic models, where a shift is present in the output space as well and adapting higher-level features may be important.

Unfortunately, the progress in the medical imaging domain seems to be to some extent independent from traditional images and thus, these methods have not been evaluated on

traditional image segmentation benchmarks.

2.3 Self-Supervised Learning

When humans learn a new task, they generally require substantially less training samples than computers nowadays do. This can be explained by humans building on top of their model of the world built from prior experience on other tasks. Analogically, deep neural network pretraining consists of learning useful representations from different data or a different task than the target one. It has been used to mitigate the issue of limited data and to speed up the training process. Up until recently, supervised pretraining on the ImageNet [32] classification dataset has been the most common approach. Recently, self-supervised pretraining has been dominating because of the availability of large-scale databases and new architectures. Self-supervised representation learning has allowed for training on large amounts of training data without the extra annotation cost, achieving impressive performance and generalization capabilities [15], [34], [35]

Before being adopted by the computer vision community, self-supervised pre-training first gained popularity in the natural language processing (NLP) domains. A common approach would be masked language modelling. Part of the text would be masked out, to be predicted by the neural network. Well-known examples of masked language modelling are [36], [37].

In vision, the first approaches have rather relied on classification tasks such as the prediction of an angle of image rotation, or contrastive, where the loss function minimizes the distance between positive samples (different viewpoints of the same object) while keeping the negative ones far enough. First, patch-based self-supervision tasks for transformer architectures would be based on predicting patch location or solving a jigsaw puzzle.

The authors of [34] argued that while image representations covariant to image transformation are beneficial for tasks such as 3D correspondence prediction, invariant representations are desirable for most semantic recognition tasks. This is also exhibited by the use of convolutional networks, which are translation invariant, and by the extensive use of data augmentation techniques.

There are also dense-prediction self-supervised task. Typically, an autoencoder is trained to reconstruct corrupted images, where the corruption may be for example noise, masking or blurring. Since predicting the probability of each possible image is intractable, first approaches inspired by NLP would discretize the output. However, training with simple mean square error has proven to work remarkably well in [14].

2.3.1 Masked Autoencoders

Masked Autoencoders (MAE) by [15] have shown to be particularly strong for self-supervised pre-training. The task is image reconstruction after a large portion of the image (75 % of patches) is masked. The architecture is built on top of a plain ViT [38] architecture. The authors show that by masking out such a large portion of an image, the performance on downstream tasks is much better and argue the models is forced to learn meaningful semantics from the data, rather than simply rely on interpolation from neighbouring visible patches. Furthermore, they enable efficient training with a very large encoder by only feeding the unmasked patches (25 % of input data) through the encoder. The rest of the data only needs to be processed by the decoder, using a learnt mask token, which is much lighter. A combination of these factors has made this approach work much better than the same idea applied to convnets. An overview of the architecture can be found in Figure 2.1, reprinted from [15]. For more details, we refer the reader to the original work.

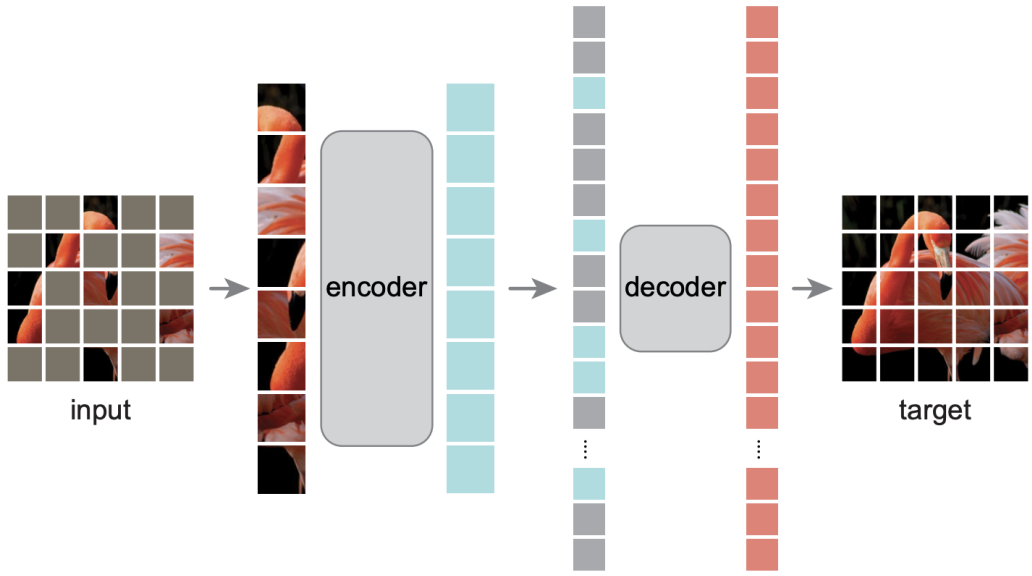


Figure 2.1: Masked Autoencoders (MAE) architecture of [15] (reprinted): In training, only 25 % of all patches are fed into the large-scale encoder, facilitating efficient training. The encoding of visible patches together with tokens of identical value representing masked patches is processed by a lightweight decoder.

In a subsequent work [39], the authors show how to apply pre-trained ViTs such as MAE to dense-prediction tasks, which are typically tackled by specialized architectures such as Swin transformers [40]. This will be covered in more detail in Section 4.1. In [41], the idea of randomly replacing patches by a mask token in the spatial domain is extended to the frequency domain with the idea that it helps to reveal the underlying

image patterns.

For more information on masked autoencoders as self-supervised vision learners in general, the survey [42] is recommended.

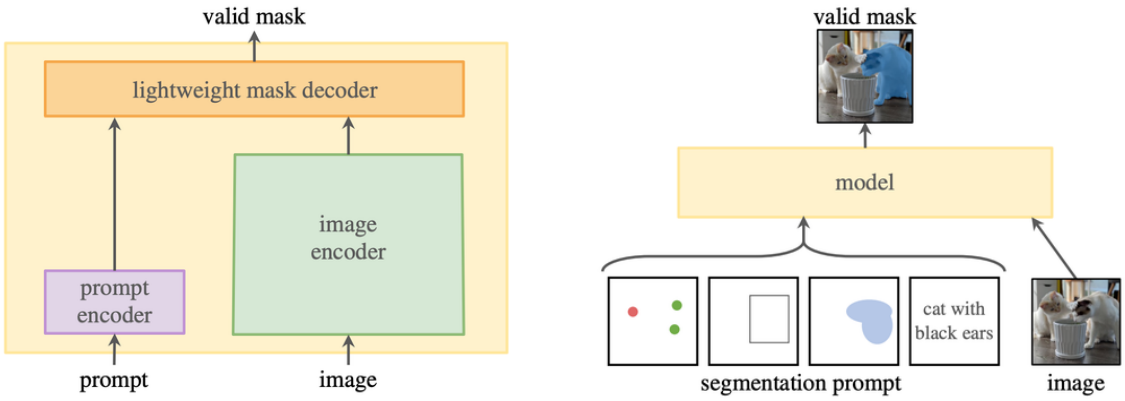
2.4 Segment Anything

the SegmentAnything Model (SAM) [21] is a large-scale pretrained model for image segmentation. It was first pre-trained in the same fashion as MAE from [15] but the architecture is not a plain Vision Transformer (ViT), instead, it is adapted for efficiency on images of higher resolution required for high-quality masks by using a window attention mechanism, rather than global attention. Further, the mask-decoder is a two-way transformer with prompt self-attention and two-direction cross attention: prompt-to-image embedding and image-to-prompt embedding. The model was trained on over 1 billion of diverse, high-quality segmentation masks and it was shown to perform well on many test datasets without any finetuning.

The model outputs instance segmentation based on prompts, which can be points (both positive and negative), bounding boxes, masks, as well as text prompts. Masks for the full image can be generated by giving a grid of points over the input image, followed by non-maxima suppression of the predicted masks. Prompts can be ambiguous, for instance, given a single point on a t-shirt of a person, it is not clear if the t-shirt only or the whole person should be segmented. For this reason, it is possible to output multiple masks. The pretrained model accepting text prompts was not released publicly.

The overall architecture can be split into three parts: A large image encoder performing most of the computation, a lightweight prompt encoder and lightweight mask decoder. The model is trained with a combination of focal loss [43] and the dice loss [44]. Apart from the masks, the mask decoder also output an estimate of the Intersection over Union (IoU) error. The IoU estimation is trained with the mean square error between the predicted mask and the ground truth mask.

Figure 2.2, reprinted from [21], provides an overview of the architecture and the prompts that can be used as input.



(a) (b)

Figure 2.2: Overview of the SegmentAnything Model (SAM) for prompted class-agnostic segmentation (reprinted from [21]). (a) The architecture: image encoder, prompt encoder and mask decoder. (b) different prompt types supported by SAM to specify the desired segmentation mask is shown. The model outputs multiple valid masks for each prompt to resolve ambiguity. .

Chapter 3

Image Segmentation and Datasets

This chapter gives an overview of image segmentation with a focus on the two tasks explored in the thesis, focused object class-agnostic semantic segmentation in Section 3.1 and point-guided class-agnostic instance segmentation in Section 3.2. Further, the datasets used for training and evaluation of models on these tasks are introduced.

Broadly, the goal of image segmentation is to divide the pixels of an image into regions where pixels assigned to the same region share common characteristics. Image segmentation simplifies the image for further processing and is important for many applications such as autonomous driving, image editing or visual tracking. Semantic segmentation is a task where the pixels are grouped together according to semantic classes such as dog, cat, car or road. All pixels of the same class form a single region, different instances of the same class are not distinguished. In instance segmentation, every object is assigned not only a semantic label but also an instance label. Panoptic segmentation is a combination of the two tasks. In contrast to image segmentation, the so-called stuff categories such as sky or grass need to be segmented. The instance labels for these categories are ignored since stuff classes can not be split into individual objects. A visual comparison of different segmentation tasks can be found in Figure 3.1

Recently, multiple so-called foundation models for image segmentation have been proposed. These are large models trained on millions of images that perform well across many domains. The models are typically prompt-based but can output segmentation of the whole image as well. Often, the full-image segmentation takes the form of open-world entity segmentation, a task similar to panoptic segmentation but without semantic labels. Such class-agnostic models are also of interest to us since their performance degrades on classes which were not in the labelled training set, which is a form of domain shift.

These models still tend to be outperformed by specialized models trained on the target datasets but are of great interest for learning with limited supervision or speeding up the annotation process. Improving performance of such general models on new domains is

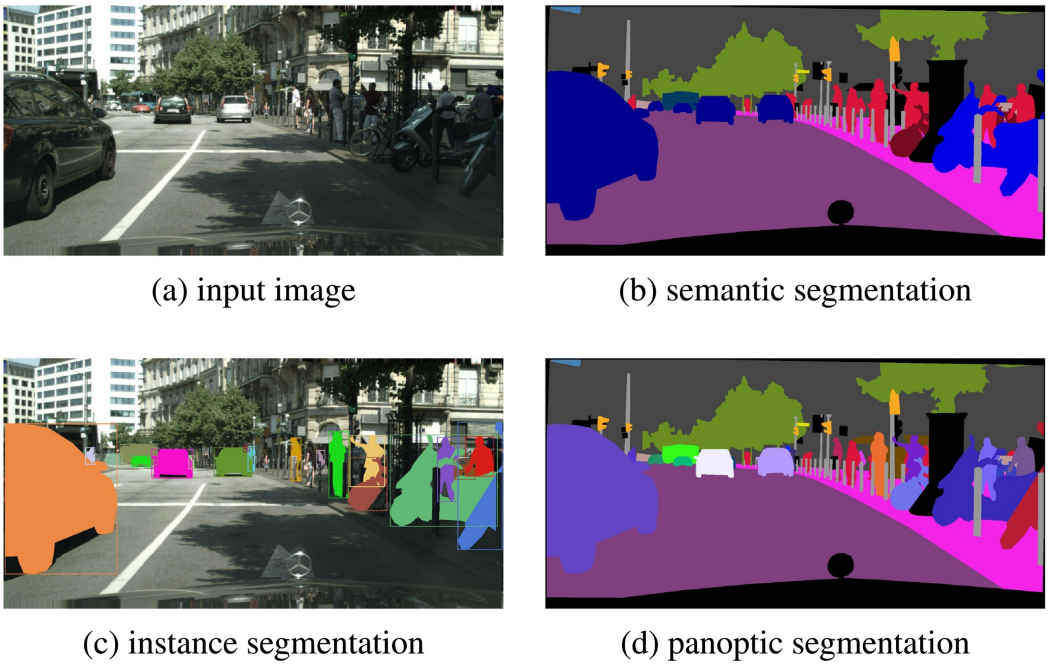


Figure 3.1: Image segmentation tasks: Comparison of outputs for the input image in the top left, reproduced from [45].

thus extremely valuable.

When this thesis project started, class-agnostic segmentation models were not very common. The most common one was saliency detection. The goal of saliency detection is to predict the most salient objects in an image, no matter what their class is. It can be thought of as segmenting the object according to how much attention it would get from a human looking at the picture.

However, as illustrated in Fig 3.2, the ground truth of saliency detection is often not well-defined and contains many mistakes. For this reason, the saliency detection datasets are only used to validate the model architecture for class-agnostic focused instance segmentation to make sure the performance is comparable to transformer-based state-of-the-art saliency detectors. Following prior work [46], the DUTS-TR [47] (10553 training, 5019 validation images) dataset based on ImageNet is used for training the model and a set of 5 standard saliency detection benchmarks is used for evaluation (OMRON [48], ECSSD [49], HKU-IS [50], PASCAL-S [51] and SOD [52]).

Foreground-background binary segmentation tasks were also considered, however, the available datasets only contain a single class such as car or bird, which makes them less favourable for our experiments.

Figure 3.2: Saliency ground truth problems. While some examples are wrongly annotated, others are often hard to define and subjective. One may ask the following questions: When does an object stop being salient enough to be annotated? Are parts of nature such as trees or stone objects? Should a bench be a salient object if there is a person sitting on it?

3.1 Focused Object Class-Agnostic Semantic Segmentation

Experimenting with a simple, class-agnostic model allows for fast experimentation. Because of the limitations of existing class-agnostic tasks described previously, a class-agnostic dataset generated from an existing semantic segmentation dataset is used.

The class-agnostic dataset creation consists of cropping bounding boxes (with a configurable offset varying the task difficulty) around connected components in the ground truth mask. The ground truth mask is also cropped in the same way and all pixels belonging to other classes than that of the particular connected component are set to zero. This setup can be described as focused object class-agnostic semantic segmentation.

While the task is well-defined, it has some limitations. For instance, in semantic segmentation, an object can be split into multiple parts resulting in multiple connected components, such as a sofa split by a person sitting on it, or a horse in front of the case. When cropped, that is, without the context of the full image, it may be very hard to say what the ground truth should be, even for humans. Also, when only a small part of an object of the same class as the focused one is visible, such as a part of a leg, it may again be hard to recognize that it really is an object of the same class, even for humans.

It would have been better to use an instance segmentation dataset to create a focused

object class-agnostic instance segmentation task. It is recommended to do so in future work. Another option is to adopt some of the recent class-agnostic open-world entity segmentation methods such as [21], [53], [54]. While these models have clear direct applications, they are more complicated and computationally expensive. As a result, a simpler model for early exploratory experiments is preferred.

Pascal-VOC The Pascal-VOC is a widely adopted dataset and compared to more recent datasets such as [10], it is considered small and easy. It is referred to as VOC in this work and is used for experiments on focused object class-agnostic semantic segmentation.

As already mentioned, the automatically generated dataset. In the training set, the noise is neglected - training on noisy datasets is common. To make sure results of TTA experiments reflect segmentation performance properly and are not influenced by these artefacts, the evaluation datasets are filtered to obtain a subset of 120 images with clear (to a human), well-defined ground truth. This set is referred to as VOC_{120}. A smaller subset of 20 images from VOC_{120} used for the more computationally demanding experiments is referred to as VOC_{20}

Only half of the classes from Pascal-VOC were used for model training and evaluation to facilitate future experiments on adapting to previously unseen classes. These experiments were not conducted in this work due to time constraints. Thus, this detail will be ignored and the dataset will simply be referred to as VOC. An interested reader can find the implementation details in the code.

3.2 Point-Guided Instance Segmentation

The second task explored in the thesis is based on the the SegmentAnything Model (SAM) [21]. This allows for the evaluation of TTA methods on a strong, general model. Further, since the model accepts various prompt kinds as input, it is possible to create prompts from ground-truth masks and evaluate them on standard datasets. This work adopts point prompts to create a point-guided instance segmentation task. For single-point prompts, the centre of mass is selected to represent each instance. The small number of instances where the centre of mass is not part of the instance is neglected but a more sophisticated method based on distance transform could be used to improve the ground-truth quality.

Other segmentation tasks were considered but the mapping of ground-truth to points is not as straightforward.

Cityscapes Similarly to other TTA works, the Cityscapes [10] dataset is adopted for TTA evaluation. It is a large-scale dataset focusing on semantic understanding of urban street scenes. The dataset was collected in 50 different cities in different seasons. The existing instance annotation for vehicle classes (car, truck, bus, on rails, motorcycle,

bicycle, caravan, trailer) and people classes (person and rider) are converted to point prompts for individual instances. To be able to use smaller GPUs while still being able to detect small instances, the image size is reduced to $\frac{1}{2}$ of the original input size by only keeping the center-part of the image (cropping the bottom/upper and right/left quarters of the image).

The dataset images, ground truth masks and point prompts are shown in Figure 3.3

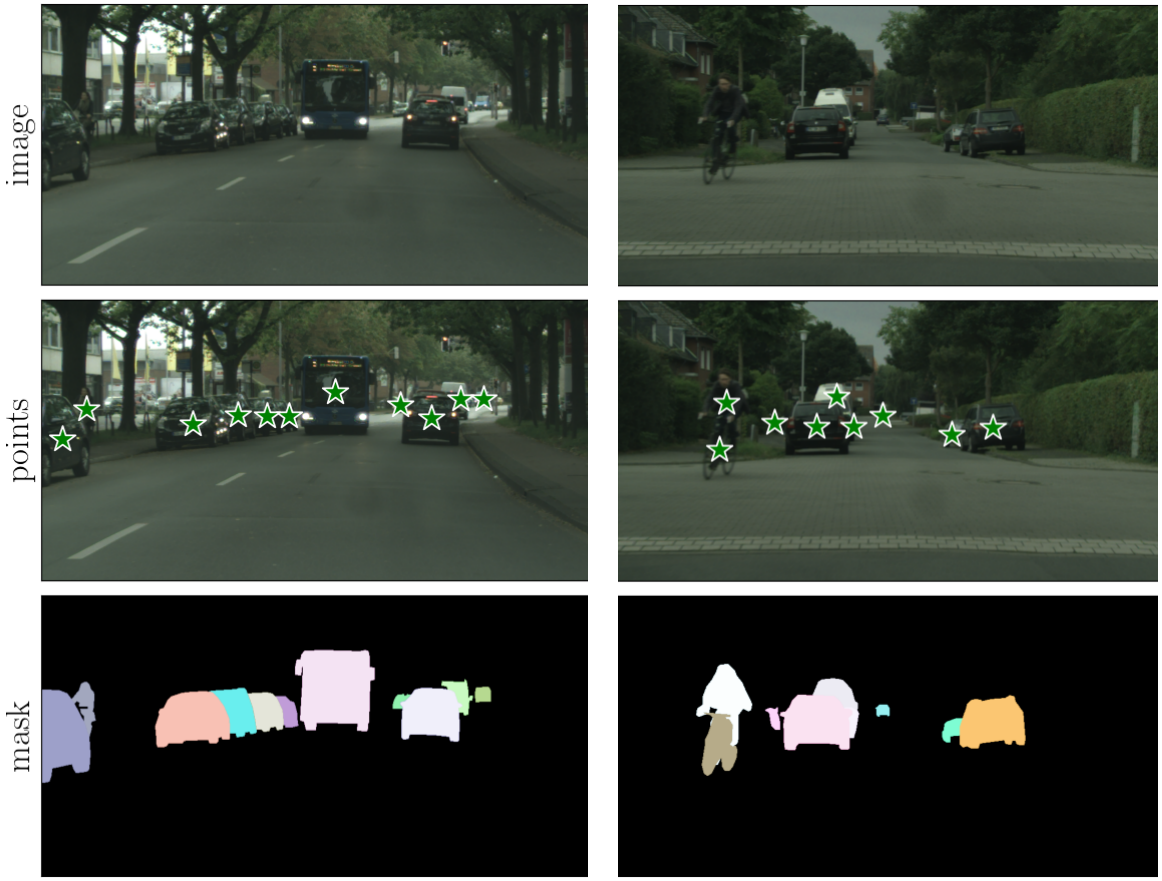


Figure 3.3: Point-guided instance segmentation. Each instance is specified by a single point ($\star$), its center of mass computed from the ground truth mask.

The dataset mostly contains images taken under mild weather conditions. Extensions with synthetic rain [55] and fog [56] added exist and are used throughout the experiments. These are referred to as Cityscapes - fog and Cityscapes - rain.

3.3 Corruptions

As discussed in Chapter 2, it is convenient to use synthetic corruptions to simulate domain shifts in a controlled environment. When the kind and level of domain-shift is known, a much more detailed evaluation of TTA methods under different circumstances is possible than on more realistic datasets. For this reason, we adopt a subset of the corruptions

proposed for the Imagenet-C benchmark [19] where the corruption type and level can be controlled and that can be applied to an arbitrary segmentation dataset.

The subset of selected corruptions is diverse and includes different intensity transformations, simulated adverse weather conditions, noise and blur. An overview together with implementation details is provided in Table 3.1.

An image corrupted by different transformations at different levels is shown in Figure 3.4.

The corruption-augmented variant of the VOC dataset is referred to as VOC-C (with the evaluation subsets denoted as VOC_{20}-C and VOC_{120}-C) and the corruption-augmented variant of the Cityscapes dataset is referred to as Cityscapes-C.

corruption	description
brightness	is an additive intensity transformation, $x_c = \text{clip}(x + b)$ where b controls the level.
contrast	is a multiplicative intensity transformation, $x_c = b(x - \overline{x}) + x$ where b controls the level and $\overline{x}$ is a per-channel mean of the image intensities.
frost	first crops a portion of one of the frost image templates at a random location of the same size as the input image, x_f. Then we compute $x_c = b_1 x + b_2 x_\text{f}$ where the weights b_1, b_2 control the level.
fog	first generates a heightmap x_h using the diamond-square algorithm [57], where the wibble is controlled by a parameter b_1. It is then combined with the input image as $x_c = \frac{(x + b_1 \cdot x_\text{h}) \cdot \max(x)}{\max(x) + b_1}$.
gaussian noise	is generated as $x_c = x + n$ where $n \sim \mathcal{N}(0, b)$ and b controls the level.
shot noise	is generated as $x_c \sim \frac{\text{Pois}(x \cdot b, \lambda=1)}{b}$ where Pois denotes the Poisson distribution and b controls the level.
spatter	simulates mud or water spoiling. The main idea of the algorithm is a combination of thresholding and blurring random noise.
defocus blur	first generates a disk kernel K with radius b_1 and alias blur b_2. The kernel is then used to filter each of the channels $x_c = \text{K}(x)$.
glass blur	first locally shuffles the pixels of a blurred version of the input image to obtain $x_\text{S} \sim \text{shuffle}(\mathcal{N}(x, b_1), b_2, b_3)$ and then blurs the output again to get the final corrupted image $x_c = \mathcal{N}(x_\text{S}, b_1)$ where b_1 controls the blur strength and b_2, b_3 control the shuffling level.
gaussian blur	corrupts the image by gaussian blurring $x_c = \mathcal{N}(x, b)$ where b controls the level.

Table 3.1: Corruptions and their implementation details, a subset from [19]. The input of the transformation is an image x normalized to the $(0, 1)$ range, the output is a corrupted image x_c. The clip function limits the values to the $[0, 1]$ range. This function is always applied to the output image after the transformation to obtain the final output $x_f = \text{clip}(x_c)$. For more details on the transformations and the values defining the level, please refer to the codebase.

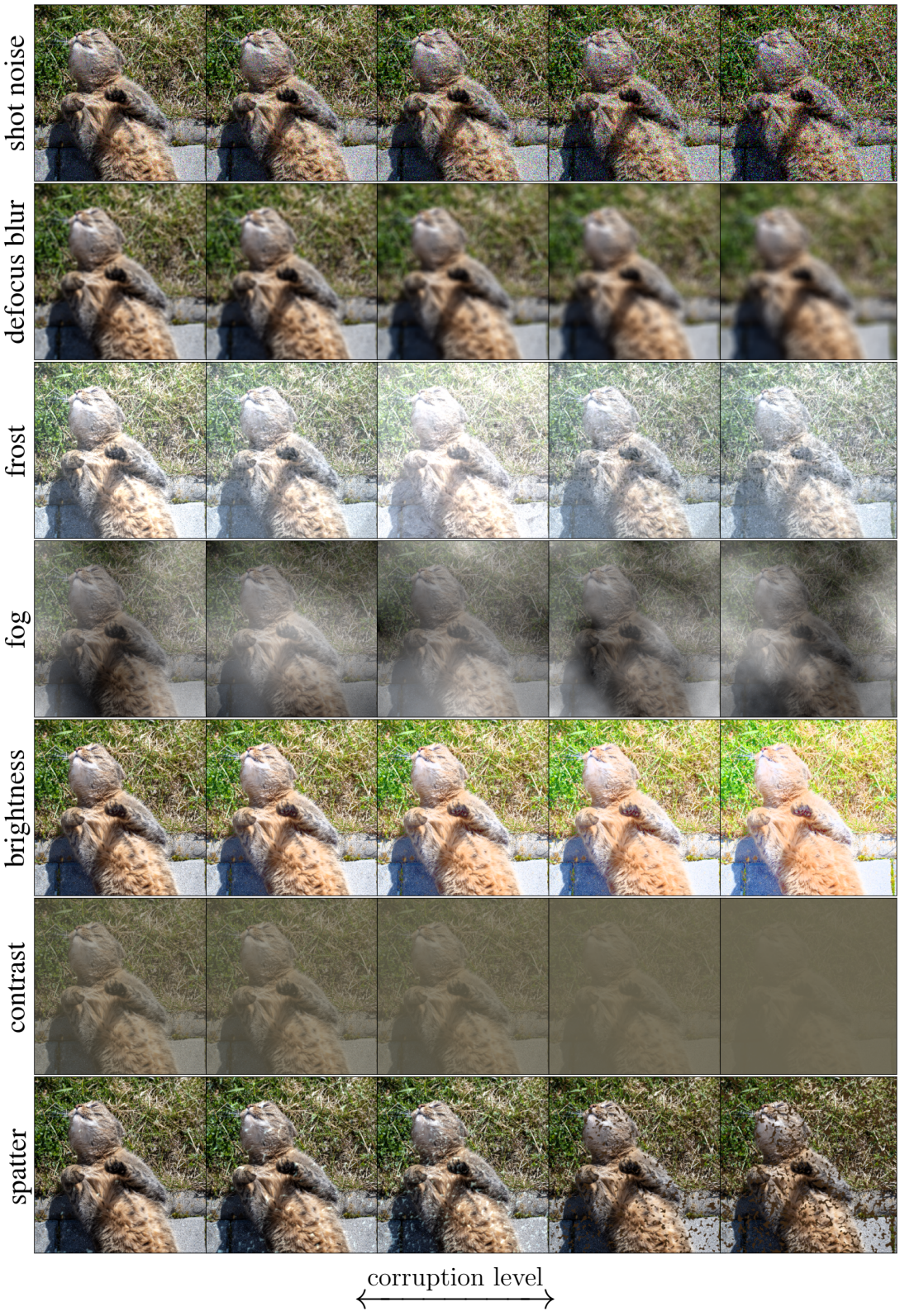


Figure 3.4: The effect of corruptions [19] on an input image. The corruption level increases from left to right.

Chapter 4

Methods

The first part of this chapter, Section 4.1, is dedicated to the architecture of our Class-Agnostic Semantic Segmentation Model (CASS) model for focused object class-agnostic semantic segmentation task introduced in Chapter 3 based on a MAE-pretrained plain Vision Transformer (ViT). The model for point-guided instance segmentation is adopted without any architecture changes and is described in Related Work, Section 2.4. In Section 4.3, the different methods for Test-Time Adaptation (TTA) and Test-Time Training (TTT) that were experimented with in the thesis are defined.

4.1 Class-Agnostic Semantic Segmentation Model

While there are many segmentation architectures publicly available based on the transformer architecture, these are built on top of segmentation-specific backbones. For instance in [21], [40], global attention is replaced by local window attention to adapt to the higher-resolution input images compared to those needed for image classification. It is not clear how important the effect of large-scale pre-training of the self-supervised task is for test-time training but it was shown that the pre-trained MAE is particularly good for classification TTA. Further, it is non-trivial to train non-pretrained transformer models on small datasets and pretraining can speed up the time until convergence, reducing computation requirements. Since the MAE pretrained models have only been released for plain ViT, it is necessary to design an architecture based on it.

The architecture is composed of three parts. A shared encoder e^{ω} with parameters ω, a reconstruction decoder d_r^{ψ} with parameters ψ, and a segmentation decoder d_s^{φ} with parameters φ. Both the encoder and the segmentation decoder are plain ViTs. The segmentation network is a ConvNet. The joint model architecture is visualized in Figure 4.1.

The reconstruction of an image x can be obtained as $r = d_r^{\psi} \circ e^{\omega}(x)$ and the segmen-

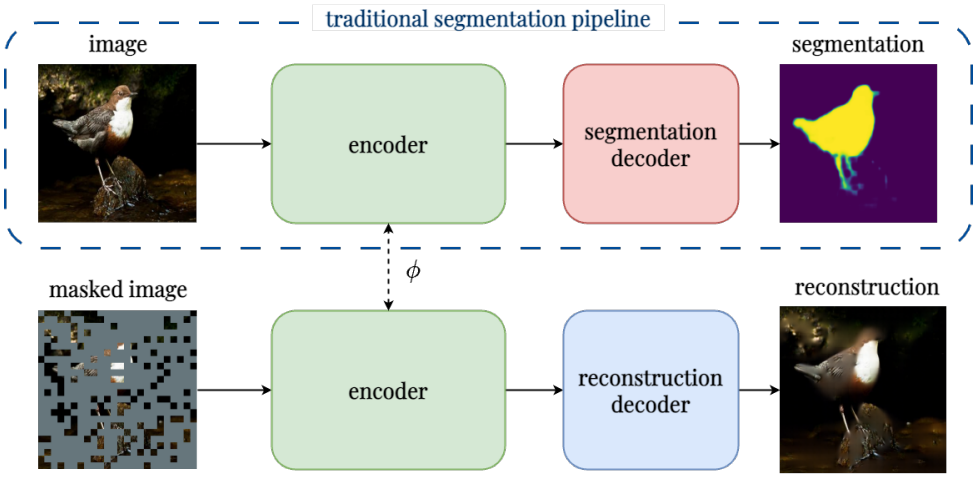


Figure 4.1: Joint reconstruction and segmentation architecture. The model consists of a large shared encoder and two lightweight decoders. In the thesis, the architecture is a transformer-convnet hybrid: The encoder and the reconstruction decoder are ViT while the segmentation decoder is a convolutional neural network specifically designed for the ViT backbone.

tation as $s = d_s^{\varphi} \circ e^{\omega}(x)$. The masking operation for reconstruction is considered as part of the decoder to simplify the notation.

Segmentation decoder When the field was still dominated by convolutional neural networks, the go-to architecture for segmentation was the U-Net [58], which can be built on top of any of the traditional backbones used as an encoder. The U-net is designed on the premise that the backbone progressively downscales the image resolution, which is not true for the ViT. Generally, most of the pretrained self-supervised models are not segmentation specific, as a majority of the work focuses on classification as a downstream task.

The issue of how to best use pretrained plain ViT [38] encoders for dense-prediction tasks has been addressed in [39], focusing on extracting features for detection. In the traditional hierarchical approach such as the U-Net, features are progressively downsampled in the encoder backbone branch. Then the decoder combines the features at different resolutions and different levels of abstraction. However, the plain ViT operates at the same resolution at all levels. The authors conclude experimentally that it is enough only to use the final encoder output layer and downsample it to different resolutions.

Inspired by these findings, we propose a U-Net-like decoder but with features from the final encoder layer only. The decoder has a variable size of n blocks. In the i-th decoder block, the features from the previous block (or the last encoder layer in the first block) are pre-processed by two convolutional layers and then downscaled by a factor of s^{n-i} with a strided convolution. These features are then merged by concatenation with upscaled features from the previous block, followed by two convolutional layers (with

batch normalization and ReLU activations). Finally, the decoder features go through a linear classifier with a sigmoid activation to obtain the segmentation masks. An overview of our architecture can be found in Figure 4.2.

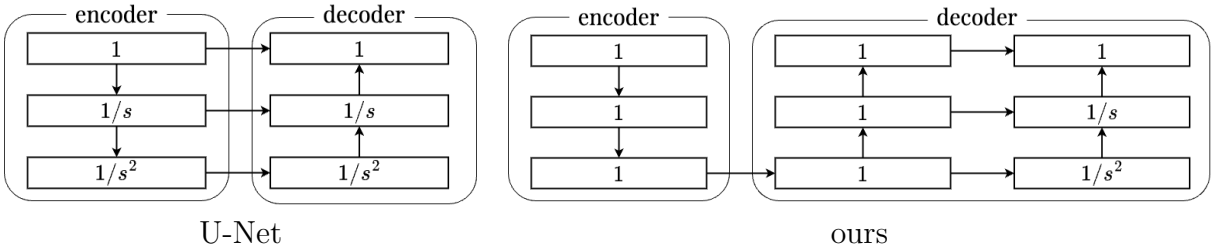


Figure 4.2: Comparison of our segmentation decoder architecture with U-Net. Our decoder is inspired by the findings from [39]. The decoder input is only the final, single-scale feature map of the encoder. In each decoder block, this feature map is then processed by two convolutional layers (at the original resolution) and then downsampled to create a hierarchical structure.

Other, more simple, variants were explored, such as reusing the reconstruction decoder architecture. For smaller resolution inputs such as those used by the plain-ViT MAEs, the original patch size is too large and results in artefacts in the output. Since the quality of the intermediate representation is more important than the final reconstruction quality, this is not an issue for the reconstruction decoder architecture. Experiments with smaller patch size would require significant effort to find reasonable finetuning hyper-parameters since the linear patch embedding would need to be retrained. Simple sequential CNN decoders were also explored but the mask quality was still not satisfactory. Visualization of the output of these intermediate models can be found in a WandB report notebook enclosed as Appendix D.

4.2 Training

In this section, pretraining and joint finetuning of the previously described model is overviewed. The pretraining of SAM is detailed in Section 2.4 of Related Work.

4.2.1 Reconstruction pre-training

This work builds on top of the large-scale pretrained MAE model of [15]. Unless stated otherwise, all the models are the pre-trained ViT-b models. This part describes the work done by the authors of [15]. The models were trained on the ImageNet-1K [32] on random crops of size 224×224. No colour augmentations were used since it only degraded the results. This is in contrast with the contrastive learning approaches, where the lack of data augmentation severely degrades the performance. This difference can be

explained by the random masking strategy of MAE, which can be considered a kind of data augmentation. The masking strategy is to mask 75 % of patches at random, which led to the best performance on downstream tasks.

The loss function is the MSE, with the addition that the target pixel values are normalized per-patch, which showed superior performance on downstream tasks.

The models were trained on 8 nodes of 8 V100 gpus each for 800 epochs. The effective batch size was 4096 and the total training time was 42 hours. For more details and hyper-parameters, please refer to the official repository on github (`https://github.com/facebookresearch/mae`).

4.2.2 Joint finetuning

The MAE-pretrained model is jointly fine-tuned on both reconstruction and segmentation. For debugging and visualization purposes, the regular mean square error loss function without per-patch normalization is used for training. Further, the input image resolution is increased to 384×384 to improve mask quality.

The reconstruction loss is computed over the non-masked patches as

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(r_i - x_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(d_r^{\psi} \circ e^{\omega}(x_i) - x_i)^2 \tag{4.1}$$

where r_i, x_i are the values of the i-th pixel in the reconstructed image $r = d_r^{\psi} \circ e^{\omega}(x)$ and the input image x, respectively, and N is the total number of pixels. Further, e^{ω} is the encoder and d_r^{ψ} is the reconstruction decoder. The overall reconstruction loss is computed as mean of the $\mathcal{L}_{\text{MSE}}$ between the masked input image patches and their reconstruction.

Following the original work, the encoder only sees the unmasked 25 % of input patches to save computational resources. The masked patches are represented by a learnt mask token when fed into the decoder.

Following [46], the segmentation loss is a weighted sum of Binary Cross Entropy (BCE) with Intersection over Union (IoU):

$$\mathcal{L} = \mathcal{L}_{\text{MSE}} + \lambda(\mathcal{L}_{\text{BCE}} + \mathcal{L}_{\text{IoU}}) \tag{4.2}$$

where

$$\mathcal{L}_{\text{BCE}} = \sum_{i=1}^{\text{N}}[\text{y}_i \cdot \log(\text{s}_i) + (1 - \text{y}_i) \cdot \log(1 - \text{s}_i)] \tag{4.3}$$

$$\mathcal{L}_{\text{IoU}} = 1 - \frac{\sum_{i=1}^{\text{N}}(\text{y}_i \cdot \text{s}_i)}{\sum_{i=1}^{\text{N}}(\text{y}_i + \text{s}_i - \text{y}_i \cdot \text{s}_i)} \tag{4.4}$$

and $\mathrm{s} = d_s^\varphi \circ e^\omega(x)$, $\mathrm{s} \in \mathbb{R}^{\mathrm{H}\times\mathrm{W}}$ is the segmentation output for image x, $\mathrm{y} \in \{0,1\}^{\mathrm{H}\times\mathrm{W}}$ is the binary segmentation ground truth, s_i and y_i are the values of the i-th pixel in s and y, respectively, and $\mathrm{N} = \mathrm{H} \times \mathrm{W}$ is the number of pixels in the input image of height H and width W. Further, $e^\omega, d_s^\varphi, d_s^\varphi$ are the encoder, segmentation decoder and reconstruction decoder, respectively. The parameter λ controls the weight of the segmentation loss.

Note that the Intersection over Union (IoU) as a measure of similarity of two sets, defined as $\frac{A\cap B}{A\cup B}$, is not differentiable since it expects both binary inputs and outputs and the thresholding operation is not differentiable, the IoU loss is a differentiable approximation from [59].

The encoder is fed with all the patches since the full image is necessary for segmentation.

The training objective can finally be formalized as

$$\omega^\star, \varphi^\star, \psi^\star = \arg\min_{\omega,\varphi,\psi} \frac{1}{|\mathcal{T}|} \sum_{(x,y)}^{\mathcal{T}} \mathcal{L}_{\mathrm{SEG}}(x, y) \tag{4.5}$$

where

$$\mathcal{L}_{\mathrm{SEG}}(x, y) = \mathcal{L}_{\mathrm{MSE}}(d_r^\psi \circ e^\omega(x), x) + \mathcal{L}_{\mathrm{IoU+BCE}}(d_s^\varphi \circ e^\omega(x), y) \tag{4.6}$$

and $\mathcal{T} = \{x_i, y_i\}_{i=1\ldots N}$ is the training set, $(x, y) \in \mathcal{T}$ are an image and its corresponding ground truth and ω, φ, ψ are the parameters of encoder e^ω, reconstruction decoder d_r^ψ and segmentation decoder d_s^φ, respectively.

The encoder and reconstruction decoder parameters ω and φ are initialized to the parameters of the reconstruction MAE-pre-trained model.

4.3 Test-Time Adaptation methods

In this section, the different test-time adaptation methods used throughout the experiments are described in detail. Since multiple of the methods are based on simulating domain shift with adversarial attacks, the first part will be dedicated to how these attacks are implemented.

An overview of the overall TTA framework can be found in Figure 4.3.

4.3.1 Adversarial Attacks

The idea of adversarial attacks is to fool a deep neural network to change its prediction by altering the input in a constrained manner, for instance, so that the change in input is smaller than a predefined threshold ε. The attacks can be targeted and untargeted, in

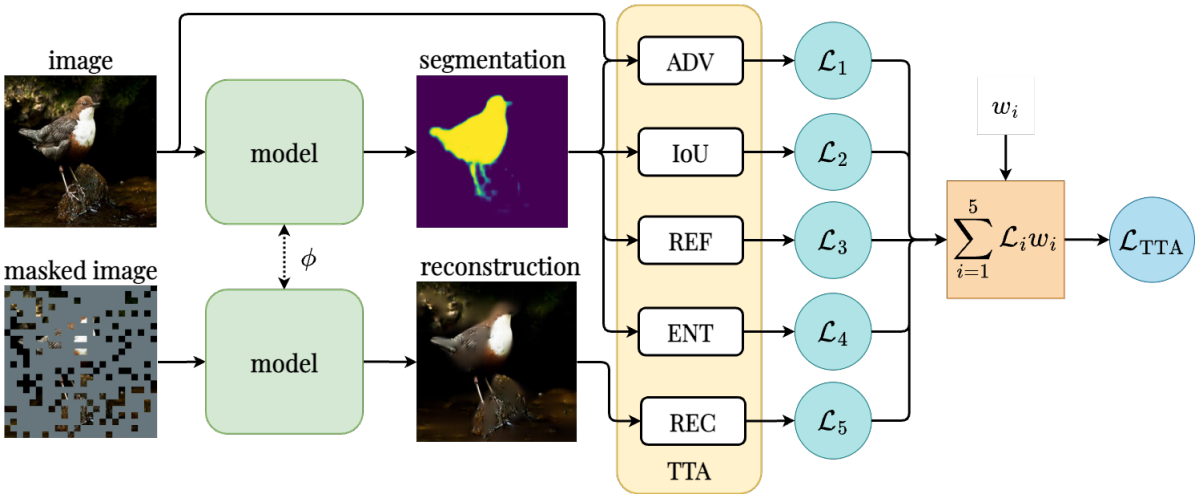


Figure 4.3: The Test-Time Adaptation (TTA) framework. The TTA loss is computed as a weighted sum of the losses of individual methods. The REC method is based on the reconstruction output of the model while the ADV, dIoU, REF and ENT methods are based on the predicted mask. The ADV method also receives the original image as input.

untargeted attacks, the input should be altered so that the network outputs a different prediction, but it doesn't matter what the new prediction will be. In the case of binary segmentation, it essentially means inverting the segmentation mask. But for semantic segmentation of multiple classes, it means that the prediction of each pixel should be changed to any other class than the original prediction for that pixel. Another option is a targeted attack when the network should be fooled to output a specific segmentation mask.

The inversion attack in early iterations creates realistic corrupted masks similar to those caused by domain shift by first changing the output in locations that are the easiest to be confused by the model. But in later iterations, the inverted masks are no longer realistic. Further, we observe that apart from wrongly (not) segmenting parts of different objects, the domain shift often results in very noisy masks. This mask corruption is simulated with a targeted attack where the target is generated by adding blurred random gaussian noise to the image. Note that while we make use of the knowledge of how masks are typically corrupted by different kinds of domain shifts, very general targeted attack strategies are proposed which do not simulate any domain shift in particular.

The inversion and the random attack ground truth masks are shown in Figure 4.4.

When computing attacks at test-time, the Fast Gradient Sign Method (FGSM) [60] is used to generate the attacks for time efficiency. Otherwise, an iterative variant which could also be considered as an attack based on Projected Gradient Descent (PGD) [61], [62] is used. The evolution of the segmentation prediction over the PGD iterations is shown in Figure 4.5.

More sophisticated approaches for adversarial attacks on segmentation that could be

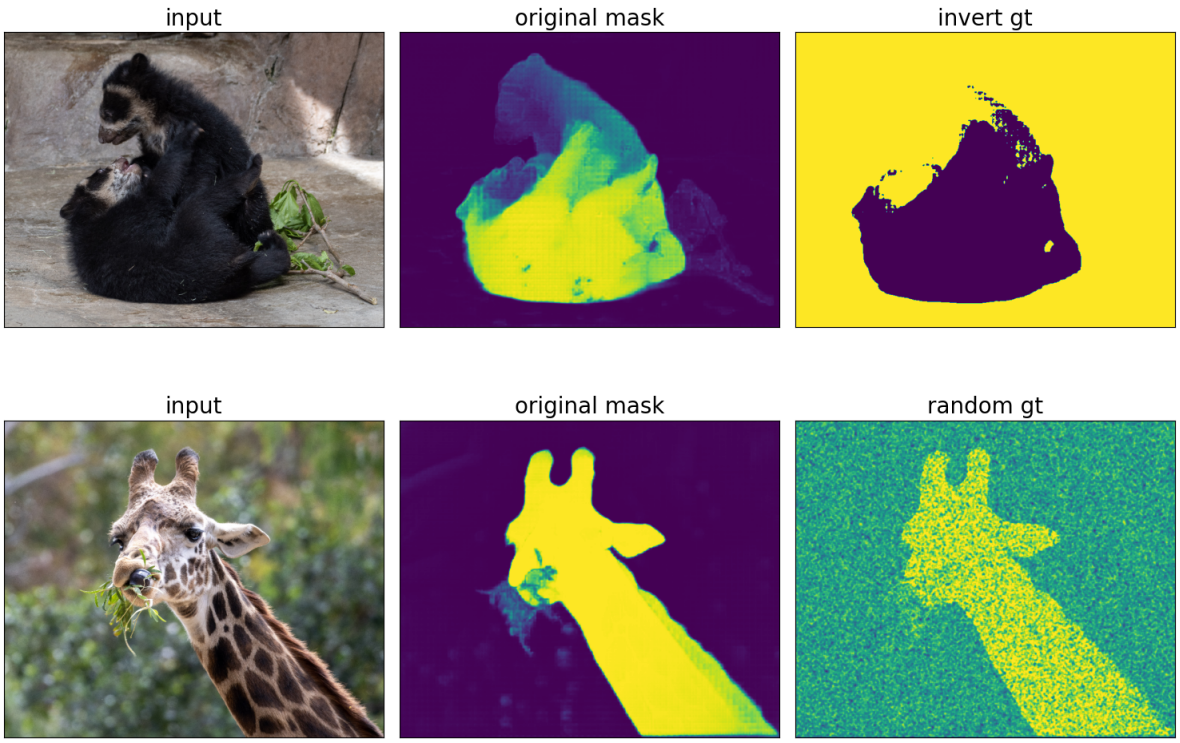


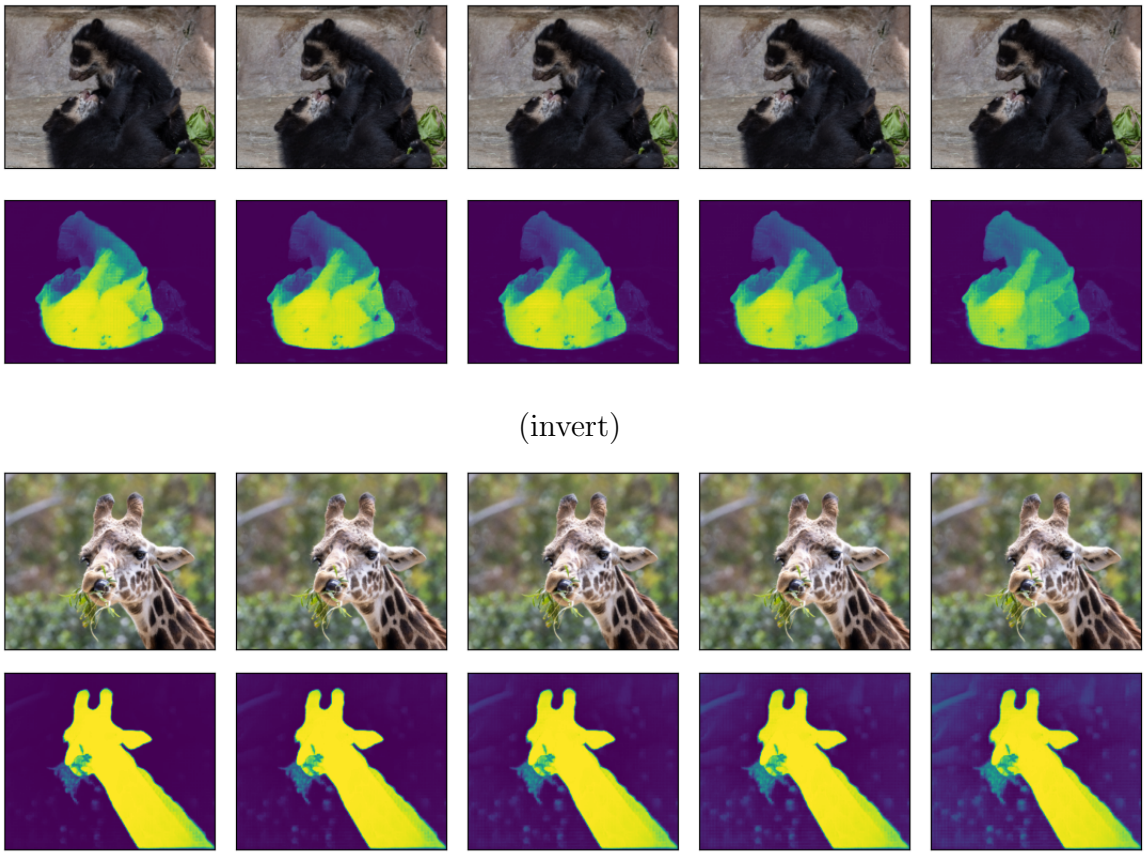
Figure 4.4: Target options for a targeted adversarial attack on segmentation.

explored in future work exist but are not covered in the thesis and could be explored in future work.

4.3.2 Entropy Minimization (ENT)

The Entropy-Minimization-Based Test-Time Adaptation (ENT) method minimizes the entropy of the segmentation predictions. In the context of learning with limited supervision, it was proposed in [63] for semi-supervised learning. In effect, it is the same as pseudo-labelling [64], both methods reduce class overlap. In TTA, there is no labelled set that could be leveraged as regularization like in semi-supervised learning but the methods were shown to work for TTA as well [6]. It was also shown that larger batch size and updating the parameters of the normalization-layers only improve stability of the method. But on segmentation, a dense-prediction task, adapting to a single image can lead to positive results [7].

The method is simple, computationally efficient and widely adopted as a baseline. More formally, the method minimizes the entropy of the segmentation model predictions $\mathrm{s} = d_s^{\varphi} \circ e^{\omega}(x)$ for an image x over the parameters of the normalization layers (such as batch [65], layer [66] and group [67] normalization) denoted as ω_n for the encoder e and φ_n for the segmentation decoder d_s:



(invert)

(random)

Figure 4.5: Adversarial attack segmentation prediction evolution. It can be seen that for the inversion attack, the pixel predictions changed in the first iterations are those that are the easiest to be confused by the network. Prompted by a point on the lower bear, the network is not confident about whether the second bear should be segmented or not. When attacked, the prediction for the second bear becomes more and more confident and vice versa. The random attack progressively introduces more and more background noise.

$$\omega_n^\star, \varphi_n^\star = \arg\min_{\omega_n, \varphi_n} \sum_{i=1}^{\mathrm{N}} \mathrm{s}_i \cdot \log(\mathrm{s}_i) \tag{4.7}$$

where s_i corresponds to the i-th pixel of the segmentation prediction s.

4.3.3 Reconstruction (REC)

In principle, the Reconstruction-Based Test-Time Training (REC) method is very simple. A network is trained for both segmentation and reconstruction at training time and at test time, where segmentation ground truth is not available, only the reconstruction is optimized. The intuition is that during joint fine-tuning, a shared representation with correlated gradients was created in the decoder. Reconstruction training finetunes the representation to the current image. This could be said about many self-supervised tasks

though. The reason why we opted for MAE-based reconstruction is that dense prediction tasks have been shown to work particular well as pre-training tasks for image segmentation [14]. Further, the same self-supervised task has already proven powerful for image classification in [16]. There are multiple options of what parameters should be optimized during test-time training with reconstruction objective, we experiment with the following two options;

First, optimizing only the encoder

$$\omega^\star = \arg\min_{\omega} \mathcal{L}_{\text{MSE}}(d_r^\psi \circ e^\omega(x), x) \tag{4.8}$$

second, optimizing both the encoder and the decoder

$$\omega^\star, \psi^\star = \arg\min_{\omega,\psi} \mathcal{L}_{\text{MSE}}(d_r^\psi \circ e^\omega(x), x) \tag{4.9}$$

where x is the input image and e^ω, d_r^ψ are the encoder and the reconstruction decoder.

Intuitively, the first option without updating the reconstruction decoder seems natural since it restricts the change in encoder representation but in [16], the full model was updated during TTA with good results so both options will be compared in this work.

While it is expected the joint reconstruction and segmentation training creates shared representation with correlated gradients, there is no mechanism enforcing it during the training process. In future work, we propose to investigate gradient-alignment strategies such as gradient regularization during training.

4.3.4 Deep IoU Loss Surrogate (dIoU)

The idea of the Deep-Intersection-over-Union-Based Test-Time Training (dIoU) is that instead of choosing an arbitrary self-supervised task such as reconstruction or rotation-prediction, one can train a surrogate loss in form of a deep net to learn a segmentation-specific loss function. The loss function can then be used at test-time for self-supervised learning. In particular, the deep IoU surrogate loss is a neural network that is trained to predict the IoU loss between a mask predicted by a segmentation model on an image from the training distribution and a mask predicted on a domain-shift-inducing transformation of the same image. Adversarial attacks are harnessed to simulate domain shifts without prior knowledge of what kind of shifts will be encountered at test-time. The attacks are generated with Projective Gradient Descent restricting the output to the valid image range. The method requires access to training distribution images for training.

The surrogate loss is essentially learning to predict how much the performance was deteriorated due to domain shift. To make it as domain-independent as possible, the sur-

rogate loss training can not update the segmentation network parameters during training and only receives the mask, not the input image. An overview of the method can be found in Figure 4.6.

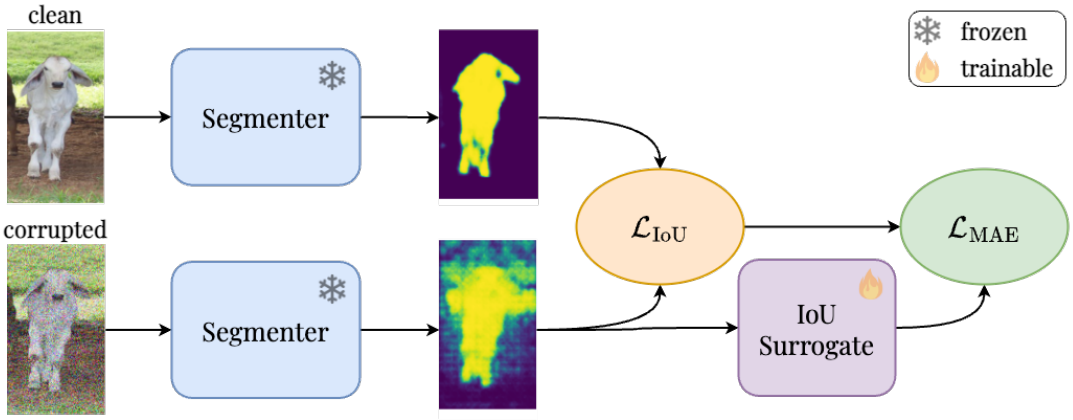


Figure 4.6: IoU surrogate module training. The segmentation pre-trained network (segmenter) receives two images as input: A clean image and a synthetically corrupted image. The synthetic corruptions simulate domain shift. The output are two masks, clean and corrupted. The IoU surrogate loss network is then trained to predict the IoU between the clean and the corrupted mask given the corrupted mask as input only. The IoU surrogate only recieves the mask as input and no gradients can flow back to the segmented. The IoU surrogate can be interpreted as a model estimating the increase in IoU loss of a mask due to domain shift.

Formally, at test-time, the IoU is estimated by a pre-trained IoU loss surrogate network $f_{\widehat{\text{IoU}}}$ based on the predicted segmentation mask $s = d_s^{\varphi} \circ e^{\omega}(x)$ of an input image x. The estimated value is then minimized as follows

$$\omega^{\star}, \varphi^{\star} = \arg\min_{\omega,\varphi} f_{\widehat{\text{IoU}}}(\text{s}) \tag{4.10}$$

where $e^{\omega}, d_s^{\varphi}$ are the encoder and the segmentation decoder.

SAM-Intersection-over-Union-Based Test-Time Training (sIoU)

While surrogate losses trained on top of the pretrained, frozen segmentation network are studied, a method TTT exploiting the IoU estimation released with SAM is also explored. The approach is similar to ours but with notable differences. The gradients during training of the surrogate can flow through the whole network, not just the parameters dedicated solely to the surrogate loss. Further, the IoU estimate was not trained with domain shift in mind. Finally, the IoU is predicted with respect to the ground-truth mask, as opposed to the training distribution mask in our case.

In future work, experiments comparing the performance of deep IoU surrogate depending on the network input (mask only or mask + image) and the training procedure (whether it can update segmentation network parameters) in more detail should be done.

4.3.5 Segmentation Refinement (REF)

The Mask-Refinement-Based Test-Time Training (REF) method is motivated similarly to the dIoU method. While dIoU predicts the segmentation error from a mask, the REF method goes a step further and trains a deep network to directly predict a refined mask. This is more informative since a predicted pixel value of 0.5 is clearly going to contribute to the loss when ground truth is binary but it is not clear in which way the prediction should change to decrease the loss. Again, adversarial attacks generated with PGD are harnessed to simulate domain shifts without prior knowledge of what kind of shifts will be encountered at test-time

In practice, a neural network f_{REF} is trained to remove the mask corruption caused by a synthetically-induced domain shift on image x' using the clean, non-corrupted image x from the training distribution as ground truth. An overview of the training pipeline is in Figure 4.7.

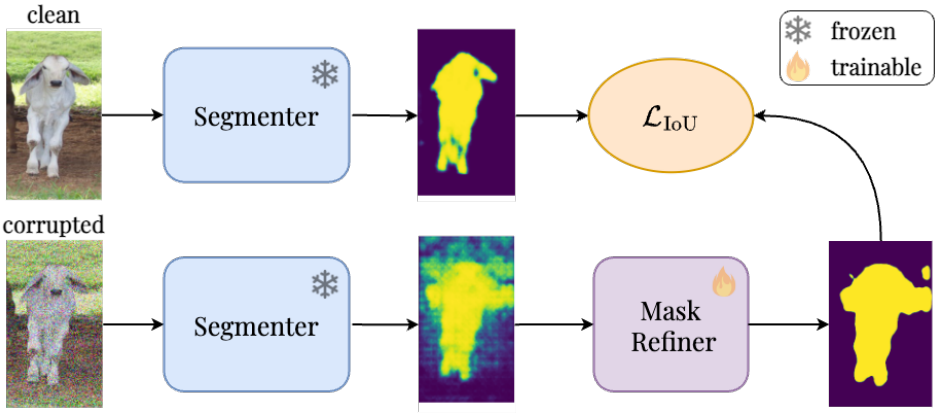


Figure 4.7: Mask refinement module training. The segmentation pre-trained network (segmenter) receives two images as input: A clean image and a synthetically corrupted image. The synthetic corruptions simulate domain shift. The output are two masks, clean and corrupted. The mask refinement module is then trained to predict the the clean mask, given the corrupted mask as input only. No gradients can flow back to the segmenter. The mask refinement module can be interpreted as a model removing corruption noise from masks.

Formally, at test-time, the model parameters are updated to minimize the loss between mask prediction and a refined mask estimated by f_{REF} from $s = d_s^{\varphi} \circ e^{\omega}(x)$

$$\omega^{\star}, \varphi^{\star} = \arg\min_{\omega,\varphi} \mathcal{L}_{\text{IoU}}(f_{\text{REF}}(s), s) \tag{4.11}$$

where e^{ω}, d_r^{ψ} are the encoder and the reconstruction decoder. The mask s is generated again with the new network parameters in each iteration.

In the experiment section, the refinement network is also explored as a simple post-processing step without updating the network parameters, as opposed to TTA. The ap-

proach is different from simply training the segmentation network on augmented images as we do not update the segmentation network parameters during the refinement module training. We hypothesize that this approach, together with not providing the refinement module with the original image, only the mask, as input, makes the network more robust to domain-shifts, but the hypothesis should be properly tested in future work.

4.3.6 Adversarial Transformation (ADV)

This method is an extension of TIPI (Test-Time Adaptation with Transformation Invariance) by [20] to image segmentation. The main idea is to make the network invariant to adversarial transformation as a representative of domain shift.

The optimization loss is computed as the reverse KL divergence loss between the model prediction $\mathrm{s}' = d_s^{\varphi} \circ e^{\omega}(x')$ where x' is an adversarially transformed image and the prediction on the original input $s = d_s^{\varphi} \circ e^{\omega}(x)$.

$$\omega_n^{\star}, \varphi_n^{\star} = \arg\min_{\omega_n, \varphi_n} \mathcal{L}_{\mathrm{KL}}(\mathrm{s}_i, \mathrm{s}_i') \tag{4.12}$$

where s_i' is the adversarially transformed prediction and KL is the Kullback-Leibler divergence loss defined as

$$\mathcal{L}_{\mathrm{KL}}(\mathrm{p}, \mathrm{q}) = \frac{1}{N} \sum_{i=1}^{N} \mathrm{q}_i \cdot \log(\frac{\mathrm{q}_i}{\mathrm{p}_i}) \tag{4.13}$$

In forward KL, p corresponds to the model prediction while q to the ground truth. Please note that, as suggested in [20], the reverse KL is used in the proposed method where the input arguments to the function are switched, compared to forward KL. Another important implementation detail is that the gradients should not flow through s_i' - the tensor needs to be detached before the loss computation.

The same adversarial attacks in terms of the ground truth as for the IoU estimation and mask refinement methods are used to generate x' but the computational complexity is reduced by using the Fast Gradient Sign Attack (FGSM) proposed in [60] instead of instead of the iterative Projected Gradient Descent (PGD). However, since the projection only consists in restricting the output to a valid range for an image, typically implemented by simply clipping the output, it is also often referred to as iterative FGSM.

Chapter 5

Experiments

5.1 Language, frameworks

All the code is developed in Python and the PyTorch [68] framework. The Timm [69] library is also used for existing model architecture implementation, such as the U-Net. Experiment tracking and hyper-parameter sweeps were implemented with Weights and Biases [70]. Other notable dependencies are OpenCV [71] and SciPy [72] for the corruption function implementation, Matplotlib [73] for the generation of plots and Pandas [74] for the generation of tables. Further environment information can be found alongside the code. Link to code repository: `https://github.com/klarajanouskova/TTA-SEG`

5.2 Architecture

Segmentation models In all experiments, both with Class-Agnostic Semantic Segmentation Model (CASS) and the SegmentAnything Model (SAM), the encoder is a ViT-B. ViT-B is the smallest pre-trained model available for both MAE and the SegmentAnything Model (SAM). The smaller model allows for faster experimentation with less GPU requirements in exchange for only a minor decrease in performance. The encoder consists of 12 transformer blocks with the dimension of the embedding set to 768, while the decoder consists of 8 transformer blocks with the embedding dimension set to 512. If ViT-L or ViT-H were chosen as encoders instead, the encoder blocks would have been 24 and 32 with embedding dimensions of 1024 and 1280, respectively.

REF and dIoU modules The architecture for the deep IoU surrogate module is inspired by [75]. It consists of 5 convolutional layers connected by the ReLU [76] activation function followed by two fully connected layers. The standard $\mathcal{L}_1$ loss function is used. The mask refinement module architecture is a Timm U-Net [58] with an EfficentNet-B0 [77] backbone pretrained on ImageNet.

5.3 Evaluation Metrics

The evaluation metric is the standard Intersection over Union (IoU), averaged over all masks in the dataset. No distinction between masks of different classes is being made.

Apart from the IoU, multiple variations of difference in error are introduced to analyze the TTA results. The values are computed from the IoU on clean, non-corrupted images, $\mathrm{IoU_c}$, the initial IoU before adaptation on corrupted images, $\mathrm{IoU_i}$, and the IoU after adaptation with optimal learning rate and optimal number of iterations found in hyper-parameter search, $\mathrm{IoU_f}$. The metrics are defined as follows

- the absolute difference between the initial IoU (before adaptation) and the final IoU (after adaptation)

$$\mathrm{diff_{abs}} = \mathrm{IoU_f} - \mathrm{IoU_i} \tag{5.1}$$

- the ratio of the absolute difference in IoU before and after adaptation to the difference between the initial IoU on corrupted images (before adaptation) and the IoU on non-corrupted images (also before adaptation), which shows by how much the loss in performance due to corruption was reduced by adaptation

$$\mathrm{diff_{clean}} = 100 \cdot \frac{\mathrm{IoU_f} - \mathrm{IoU_i}}{\mathrm{IoU_c} - \mathrm{IoU_i}} \tag{5.2}$$

  This value can, and often is, greater than 100. That means the adapted performance on the corrupted images is higher than non-adapted on clean images.

- the ratio of the absolute difference in IoU between the initial (before adaptation) and final (after adaptation) IoU to the total IoU error, which shows how much the total segmentation error was reduced by adaptation

$$\mathrm{diff_{total}} = 100 \cdot \frac{\mathrm{IoU_f} - \mathrm{IoU_i}}{100 - \mathrm{IoU_i}} \tag{5.3}$$

5.4 Class Agnostic Semantic Segmentation

Class-Agnostic Semantic Segmentation Model (CASS) The CASS is jointly trained for segmentation and reconstruction in the same way as the saliency model described in Appendix A.

The training dataset is the training split of VOC as described in Chapter 3. Only the first 40 batches of the validation set are used for validation to speed up training.

The input size of the encoder is increased from 224×224 to 384×384 while keeping the patch size of 16×16 so that the pretrained patch embedding layer parameters can be

kept. This may have an impact on the REC TTA method performance since it was shown in [15] that masking out larger blocks of the image improves performance on downstream tasks. Increasing the input size while keeping the patch size essentially makes the masked blocks smaller but the impact of this decision is not ablated in this work.

The AdamW [78] optimizer with a learning rate of $5e^{-5}$, batch size of 16 and a single linear warmup epoch is used. As opposed to the original finetuning setup for classification from [15], no layer decay is used. The weights of the segmentation and reconstruction losses are set to be equal since no impact of increasing segmentation weight on segmentation performance was observed. The model converges in about 10 epochs. Otherwise, all hyper-parameters are kept the same as in [15].

Only the scenario of training on top of the MAE-pretrained parameters is explored and an ablation study on how the large-scale reconstruction pertaining influences the performance of reconstruction-based TTA compared to joint training from scratch could be explored in future work. As shown in Appendix E, the optimization when part or all of the weights are re-initialized converges much slower and completely different hyperparameters would likely be needed. Training and validation curves from training the model can also be found in Appendix E.

IoU Estimation and Mask Refinement Modules The IoU estimation and mask refinement modules are trained on the same training set as the segmentation network, the training set of the VOC dataset. These modules are only trained after the training process of the segmentation network is finished. There are many alternatives to this approach which are not explored in this work.

The number of iterations i for the adversarial attack simulating corruption is picked randomly from $i \in [0, \cdots 10]$ with a probability of $1/(i{+}2)$ to decrease computational costs. The inverted and the random ground truth options are sampled with equal probability. The learning rate for the inversion attack is set to 0.001 and the learning rate for the random attack to 0.005. The permitted pixel intensity change is set to $\epsilon = 0.05$.

The models are trained for 25 epochs but converge faster.

Results of a small experiment exploring the mask refinement module as a postprocessing method, rather than a TTA method, are reported in Table 5.1. Training with true corruptions, as opposed to the adversarial ones, is also compared. When evaluated on corrupted images, the results of both corruption and adversarial training are comparable. When evaluated on clean images, the adversarial training greatly outperforms the corruption-based one, improving the IoU by almost 4 % while the refinement module trained on corruptions deteriorates the performance. The results are promising and this application of the mask-refinement module could be explored in future work,

Clean images			Corrupted images		
Refinement training			Refinement training		
adversarial	corruptions	IoU	adversarial	corruptions	IoU
-	-	85.47	-	-	76.04
✗	✓	84.68	✗	✓	76.49
✓	✗	88.44	✓	✗	76.48

Table 5.1: Mask-refinement module as a post-processing step. The output of the refinement module on the segmentation mask is evaluated, without updating the parameters of the segmentation network. The first line corresponds to the non-refined segmentation. The results show that on corrupted images, the adversarially trained mask-refinement network is only marginally worse than the refinement network trained on the same corruptions as the test images. Further, it can be seen that the adversarially-trained refinement network improves the performance on clean images by 3 %, while the corruption-trained refinement network decreases the performance on clean images. The results for corrupted images are averaged across all corruptions kinds, 3 severity levels for each.

5.4.1 Test-Time Adaptation

Evaluation datasets The VOC_{20}-C and VOC_{120}-C datasets are used for evaluation. The evaluation is performed on 10 different kinds of corruption, 3 levels (level 1, 3, 5) of each. Further, the images are extended by two extra copies of the non-corrupted images to balance the number of samples of each corruption category during evaluation (clean images are considered a corruption kind here). This means that the VOC_{20}-C actually consists of 660 ($20 \times 10 \times 3 + 60$ non-corrupted) different image samples created from the 20 images and PASCAL-C-120 consists of 3960 ($120 \times 10 \times 3 + 360$ non-corrupted) image samples.

Finding Hyper-Parameters The performance of each method is first studied separately on the VOC_{20}-C dataset to find optimal hyper-parameters. The hyper-parameters are the learning rate and the number of TTA iterations. Since there is no stopping rule, methods that do not diverge, that is, the optimal number of iterations is the total number of iterations, are desirable. We perform 10 TTA iterations to keep the computational cost reasonable.

Aggregated results as well as the found hyper-parameters for each method are reported in Table 5.2. It can be seen that the REC method outperforms all the other methods, reducing the IoU error caused by corruption by 12.73 %. The second best method is REF which reduces the error by 11.04 %. Both of these method perform best at the last iteration. The next best performing method is ENT, but the method diverges after 4 iterations. The error reduction at iteration 4 is 6.90 %. The dIoU method again performs best in the last iteration but reduces the error by 5.79 % only. No hyper-parameters

performing well across all coruptions were found for the ADV method. It could be caused by the evaluation setup being very different from that of [20] or by an implementation error.

	IoU		error difference			best setting	
TTA method	initial	best	abs	base (%)	total (%)	it	lr
ENT	77.22	77.89	0.67	6.90	2.92	4	0.005
REC	77.22	78.45	1.23	12.73	5.39	10	0.05
dIoU	77.22	77.78	0.56	5.79	2.45	10	0.0005
REF	77.22	78.29	1.06	11.04	4.67	10	0.001
ADV	77.22	77.22	0.00	0.00	0.00	0	0.1

Table 5.2: Aggregated TTA results across all corruption settings of the Class-Agnostic Semantic Segmentation Model (CASS) on Cityscapes-C. The initial, non-adapted value, the best TTA value, error differences, as well as the optimal learning rate and iteration settings, are reported. No overall positive settings are found for ADV. While ENT yields positive results, the best hyper-parameter settings diverge within the 10 iterations.

Detailed results with per-corruption and per-level performance can be found in Appendix C. When the best reported iteration is 0, it means that no improvement has been achieved and it was better not to adapt. All optimization is done with the SGD optimizer.

A visualization of IoU evolution of each method over TTA iterations for each of the methods is shown in 5.1. Figure 5.2 shows a detailed analysis of per-sample performance of different methods.

Overall, the results reveal that none of the methods performs well on all of the corruptions. The absolute improvements in IoU differ a lot between different corruptions and corruption levels.

Comparison of different methods Taking the best overall hyper-parameter settings found in previous experiments, the results of different methods are compared side-by-side in Table B.6 and best results for each corruption setting are highlighted. It can be seen that the combination of reconstruction, IoU estimation and refinement-based adaptation seems quite complementary. It can also be seen that entropy minimization performs quite well on the intensity-transformation-based corruptions included in the evaluation (brightness and contrast). Two method combinations were explored. The first combination consists of the two best-performing methods REC and REF. The results are reported in B.7 and reveal the corruption error is reduced by 15.47 %, more than by any of the individual methods. In the second option, the dIoU method is added since it is the most complementary to the previous two. The corruption error is reduced by 14.36 %, which is less than by the previous combination but more than by the individual methods. It is possible that a better hyper-parameter setting would further improve the results but

the number of combinations is too high for a hyper-parameter sweep to be practical.

Finally, two oracle options are also explored, referred to as oracle and oracle+. The oracle method assumes it is known which method is optimal for each sample and the oracle+ option assumes both the optimal method and iteration are known for each sample. The results indicate that the results can improve significantly by knowing which method to choose. While the results further improve by incorporating the information about best iteration, the difference is small. The results are reported in Figure 5.3.

Finally, all of the proposed methods, including the method combinations and the oracle options, are compared in Figure 5.4. The results reveal that there is a big gap between the best-performing method and the oracle options.

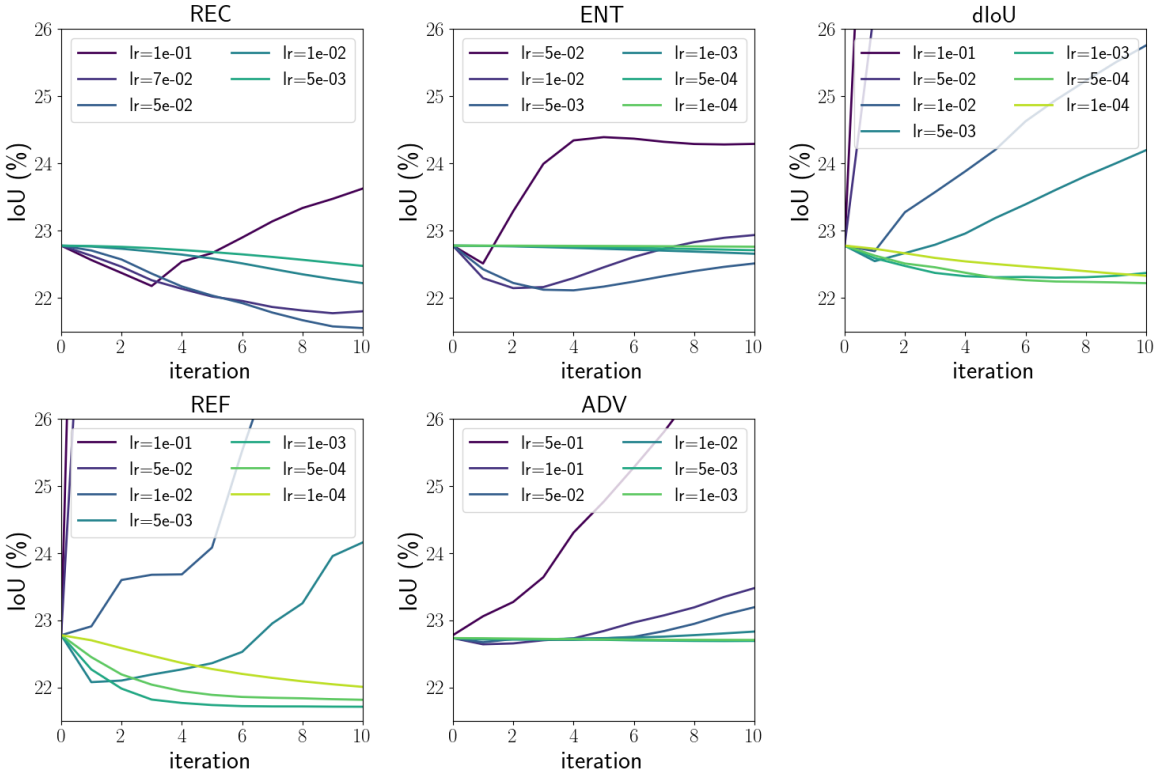


Figure 5.1: IoU evolution over Test-Time Adaptation (TTA) iterations as a function of the learning rate. The results suggest that the Reconstruction-Based Test-Time Training (REC) method would benefit from more iterations. The Deep-Intersection-over-Union-Based Test-Time Training (dIoU) and Mask-Refinement-Based Test-Time Training (REF) methods behave similarly and seem to converge within the 10 iterations. The Entropy-Minimization-Based Test-Time Adaptation (ENT) method converges but rather slowly compared to other methods. If there was an early stopping option, higher learning rate with better performance could be used. The Adversarial-Attack-Based Test-Time Training (ADV) method improves in the first few iterations for some learning rate values but then diverges, smaller learning rate values do not improve the performance.

Effect of freezing/not freezing the decoder for reconstruction Results of ablation study on whether it is better to optimize only the encoder parameters during

REC-based adaptation or both the encoder and the reconstruction decoder are shown in Table 5.3. Optimizing the encoder only clearly outperforms the full optimization. Intuitively, it makes sense that freezing the reconstruction decoder parameters serves as a regularization not to change the representation too much and thus diverge from the segmentation. This is different from the results obtained in [16] on image classification.

	IoU		error difference		
decoder optimized	initial	final	abs	base (%)	total (%)
✗	77.22	78.45	1.23	12.73	5.39
✓	77.22	77.79	0.57	5.88	2.49

Table 5.3: Influence of decoder parameter freezing on Reconstruction-Based Test-Time Training (REC). The results clearly show performance is better when only the encoder parameters are optimized.

Gradient clipping Motivated by the observation that the reconstruction-based optimization sometimes results in very large changes in the output, experiments with gradient clipping during TTA were also performed. The improvements were marginal and for different values for different methods and thus, gradient clipping was not used in other experiments.

Batch size The REC and ENT methods were also evaluated in multi-image setting where the optimization was not performed on a single image but on a batch of images. The experiments were performed on the VOC_{120}-C dataset. The results show the ENT method improves when increasing the batch size while the REC method slightly deteriorates. At 8 images (the largest batch size evaluated), the ENT IoU is 80.17 and the REC IoU is 79.72 %, compared to IoUs of 79.92 % and 80.31 % with the batch size of 1. Other methods were not evaluated in this setting due to time constraints. 5.4

	batch size			
TTA method	1	3	5	8
ENT	79.92	80.10	80.13	**80.17**
REC	**80.31**	79.88	79.81	79.72

Table 5.4: The effect of batch size on intersection over union performance of TTA methods. The results are aggregated over all possible distortion and severity combinations. Evaluated on VOC_{120}-C images from the validation set. The best results for each method are **highlighted**.

Method combination In Table B.6, it was shown that different methods perform best on different corruptions and levels, showing high complementarity of REC, REF and dIoU. Two TTA method combination were explored: The two best-performing methods,

REC + REF and REC + REF + dIoU. The TTA loss in this case is computed as a weighted combination of the individual losses

In Table B.7, the results of the first combination are shown. The TTA learning rate $\mathrm{lr}_{\mathrm{tta}}$ is set to the best REC learning rate value found in previous experiments, $\mathrm{lr}_{\mathrm{tta}} = \mathrm{lr}_{\mathrm{REC}}$. The weights are set as $w_{\mathrm{REC}} = 1$, $w_{\mathrm{REF}} = \frac{\mathrm{lr}_{\mathrm{REF}}}{\mathrm{lr}_{\mathrm{REC}}}$. Multiple $\mathrm{lr}_{\mathrm{REF}}$ values around the optimal value found previously are explored, resulting in $w_{\mathrm{REF}} \in \{0.01, 0.002, 0.001, 0.0002\}$. This results in corruption error reduction by 15.47 %. The best value achieved by individual methods was 12.73 %.

Figure 5.5 provides examples of segmentation prediction evolution over TTA iterations for some of the images where the IoU loss has decreased. The same is shown in Figure 5.6 for images where the loss increased during TTA.

In Table B.8. The weights are set as $w_{\mathrm{REC}} = 1$, $w_{\mathrm{REF}} = \frac{\mathrm{lr}_{\mathrm{REF}}}{\mathrm{lr}_{\mathrm{REC}}}$ and $w_{\mathrm{dIoU}} = \frac{\mathrm{lr}_{\mathrm{dIoU}}}{\mathrm{lr}_{\mathrm{REC}}}$ where $\mathrm{lr}_{\mathrm{REF}}$ and $\mathrm{lr}_{\mathrm{dIoU}}$ are the best learning rate values found in previous experiments. Only a single learning rate configuration was evaluated since the number of sensible configurations grows exponentially with the number of methods. With IoU reduction by 14.36 %, the results are better than those of individual methods but slightly worse than in the previous experiment.

Both of the method combinations and the second one, in particular, may benefit from a thorough hyper-parameter search.

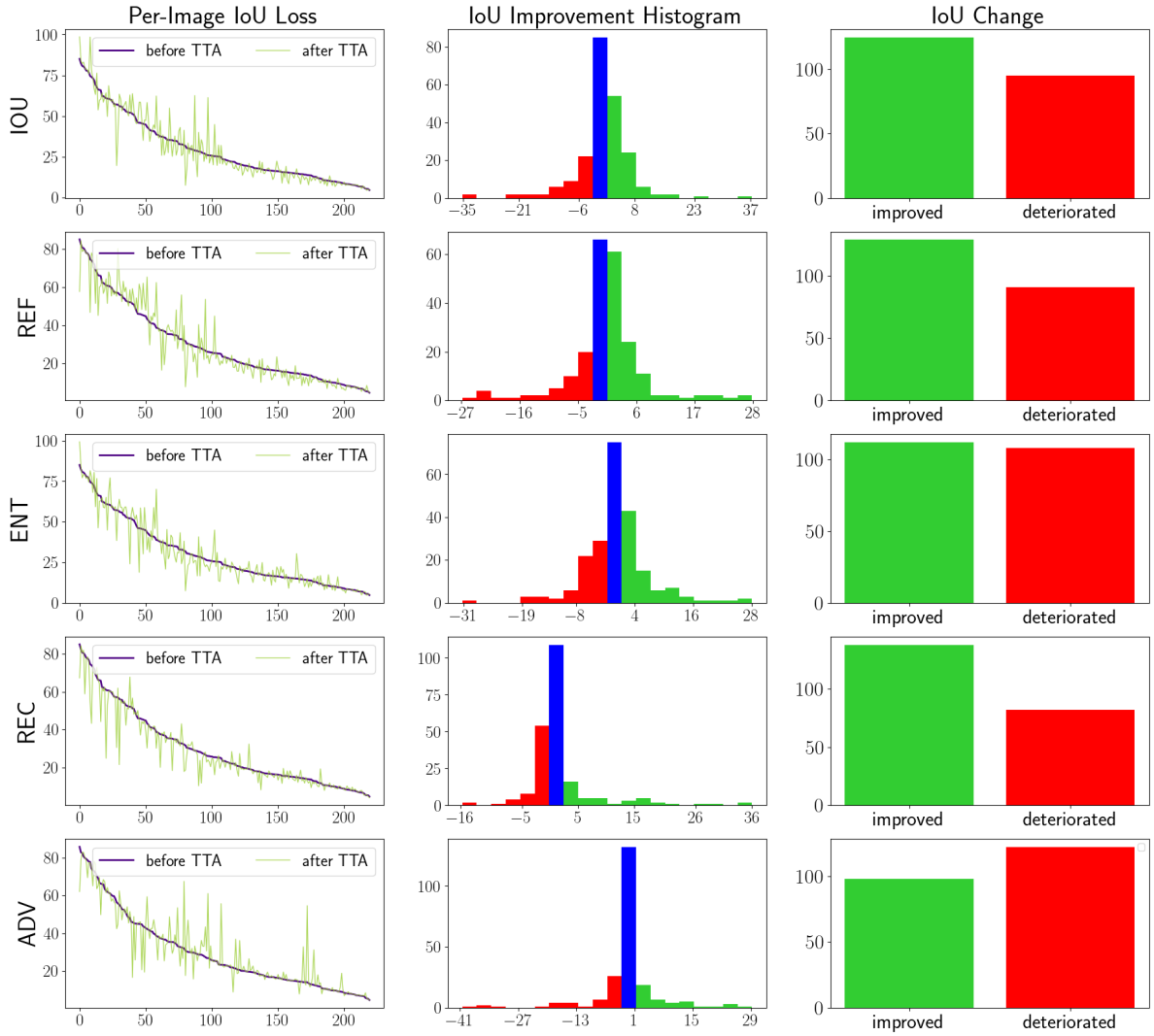


Figure 5.2: Per-image IoU analysis of Test-Time Adaptation (TTA) methods on images with the highest level of corruption applied. Best viewed zoomed in. It can be seen that the Reconstruction-Based Test-Time Training (REC), Deep-Intersection-over-Union-Based Test-Time Training (dIoU) and the Mask-Refinement-Based Test-Time Training (REF) methods increase the loss on some samples but not on those where the loss is low before TTA. This means that when the segmentation quality is high, these methods do not deteriorate it. Other methods do not have this property. The image samples in the plot on the left are sorted according to the loss before TTA, in descending order. The distribution of absolute change in IoU is shown in the middle plot. The number of improved and deteriorated samples for each method is shown on the right. Improved, deteriorated and (close to) unchanged performance is highlighted.

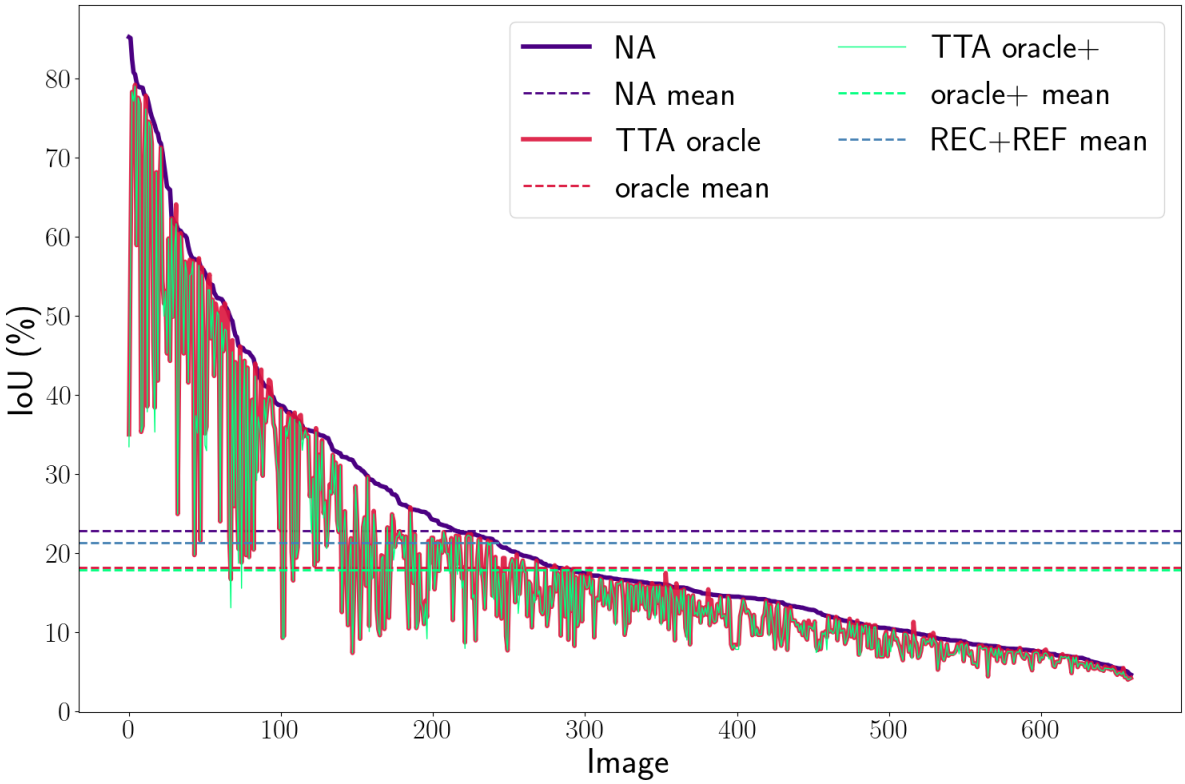


Figure 5.3: TTA oracle and oracle+ results. Oracle means the best Test-Time Adaptation (TTA) method for each image is known. Oracle+ means both the best method and iteration is known. NA refers to non-adapted results. The mean IoU loss of the best-performing method, REC+REF, is also shown. It can be observed that there is still a big gap between the best method combination and the oracle, but there is only a small difference between oracle and oracle+. The images are sorted according to the IoU loss in descending order.

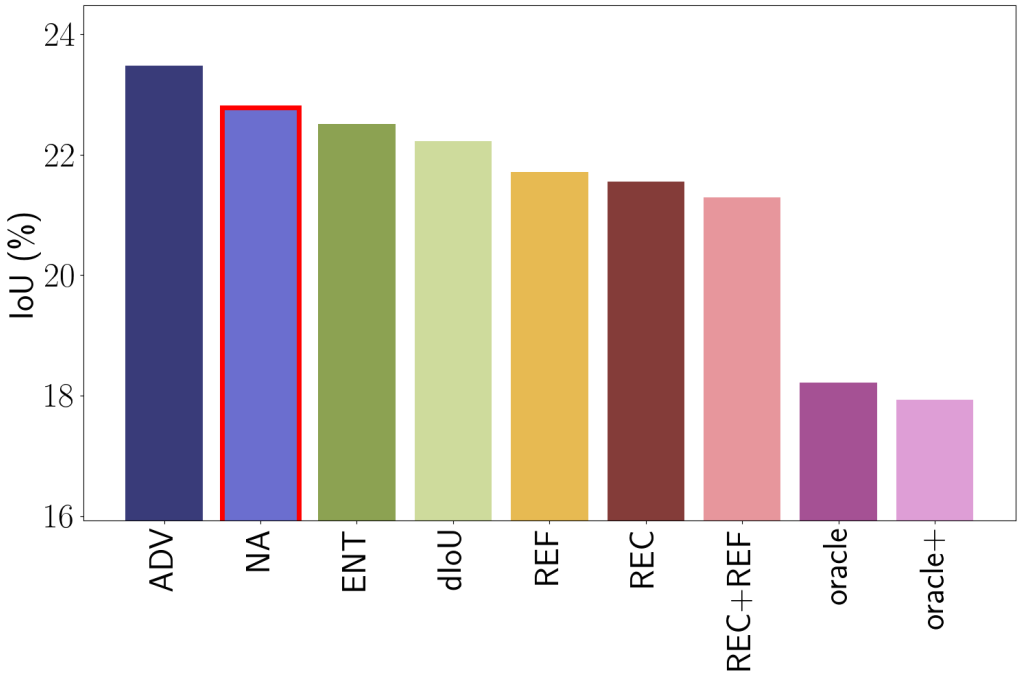


Figure 5.4: Test-Time Adaptation (TTA) method comparison. The highlighted NA method refers to non-adapted results. A big gap between the best-performing method combination found, the REC + REF, and the oracle/oracle+ options where the best method/method and iteration are known can be observed. Note the y axis does not start at 0.

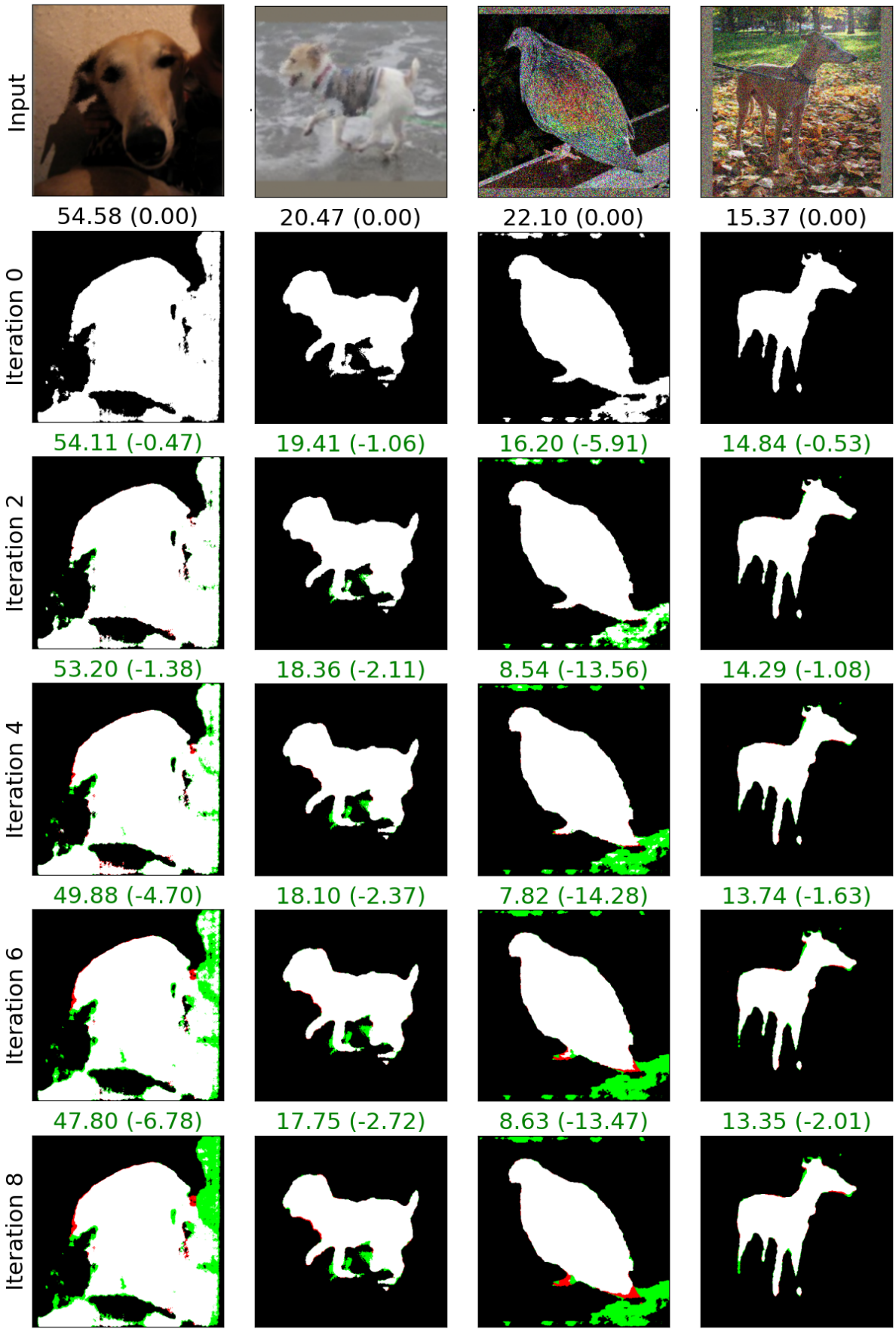


Figure 5.5: Segmentation evolution on images improved by TTA with the best method combination (REC + REF). Pixels where the loss decreased and increased are highlighted. Examples are hand-picked.

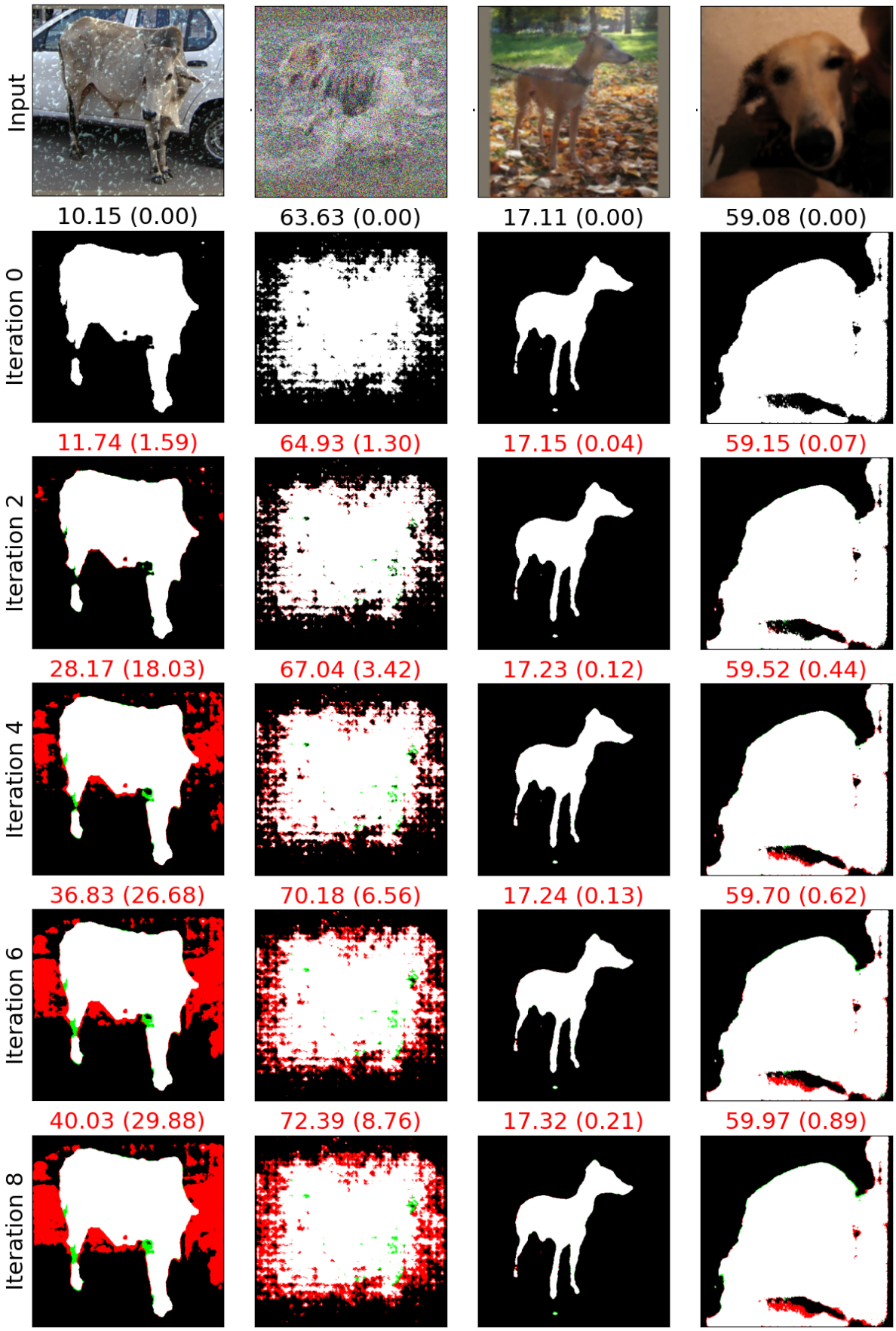


Figure 5.6: Segmentation evolution on images deteriorated by TTA with the best method combination (REC + REF). Pixels where the loss decreased and increased are highlighted. Examples are hand-picked.

5.5 Point-Guided Instance Segmentation

In this section, the experiments with the SAM model are described. The model was trained on image segmentation by the authors [21]. The network was also trained to output an IoU estimation for each prediction. Since the model was not released with a reconstruction decoder, experiments with the REC-based method which performed the best in previous experiments could not be done. The second best performing method, REF, and the baseline, ENT, are evaluated, together with a method based on the pre-trained IoU estimation released with the model, sIoU.

While the TTA is still done on a single image, the setup is slightly different - since there is a variable number of masks for each image according to the number of instances, the adaptation is done for all instance masks at once and the instance losses are averaged. The maximum number of instances per image is limited to 12 to fit on a single 40 GB NVIDIA A100 GPU.

Evaluation datasets All of the evaluation is based on the validation set of the Cityscapes dataset and its synthetic extensions, Cityscapes - fog and Cityscapes - rain. Following the same setup as in previous experiments, a subset of 20 images selected at random is used to create the Cityscapes_{120}-C dataset.

The experiments with the original Cityscapes, Cityscapes - fog and Cityscapes - rain datasets are performed on the first 500 images of the validation set. The train set is only used for the training of the mask refinement module.

Mask Refinement Module In the experiments with SAM, the mask-refinement module used by the REF method is trained on the cityscapes dataset. Since it was not part of the training dataset, the predicted masks used for training are filtered according to the IoU estimate given as output by the pretrained model. The model is trained only on masks with an estimated IoU greater than 90 %.

The number of iterations i for the adversarial attack simulating corruption is picked randomly from $i \in [0, \cdots 10]$ with a probability of $1/(i+2)$ to decrease computational costs. The inverted and the random ground truth options are sampled with equal probability. The learning rate for both types of attack to 0.001.

The models were trained for 1 epoch.

5.5.1 Test-Time Adaptation

Finding Hyper-Parameters The hyper-parameters are first found for each method on the Cityscapes_{120}-C dataset. A detailed analysis of the results including the TTA performance for each corruption and corruption level separately, can be found in C. Only level 1, 2 and 3 corruptions are evaluated since the corruptions impact the images differently

because of a higher input-resolution and smaller average instance sizes.

The aggregated results of these experiments can be found in Table 5.5 table with aggregated results.

	IoU		error difference			best setting	
TTA method	initial	best	abs	base (%)	total (%)	it	lr
ENT	52.19	54.02	1.83	17.51	3.82	10	0.005
sIoU	52.19	52.19	0.00	0.00	0.00	0	0.001
REF	52.09	54.60	2.52	25.12	5.26	10	0.0001

Table 5.5: Aggregated TTA results of the SegmentAnything Model (SAM) on Cityscapes-C. The initial, non-adapted value, the best TTA value, error differences, as well as optimal learning rate and iteration settings, are reported. Initial results for REF are slightly lower because the number of instances is limited to 12, while all instances were evaluated for the other methods. REF clearly outperforms the other methods. No overall positive set of hyper-parameters was found for sIoU.

Cityscapes fog and rain The ENT and the REF methods with the hyper-parameters found in the previous experiments are further evaluated on the original Cityscapes, Cityscapes - fog and Cityscapes - rain datasets. Neither the ENT or the sIoU methods have yielded positive results. The REF method results in an IoU improvement of 0.99 % on the Cityscapes, 0.55 % on Cityscapes - rain and 1.26 % on Cityscapes - fog. These findings are similar to those in [7] where little improvement or even degradation in performance was observed in the naive, single-image TTA setup on the same dataset. The results are reported in Table5.6.

The results of TTA with the SAM model are consistent with the findings from the CASS model experiments, suggesting these findings could also transfer to different segmentation models and segmentation tasks..

ENT TTA method				
	IoU		error difference	
weather	initial	best	abs	total (%)
clear	66.00	66.00	0.00	0.00
rain	64.05	64.05	0.00	0.00
fog	64.79	64.79	0.00	0.00
sIoU TTA method				
	IoU		error difference	
weather	initial	best	abs	total (%)
clear	66.00	66.00	0.00	0.00
rain	64.05	64.05	0.00	0.00
fog	64.79	64.79	0.00	0.00
REF TTA method				
	IoU		error difference	
weather	initial	best	abs	total (%)
clear	66.00	66.99	0.99	2.92
rain	64.05	64.60	0.55	1.54
fog	64.79	66.04	1.26	3.56

Table 5.6: TTA with SAM interactive segmentation on the Cityscapes dataset [10] and its synthetic variations simulating rainy [55] and foggy [56] weather. The best hyper-parameter settings found on the Cityscapes-C dataset is used. Only the Mask-Refinement-Based Test-Time Training (REF) method achieves positive results.

Chapter 6

Conclusions

Multiple test-time adaptation methods for image segmentation models were proposed with focus on Test-Time Training (TTT). Test-time adaptation improves the robustness of the deep neural network to data shift using a single unlabelled image. No other data available are available and thus can not serve as a regularizer. TTT methods require modifications to the training procedure and have not been applied to image segmentation before. Our experiments show the superiority of TTT methods to Test-Time Adaptation (TTA) approaches.

The methods are evaluated on two different models. The first one is a custom model built on top of a reconstruction pre-trained plain Vision Transformer (ViT) trained on a binary segmentation task on a small dataset. A convolutional-neural-network decoder is proposed for the plain ViT backbone which performs on par with state-of-the-art saliency segmentation models but trains significantly faster. The second model is a recently released model trained on 1 billion segmentation masks, the the SegmentAnything Model (SAM). The best method reduces the segmentation error caused by synthetic corruptions for both models by 15.47 % and 25.12 %. SAM is also evaluated on the Cityscapes dataset and its extensions simulating foggy and rainy weather. While the baseline TTA methods have not improved the model's performance on these datasets, the proposed Mask-Refinement-Based Test-Time Training (REF) method improves the performance by about 1 % on all of these datasets while the commonly used entropy-minimization baseline doesn't lead to any improvements.

Further, the proposed mask-refinement module is evaluated as a post-processing method, as opposed to TTA. The results are promising, showing the adversarial training could be particularly powerful to improve performance even on clean, non-corrupted images.

The code will be released at `https://github.com/klarajanouskova/TTA-SEG`

Chapter 7

Limitations and Future Work

The thesis is one of the first works on the topic of TTA for segmentation. As a result, there are many obvious experiments to be done to gain more understanding and possibly to improve the proposed methods further. For instance, different optimizers and optimizer settings could be evaluated.

The methods could also be evaluated in a multi-image or continual setup, as opposed to a single image at a time. Some of the most recent methods proposed for segmentation TTA were not implemented and since the setup is very different (the methods assume the full dataset is available at once), the results can not be compared and it is not clear if these methods would work in the single-image setup.

More realistic benchmarks could be added to further support the thesis findings such as synthetic-to-real domain shift or Cityscapes-to-ACDC to complement the synthetic-corruption-based study.

Some improvements specific to the different methods could have been explored - the influence of varying Reconstruction-Based Test-Time Training (REC) mask-ratio could be explored. For the Mask-Refinement-Based Test-Time Training (REF) and Deep-Intersection-over-Union-Based Test-Time Training (dIoU) methods, similar approaches such as training the modules jointly with the segmentation task (allowing backpropagation of gradients to the segmentation network) could be compared.

More ablations studying the impact of optimizing different parameter subsets such as normalization layers only or the full model should also be performed.

The method that clearly outperformed all other methods is based on optimizing a reconstruction task at test time. While self-supervised test-time training was shown to work if the task gradients are correlated and intuitively, we can imagine why this would work, there is nothing in the training process enforcing such a thing. The idea of regularizing the gradients to increase correlation has been explored for example in meta [79] and continual [80] learning and these methods could improve the performance of the

REC method.

Most importantly, the computational costs of TTA methods are neglected in this work. While the methods that perform optimization of all the parameters are much slower and require more GPU memory than the methods that only optimize a subset of the parameters such as those of normalization layers. Recently, there has been a lot of work on parameter-efficient finetuning of large models that could be explored to improve the efficiency of TTA methods such as LoRA (Low-Rank Adaptation of Large Language Models) [81].

Appendix A

Finetuning MAE

A.1 Reconstruction Only Finetuning

Since there was no implementation of segmentation with masked autoencoders available, the goal of the first experiment is to validate our architecture choice, as well as to gain insights into how to fine-tune the pretrained model. While not related to test-time adaptation, these experiments justify many of the experiment setup choices. All the hyperparameters not discussed here are kept the same as in [15].

As a first step, different hyperparameters were explored to understand how to finetune the pretrained MAE model on a new dataset for reconstruction only. The insights from these experiments are that a very small learning rate (smaller than $1e^{-5}$) is required to avoid overfitting. Experiments with larger batch sizes via gradient accumulation still required small learning rate for healthy training behaviour. Also, removing layer decay and learning rate scheduling (except for linear warmup of 1 epoch) worked best in our experiments. It is possible that if more time was dedicated to exploring different settings, these would work better than our basic optimization approach. A learning rate warmup during which the learning rate is linearly increased for one epoch is used. Loss curves from these hyper-parameter sweeps of these early experiments can be found in a WandB report notebook enclosed in Appendix C.

A.2 Saliency detection

In order to compare to existing models on a task close to the class-agnostic segmentation, the model is first trained on the saliency detection task. To train the saliency model jointly with reconstruction, the learning rate was set to $1e^{-5}$. The weight of the loss terms for both tasks is the same as the segmentation performance was not influenced by an increased segmentation loss weight. Furthermore, it was observed that training

segmentation only (reconstruction weight set to 0) doesn't improve the performance of the model. This result hints at the complementarity of the two tasks.

The results of this model on 5 standard benchmarks (OMRON [48], ECSSD [49], HKU-IS [50], PASCAL-S [51] and SOD [52]), compared to state-of-the-art transformer-based models (Pvt v2 and Swin Transformer [40], [82] of [46]), can be found in Table A.1. The proposed hybrid model based on a pre-trained ViT MAE and with a CNN segmentation decoder performs on par with the state-of-the-art transformer-based saliency detection models. This result was achieved through binarization of the segmentation by thresholding. When evaluating on the raw output of the network, the performance of the model was worse compared to other methods and it was observed that the outputs of our model is not as close to binary as the models of [46]. We hypothesize this may be because the integrity constraints used by [46] were left out in our implementation.

It is worth pointing out that the model has achieved comparable performance in only about 1/5 of the training epochs compared to [46], which is likely due to the MAE pretraining.

method	Dataset																			
	DUT-OMRON				ECSSD				HKU-IS				PASCAL-S				SOD			
	$S_m \uparrow$	$E_m \uparrow$	$F_w \uparrow$	$M \downarrow$	$S_m \uparrow$	$E_m \uparrow$	$F_w \uparrow$	$M \downarrow$	$S_m \uparrow$	$E_m \uparrow$	$F_w \uparrow$	$M \downarrow$	$S_m \uparrow$	$E_m \uparrow$	$F_w \uparrow$	$M \downarrow$	$S_m \uparrow$	$E_m \uparrow$	$F_w \uparrow$	$M \downarrow$
Ours-raw	0.713	0.662	0.37	0.204	0.829	0.733	0.54	0.173	0.811	0.732	0.498	0.168	0.764	0.681	0.486	0.205	0.761	0.667	0.506	0.207
Ours-0.1	0.85	0.895	0.799	0.048	0.931	0.967	0.943	0.024	0.924	0.968	0.931	0.022	0.867	0.916	0.851	0.054	0.802	0.841	0.793	0.089
ICON-S	0.869	0.900	0.804	0.043	0.941	0.966	0.936	0.023	0.935	0.968	0.925	0.022	0.885	0.924	0.854	0.048	0.825	0.856	0.802	0.083
ICON-P	0.865	0.896	0.793	0.047	0.940	0.964	0.933	0.024	0.935	0.967	0.925	0.022	0.882	0.921	0.847	0.051	0.832	0.864	0.813	0.078

Table A.1: Evaluation of model with the proposed decoder architecture saliency datasets. ICON-S refers to the Swin Transformer [40] and ICON-P refers to the Pvt v2 [82] models from [46]. $S_m \uparrow$, $E_m \uparrow$, $F_w \uparrow$, $M \downarrow$ are the S-measure, E-measure, weighted F-measure and the mean average error, as defined in [46]. The signs $\uparrow/\downarrow$ mean higher/lower value of the metric is better. The Ours-raw method corresponds to the evaluation of the raw, non-binarized output of the model while Ours-0.2 corresponds to binarization by thresholding with threshold set to 0.1. It can be seen that our model is on-par with state-of-the-art transformer-based methods.

Appendix B

TTA Hyper-parameters

The performance of each method is studied separately in the single-image setup. That is, batch size of 1 and no continual learning. A good starting learning rate is found for each method and a hyper-parameter sweep is run in the neighbourhood of the initial one. Further learning rate values were added if the first experiments suggested more values should be explored. For this reason, not all methods were tested on the same number of learning rate values. All optimization is done with the SGD optimizer. AdamW seemed to diverge easily in our early experiments and was not used for further analysis. The number of iterations is capped to 10, as more than 10 iterations on a single image is very slow. Methods that do not diverge and tend to have best results at the last iteration are favoured since there is no stopping rule or a way to determine a good number of iterations. If a reliable stopping rule is developed, one would strive for methods that achieve the best possible performance as quickly as possible.

B.1 Class-Agnostic Semantic Segmentation

First, we evaluate the most common baseline method used for comparison in the test-time adaptation literature based on entropy minimization. The detailed results can be found in Table B.1. The corruption error in the best setting when evaluating overall performance across all corruptions and levels is reduced by 6.9 %, at iteration 4. This suggests we are not able to find good hyperparameters for the method in this setup where the segmentation loss would not diverge. We can also observe that the best settings vary greatly for different corruption settings. For the highest level of shot noise and brightness, the method always deteriorates the segmentation. Generally, the performance seems better for lower levels of corruption.

Results of test-time adaptation with reconstruction are shown in Table B.2. The corruption error was reduced by 11.86 %, significantly more than the entropy minimization

baseline. Further, the method doesn't diverge within the 10 iterations in the overall settings and good hyperparameter settings were found for each corruption type and level. However, many corruption settings when performance would improve with an early stopping rule can be found. Higher gains can be observed for noise and blur corruptions, as opposed to weather or intensity-transformation-based ones.

Table B.3 shows the results of the deep IoU estimation method. The overall improvement here is comparable to that of entropy minimization, the corruption error is reduced by 6.54 %. Notably, the segmentation loss doesn't diverge in the majority of the settings. However, we can see that the optimal learning rates vary greatly, we can observe values as different as 0.05 and 0.0005.

It can be argued that deep mask refinement with a binary segmentation model doesn't make much sense, we could as well just apply it as a post-processing step. This is not the case for more complicated segmentation tasks where features from different classes may interact and thus it is worth gaining insight into the method even in this simpler setup. The results can be found in Table B.4. The method achieves a reduction in corruption error of 11.33 % overall, which is comparable to the performance of reconstruction. While the overall best results are again achieved in the last iteration, the numbers again vary quite a bit for different corruption settings. The best learning rate value, on the other hand, is very stable compared to the other methods.

Finally, in Table B.5, the results of the adversarial-transformation-based optimization are reported. While the method achieves good results in many of the corruption settings, no hyperparameters that would perform well over the set of all corruptions were found. It is important to note that the adversarial attacks on segmentation models may be done in many different ways and our choice of adversarial ground truth may be the bottleneck here. Further, the authors of [20] have evaluated the method on classification in a very different setup (continual learning with single iteration per image).

A side-by-side comparison of the performance of all the methods with their best overall settings is reported in Table B.6.

Promising method combinations, REC + REF and REC + REF + dIoU were also evaluated. The results are shown in Table B.7 and Table B.8, respectively.

corruption		IoU		error difference			best setting	
type	level	initial	best	abs	base (%)	total (%)	it	lr
all	5, 3, 1	77.22	77.89	0.67	6.90	2.92	4	0.005
none	5	86.86	87.36	0.49	-	3.76	2	0.005
frost	5	72.82	73.90	1.08	7.69	3.97	1	0.01
frost	3	74.71	74.81	0.09	0.78	0.37	10	0.001
frost	1	81.48	81.48	0.00	0.00	0.00	0	0.05
fog	5	64.95	65.97	1.03	4.68	2.93	3	0.005
fog	3	71.69	71.92	0.24	1.56	0.84	2	0.005
fog	1	74.82	75.64	0.82	6.82	3.26	4	0.005
gaussian noise	5	58.75	58.89	0.14	0.48	0.33	1	0.005
gaussian noise	3	70.39	72.30	1.91	11.60	6.45	10	0.005
gaussian noise	1	82.62	84.72	2.10	49.54	12.09	2	0.01
shot noise	5	55.45	55.45	0.00	0.00	0.00	0	0.05
shot noise	3	72.12	72.28	0.15	1.05	0.55	3	0.005
shot noise	1	82.93	83.26	0.33	8.44	1.94	1	0.005
spatter	5	77.92	79.43	1.51	16.89	6.84	4	0.005
spatter	3	84.53	84.74	0.21	8.85	1.34	1	0.005
spatter	1	86.25	87.09	0.84	136.65	6.09	4	0.005
defocus blur	5	69.59	72.93	3.34	19.33	10.98	10	0.05
defocus blur	3	76.01	79.82	3.81	35.08	15.87	10	0.01
defocus blur	1	82.93	84.03	1.10	27.97	6.45	3	0.01
glass blur	5	77.85	81.45	3.60	39.94	16.25	7	0.005
glass blur	3	81.02	82.89	1.87	31.98	9.84	7	0.005
glass blur	1	85.63	86.13	0.50	40.53	3.47	1	0.01
gaussian blur	5	67.75	71.33	3.57	18.70	11.08	10	0.05
gaussian blur	3	76.81	80.68	3.86	38.43	16.66	10	0.01
gaussian blur	1	86.11	86.37	0.26	34.40	1.86	1	0.005
brightness	5	83.51	83.52	0.01	0.27	0.06	10	0.001
brightness	3	85.40	86.36	0.97	65.87	6.61	3	0.005
brightness	1	86.74	86.89	0.15	126.06	1.14	3	0.005
contrast	5	56.88	59.15	2.27	7.56	5.26	4	0.005
contrast	3	76.03	76.41	0.38	3.53	1.60	6	0.005
contrast	1	83.97	85.58	1.60	55.48	10.00	3	0.005

Table B.1: Entropy-Minimization-Based Test-Time Adaptation (ENT) results, batch size 1. Sweep over learning rate values of [5e-2, 1e-2, 5e-3, 1e-3, 5e-4, 1e-4].

corruption		IoU		error difference			best setting	
type	level	initial	best	abs	base (%)	total (%)	it	lr
all	5, 3, 1	77.22	78.45	1.23	12.73	5.39	10	0.05
none	5	86.86	86.97	0.11	-	0.83	1	0.05
frost	5	72.82	74.02	1.20	8.56	4.42	9	0.07
frost	3	74.71	75.86	1.14	9.42	4.53	10	0.05
frost	1	81.48	83.76	2.28	42.29	12.29	4	0.1
fog	5	64.95	65.25	0.31	1.40	0.87	10	0.07
fog	3	71.69	72.24	0.56	3.66	1.96	10	0.1
fog	1	74.82	76.02	1.20	9.96	4.76	10	0.07
gaussian noise	5	58.75	61.42	2.67	9.49	6.47	10	0.1
gaussian noise	3	70.39	76.71	6.33	38.40	21.37	10	0.1
gaussian noise	1	82.62	83.34	0.72	16.87	4.12	4	0.05
shot noise	5	55.45	65.10	9.65	30.72	21.66	10	0.05
shot noise	3	72.12	78.67	6.54	44.39	23.47	8	0.1
shot noise	1	82.93	83.56	0.63	15.97	3.68	2	0.1
spatter	5	77.92	81.15	3.22	36.04	14.60	8	0.05
spatter	3	84.53	85.32	0.79	33.94	5.12	10	0.1
spatter	1	86.25	87.10	0.85	139.38	6.21	7	0.05
defocus blur	5	69.59	71.99	2.40	13.88	7.88	10	0.1
defocus blur	3	76.01	78.56	2.55	23.48	10.62	10	0.1
defocus blur	1	82.93	85.74	2.82	71.59	16.51	7	0.05
glass blur	5	77.85	80.52	2.67	29.64	12.06	10	0.07
glass blur	3	81.02	84.31	3.28	56.22	17.30	6	0.1
glass blur	1	85.63	86.04	0.41	32.91	2.82	7	0.1
gaussian blur	5	67.75	68.93	1.18	6.17	3.66	8	0.1
gaussian blur	3	76.81	79.86	3.05	30.35	13.15	10	0.05
gaussian blur	1	86.11	86.84	0.73	97.29	5.25	10	0.05
brightness	5	83.51	83.88	0.36	10.83	2.20	6	0.05
brightness	3	85.40	85.67	0.27	18.63	1.87	2	0.1
brightness	1	86.74	87.03	0.28	235.96	2.13	4	0.07
contrast	5	56.88	56.96	0.08	0.25	0.18	8	0.07
contrast	3	76.03	77.11	1.08	9.93	4.49	10	0.1
contrast	1	83.97	83.97	0.00	0.00	0.00	0	0.1

Table B.2: Reconstruction-Based Test-Time Training (REC) results, batch-size 1, no gradient clipping. Sweep over learning rate values of [1e-1, 7e-2, 5e-2, 1e-2, 5e-3].

corruption		IoU		error difference			best setting	
type	level	initial	best	abs	base (%)	total (%)	it	lr
all	5, 3, 1	77.22	77.78	0.56	5.79	2.45	10	0.0005
none	5	86.86	87.52	0.66	-	5.00	2	0.005
frost	5	72.82	73.71	0.89	6.32	3.27	9	0.0001
frost	3	74.71	74.71	0.00	0.00	0.00	0	0.1
frost	1	81.48	81.52	0.03	0.61	0.18	10	0.0005
fog	5	64.95	66.63	1.69	7.69	4.81	10	0.001
fog	3	71.69	72.57	0.88	5.82	3.12	10	0.001
fog	1	74.82	76.64	1.83	15.16	7.25	2	0.005
gaussian noise	5	58.75	58.75	0.00	0.00	0.00	0	0.1
gaussian noise	3	70.39	70.75	0.37	2.23	1.24	2	0.001
gaussian noise	1	82.62	83.58	0.96	22.59	5.51	6	0.0001
shot noise	5	55.45	55.45	0.00	0.00	0.00	0	0.1
shot noise	3	72.12	73.41	1.28	8.71	4.61	10	0.001
shot noise	1	82.93	84.30	1.36	34.71	7.99	6	0.001
spatter	5	77.92	80.21	2.28	25.55	10.35	4	0.001
spatter	3	84.53	86.32	1.79	76.67	11.57	4	0.001
spatter	1	86.25	87.24	0.99	161.67	7.20	4	0.0005
defocus blur	5	69.59	74.40	4.81	27.83	15.81	4	0.01
defocus blur	3	76.01	80.12	4.11	37.88	17.13	10	0.0005
defocus blur	1	82.93	84.02	1.09	27.78	6.41	9	0.0005
glass blur	5	77.85	82.57	4.72	52.34	21.29	3	0.005
glass blur	3	81.02	81.98	0.96	16.40	5.05	1	0.005
glass blur	1	85.63	86.54	0.91	74.00	6.34	1	0.01
gaussian blur	5	67.75	71.96	4.21	22.02	13.05	9	0.001
gaussian blur	3	76.81	81.13	4.32	42.94	18.61	2	0.01
gaussian blur	1	86.11	86.74	0.63	84.02	4.54	1	0.01
brightness	5	83.51	83.51	0.00	0.00	0.00	0	0.1
brightness	3	85.40	86.37	0.97	66.08	6.63	4	0.0005
brightness	1	86.74	86.93	0.19	154.94	1.40	4	0.001
contrast	5	56.88	56.88	0.00	0.00	0.00	0	0.1
contrast	3	76.03	77.37	1.33	12.32	5.57	4	0.005
contrast	1	83.97	84.09	0.11	3.89	0.70	1	0.001

Table B.3: Deep-Intersection-over-Union-Based Test-Time Training (dIoU) results, batch size 1. Sweep over learning rate values of [1e-1, 5e-2, 1e-2, 5e-3, 1e-3, 5e-4, 1e-4] .

corruption		IoU		error difference			best setting	
type	level	initial	best	abs	base (%)	total (%)	it	λ
all	5, 3, 1	77.22	78.29	1.06	11.04	4.67	10	0.001
none	5	86.86	87.68	0.82	-	6.25	6	0.0005
frost	5	72.82	72.82	0.00	0.00	0.00	0	0.1
frost	3	74.71	74.78	0.07	0.55	0.26	2	0.005
frost	1	81.48	84.65	3.17	58.94	17.12	4	0.001
fog	5	64.95	66.46	1.52	6.93	4.33	1	0.005
fog	3	71.69	72.66	0.97	6.42	3.44	10	0.001
fog	1	74.82	75.27	0.45	3.75	1.79	1	0.001
gaussian noise	5	58.75	58.87	0.12	0.44	0.30	4	0.0001
gaussian noise	3	70.39	74.49	4.11	24.94	13.88	7	0.001
gaussian noise	1	82.62	85.06	2.44	57.46	14.02	5	0.001
shot noise	5	55.45	56.97	1.52	4.84	3.41	1	0.005
shot noise	3	72.12	76.24	4.12	27.96	14.78	2	0.005
shot noise	1	82.93	84.69	1.76	44.84	10.32	5	0.001
spatter	5	77.92	78.93	1.00	11.22	4.54	10	0.001
spatter	3	84.53	86.59	2.06	88.14	13.29	6	0.001
spatter	1	86.25	87.47	1.22	199.92	8.89	7	0.0005
defocus blur	5	69.59	72.68	3.09	17.89	10.16	10	0.001
defocus blur	3	76.01	80.35	4.34	40.00	18.09	10	0.001
defocus blur	1	82.93	83.54	0.61	15.58	3.59	10	0.0001
glass blur	5	77.85	81.96	4.10	45.55	18.53	10	0.001
glass blur	3	81.02	82.12	1.10	18.85	5.80	10	0.0005
glass blur	1	85.63	86.20	0.57	46.18	3.96	10	0.0001
gaussian blur	5	67.75	72.45	4.69	24.55	14.55	5	0.01
gaussian blur	3	76.81	81.30	4.49	44.66	19.35	10	0.001
gaussian blur	1	86.11	86.67	0.56	74.55	4.02	1	0.005
brightness	5	83.51	83.57	0.05	1.63	0.33	2	0.0001
brightness	3	85.40	86.70	1.30	88.97	8.93	2	0.001
brightness	1	86.74	86.77	0.02	20.31	0.18	3	0.0001
contrast	5	56.88	56.88	0.00	0.00	0.00	0	0.1
contrast	3	76.03	78.26	2.23	20.59	9.30	7	0.001
contrast	1	83.97	84.72	0.75	25.79	4.65	8	0.0005

Table B.4: Mask-Refinement-Based Test-Time Training (REF) results, batch-size 1, no gradient clipping. Sweep over learning rate values of [5e-4, 1e-4, 5e-5, 1e-5]

corruption		IoU		error difference			best setting	
type	level	initial	best	abs	base (%)	total (%)	it	lr
all	5, 3, 1	77.22	77.22	0.00	0.00	0.00	0	0.1
none	5	86.86	86.86	0.00	-	0.00	0	0.1
frost	5	72.82	73.80	0.98	7.00	3.62	2	0.1
frost	3	74.71	74.71	0.00	0.00	0.00	0	0.1
frost	1	81.48	83.50	2.02	37.53	10.90	7	0.1
fog	5	64.95	64.95	0.00	0.00	0.00	0	0.1
fog	3	71.69	73.08	1.39	9.16	4.91	7	0.1
fog	1	74.82	79.05	4.23	35.11	16.80	10	0.1
gaussian noise	5	58.75	58.75	0.00	0.00	0.00	0	0.1
gaussian noise	3	70.39	70.39	0.00	0.00	0.00	0	0.1
gaussian noise	1	82.62	82.62	0.00	0.00	0.00	0	0.1
shot noise	5	55.45	55.96	0.51	1.62	1.15	3	0.5
shot noise	3	72.12	73.42	1.29	8.77	4.64	2	0.5
shot noise	1	82.93	83.72	0.79	19.99	4.60	9	0.05
spatter	5	77.92	80.86	2.93	32.80	13.28	9	0.1
spatter	3	84.53	86.11	1.58	67.75	10.22	9	0.05
spatter	1	86.25	86.25	0.00	0.00	0.00	0	0.1
defocus blur	5	69.59	70.29	0.70	4.03	2.29	2	0.5
defocus blur	3	76.01	77.67	1.66	15.30	6.92	4	0.5
defocus blur	1	82.93	83.43	0.51	12.88	2.97	3	0.1
glass blur	5	77.85	77.95	0.10	1.12	0.46	6	0.05
glass blur	3	81.02	81.90	0.88	15.04	4.63	7	0.05
glass blur	1	85.63	85.63	0.00	0.00	0.00	0	0.1
gaussian blur	5	67.75	68.10	0.35	1.83	1.08	2	0.5
gaussian blur	3	76.81	76.81	0.00	0.00	0.00	0	0.1
gaussian blur	1	86.11	86.24	0.13	17.29	0.93	10	0.05
brightness	5	83.51	84.08	0.57	16.94	3.44	5	0.5
brightness	3	85.40	86.24	0.84	57.28	5.75	1	0.5
brightness	1	86.74	86.74	0.00	0.00	0.00	0	0.1
contrast	5	56.88	58.11	1.23	4.11	2.86	4	0.1
contrast	3	76.03	78.40	2.37	21.84	9.87	1	0.5
contrast	1	83.97	85.23	1.25	43.41	7.83	3	0.5

Table B.5: Adversarial-Attack-Based Test-Time Training (ADV) results, batch size 1. Sweep over learning rate values of [1e-1, 5e-1, 5e-2, 1e-2, 5e-3, 1e-3]

corruption		IoU					
type	level	NA	REC	dIoU	REF	ENT	ADV
all	3, 4, 5	77.22	**78.36**	77.85	78.31	77.89	77.22
none	5	86.86	86.54	87.35	**87.52**	87.24	86.86
frost	5	72.82	74.14	**75.31**	72.43	72.86	72.82
frost	3	74.71	75.64	**76.39**	75.24	73.61	74.71
frost	1	81.48	82.94	**83.49**	81.47	80.69	81.48
fog	5	64.95	65.04	**66.30**	66.08	65.86	64.95
fog	3	71.69	72.00	**72.61**	72.22	71.69	71.69
fog	1	74.82	76.14	**76.47**	76.32	75.64	74.82
gaussian noise	5	58.75	**61.79**	58.64	56.97	57.64	58.75
gaussian noise	3	70.39	**74.25**	71.21	72.39	71.54	70.39
gaussian noise	1	82.62	82.78	83.29	**85.07**	84.66	82.62
shot noise	5	55.45	**64.38**	57.13	58.52	52.42	55.45
shot noise	3	72.12	**74.80**	71.82	72.40	72.26	72.12
shot noise	1	82.93	83.06	**83.57**	83.02	83.14	82.93
spatter	5	77.92	**80.75**	79.09	80.09	79.43	77.92
spatter	3	84.53	**84.86**	83.93	84.11	84.03	84.53
spatter	1	86.25	86.73	87.19	**87.46**	87.09	86.25
defocus blur	5	69.59	70.89	68.90	**73.78**	71.77	69.59
defocus blur	3	76.01	76.31	75.73	**80.45**	78.13	76.01
defocus blur	1	82.93	**85.32**	83.34	84.07	84.03	82.93
glass blur	5	77.85	79.21	77.36	**82.25**	81.06	77.85
glass blur	3	81.02	**83.52**	81.05	82.44	82.62	81.02
glass blur	1	85.63	85.90	**86.15**	86.08	85.96	85.63
gaussian blur	5	67.75	69.05	67.25	**71.26**	70.14	67.75
gaussian blur	3	76.81	79.17	76.66	**81.52**	79.25	76.81
gaussian blur	1	86.11	**86.88**	86.49	86.32	86.15	86.11
brightness	5	83.51	83.61	**83.64**	82.87	82.77	83.51
brightness	3	85.40	84.51	**86.45**	84.35	86.29	85.40
brightness	1	86.74	86.37	**87.25**	86.30	86.82	86.74
contrast	5	56.88	56.94	58.98	54.65	**59.15**	56.88
contrast	3	76.03	76.17	**77.33**	76.98	76.35	76.03
contrast	1	83.97	83.81	84.05	84.67	**85.49**	83.97

Table B.6: Test-Time Adaptation (TTA) methods comparison. Best overall (not per individual corruptions) setting for each method found in previous experiments was used. NA refers to no adaptation.

corruption		IoU		error difference			best setting	
type	level	initial	best	abs	base (%)	total (%)	it	λ
all	5, 3, 1	77.22	78.71	1.49	15.47	6.55	10	0.001
none	5	86.86	87.41	0.54	-	4.13	2	0.01
frost	5	72.82	73.85	1.03	7.32	3.78	10	0.001
frost	3	74.71	76.14	1.43	11.74	5.64	10	0.0002
frost	1	81.48	83.83	2.35	43.65	12.68	10	0.002
fog	5	64.95	65.43	0.49	2.23	1.39	10	0.0002
fog	3	71.69	71.96	0.28	1.83	0.98	10	0.0002
fog	1	74.82	76.26	1.44	11.95	5.72	10	0.01
gaussian noise	5	58.75	61.08	2.33	8.30	5.65	10	0.0002
gaussian noise	3	70.39	75.05	4.66	28.30	15.75	10	0.001
gaussian noise	1	82.62	84.08	1.46	34.41	8.40	5	0.01
shot noise	5	55.45	64.18	8.73	27.79	19.60	10	0.0002
shot noise	3	72.12	76.88	4.76	32.29	17.07	10	0.001
shot noise	1	82.93	83.35	0.42	10.63	2.45	4	0.001
spatter	5	77.92	81.42	3.49	39.08	15.82	8	0.001
spatter	3	84.53	86.11	1.58	67.90	10.24	10	0.001
spatter	1	86.25	87.29	1.04	169.60	7.55	6	0.01
defocus blur	5	69.59	74.34	4.75	27.51	15.62	10	0.01
defocus blur	3	76.01	80.93	4.92	45.33	20.50	10	0.01
defocus blur	1	82.93	86.11	3.19	81.00	18.68	10	0.002
glass blur	5	77.85	81.81	3.96	43.94	17.87	9	0.01
glass blur	3	81.02	85.49	4.46	76.43	23.52	10	0.001
glass blur	1	85.63	86.62	0.98	79.95	6.85	6	0.001
gaussian blur	5	67.75	71.83	4.08	21.34	12.65	10	0.01
gaussian blur	3	76.81	82.38	5.56	55.38	24.00	10	0.01
gaussian blur	1	86.11	87.43	1.32	176.30	9.52	9	0.002
brightness	5	83.51	83.77	0.26	7.82	1.59	5	0.0002
brightness	3	85.40	85.45	0.05	3.28	0.33	3	0.002
brightness	1	86.74	86.86	0.11	94.99	0.86	1	0.002
contrast	5	56.88	59.31	2.44	8.12	5.65	4	0.01
contrast	3	76.03	78.34	2.31	21.29	9.62	10	0.01
contrast	1	83.97	85.53	1.55	53.78	9.69	7	0.01

Table B.7: Best Test-Time Adaptation (TTA) method combination REC + REF results, batch size 1. The reconstruction weight is 1, λ is the refinement weight. Sweep over λ values of [0.01, 0.002, 0.001, 0.0002].

corruption		IoU		error difference		
type	level	initial	best	abs	base (%)	total (%)
all	5, 3, 1	77.22	78.61	1.39	14.36	6.08
none	-	86.86	87.62	0.76	-	5.76
frost	5	72.82	72.82	0.00	0.00	0.00
frost	3	74.71	75.38	0.67	5.50	2.64
frost	1	81.48	85.07	3.58	66.63	19.35
fog	5	64.95	66.02	1.07	4.89	3.06
fog	3	71.69	72.90	1.21	7.99	4.28
fog	1	74.82	76.25	1.44	11.95	5.72
gaussian noise	5	58.75	58.75	0.00	0.00	0.00
gaussian noise	3	70.39	74.80	4.41	26.76	14.89
gaussian noise	1	82.62	85.14	2.52	59.36	14.48
shot noise	5	55.45	58.75	3.30	10.50	7.41
shot noise	3	72.12	73.30	1.17	7.97	4.21
shot noise	1	82.93	84.67	1.74	44.17	10.17
spatter	5	77.92	79.16	1.24	13.88	5.62
spatter	3	84.53	86.60	2.07	88.86	13.40
spatter	1	86.25	87.41	1.16	189.37	8.43
defocus blur	5	69.59	72.82	3.23	18.69	10.61
defocus blur	3	76.01	81.54	5.53	50.99	23.06
defocus blur	1	82.93	83.36	0.43	11.00	2.54
glass blur	5	77.85	82.64	4.78	53.11	21.61
glass blur	3	81.02	82.06	1.03	17.68	5.44
glass blur	1	85.63	86.02	0.39	31.57	2.71
gaussian blur	5	67.75	71.96	4.21	22.01	13.04
gaussian blur	3	76.81	82.02	5.21	51.81	22.45
gaussian blur	1	86.11	86.67	0.56	74.58	4.03
brightness	5	83.51	83.51	0.00	0.00	0.00
brightness	3	85.40	86.66	1.26	86.03	8.64
brightness	1	86.74	86.76	0.02	13.45	0.12
contrast	5	56.88	57.26	0.38	1.28	0.89
contrast	3	76.03	78.80	2.77	25.58	11.56
contrast	1	83.97	84.47	0.50	17.36	3.13

Table B.8: Test-Time Adaptation (TTA) method combination REC + REF + dIoU results, batch size 1. The weights of REC, REF, dIoU are set to 1, 0.02 and 0.01, respectively.

B.2 Segment Anything

The same hyper-parameter sweep to find good leraning rate value were performed with the TTA methods proposed for the SAM model.

The ENT baseline results are reported in Table B.9. The error caused by corruption domain shifts was reduced by 17.51 % on average. For many corruptions including the aggregated, best results are achieved in the last TTA iteration.

Table B.9 shows that entropy minimization performs well with the SAM model as well, reducing the error caused by corruption by 17.51 %.

Table B.10 reports the results of the sIoU method. While the method works on some of the corruptions, in particular, on those based on intensity transformations, no hyper-parameters performing well across all corruptions were found. For many of the methods, none of the hyper-parameter settings resulted in positive results.

The results of the REF method are reported in Table B.11. The results show a corruption-error reduction of 25.12 % on average. For many corruptions including the aggregated, best results are achieved in the last TTA iteration.

The results are consistent with the findings form the previous experiments where the REF method also outperforms the ENT method.

corruption		IoU		error difference			best setting	
type	level	initial	best	abs	base (%)	total (%)	it	lr
all	3, 2, 1	52.19	54.02	1.83	17.51	3.82	10	0.005
none	-	62.61	62.90	0.28	-	0.75	4	0.005
frost	3	43.12	47.51	4.39	22.54	7.72	9	0.005
frost	2	49.31	51.91	2.60	19.56	5.13	7	0.01
frost	1	56.60	58.36	1.75	29.19	4.04	6	0.005
fog	3	58.55	59.05	0.50	12.24	1.20	4	0.0005
fog	2	58.86	59.06	0.20	5.32	0.48	3	0.01
fog	1	59.48	59.77	0.29	9.21	0.71	6	0.05
gaussian noise	3	9.50	28.99	19.48	36.68	21.53	8	0.01
gaussian noise	2	36.61	42.93	6.31	24.28	9.96	7	0.01
gaussian noise	1	48.50	50.41	1.91	13.56	3.72	10	0.005
shot noise	3	13.02	32.62	19.60	39.52	22.53	7	0.01
shot noise	2	40.49	43.22	2.73	12.35	4.59	10	0.01
shot noise	1	49.97	51.75	1.78	14.07	3.55	7	0.01
spatter	3	50.18	51.28	1.10	8.88	2.22	10	0.001
spatter	2	56.94	57.47	0.52	9.25	1.22	10	0.0005
spatter	1	62.42	62.50	0.08	41.18	0.21	10	0.001
defocus blur	3	41.38	46.82	5.44	25.61	9.28	4	0.05
defocus blur	2	51.46	52.08	0.63	5.64	1.30	10	0.01
defocus blur	1	55.93	56.98	1.05	15.66	2.37	10	0.01
glass blur	3	52.52	52.70	0.17	1.73	0.37	10	0.005
glass blur	2	56.77	56.84	0.07	1.24	0.17	10	0.01
glass blur	1	59.44	59.65	0.20	6.43	0.50	10	0.01
gaussian blur	3	45.88	48.67	2.80	16.71	5.17	3	0.05
gaussian blur	2	53.87	55.29	1.42	16.29	3.09	10	0.01
gaussian blur	1	60.35	60.35	0.00	0.12	0.01	1	0.0005
brightness	3	61.32	61.61	0.29	22.35	0.75	9	0.01
brightness	2	61.90	62.34	0.44	61.53	1.15	9	0.005
brightness	1	62.52	62.76	0.24	252.40	0.64	10	0.005
contrast	3	57.60	58.82	1.22	24.36	2.88	10	0.01
contrast	2	59.52	60.04	0.52	16.72	1.28	9	0.01
contrast	1	60.39	60.75	0.35	15.92	0.89	10	0.01

Table B.9: SAM on Cityscapes: Entropy-Minimization-Based Test-Time Adaptation (ENT). Batch size varies according to the number of instances in an image. Sweep over learning rate values of [5e-2, 1e-2, 5e-3, 1e-3, 5e-4].

corruption		IoU		error difference			best setting	
type	level	initial	best	abs	base (%)	total (%)	it	lr
all	3, 2, 1	52.19	52.19	0.00	0.00	0.00	0	0.001
none	-	62.61	62.65	0.04	-	0.10	6	1e-05
frost	3	43.12	43.12	0.00	0.00	0.00	0	0.001
frost	2	49.31	49.34	0.03	0.19	0.05	1	0.0005
frost	1	56.60	56.60	0.00	0.00	0.00	0	0.001
fog	3	58.55	58.55	0.00	0.00	0.00	0	0.001
fog	2	58.86	58.97	0.11	2.95	0.27	6	5e-05
fog	1	59.48	59.48	0.00	0.02	0.00	2	1e-05
gaussian noise	3	9.50	17.24	7.73	14.56	8.55	7	0.0005
gaussian noise	2	36.61	36.61	0.00	0.00	0.00	0	0.001
gaussian noise	1	48.50	48.50	0.00	0.00	0.00	0	0.001
shot noise	3	13.02	19.44	6.42	12.94	7.38	8	0.0005
shot noise	2	40.49	40.49	0.00	0.00	0.00	0	0.001
shot noise	1	49.97	49.97	0.00	0.00	0.00	0	0.001
spatter	3	50.18	50.22	0.04	0.35	0.09	1	0.0005
spatter	2	56.94	56.94	0.00	0.00	0.00	0	0.001
spatter	1	62.42	62.43	0.01	4.03	0.02	6	1e-05
defocus blur	3	41.38	41.38	0.00	0.00	0.00	0	0.001
defocus blur	2	51.46	51.46	0.00	0.00	0.00	0	0.001
defocus blur	1	55.93	55.93	0.00	0.00	0.00	0	0.001
glass blur	3	52.52	52.52	0.00	0.00	0.00	0	0.001
glass blur	2	56.77	56.77	0.00	0.00	0.00	0	0.001
glass blur	1	59.44	59.44	0.00	0.00	0.00	0	0.001
gaussian blur	3	45.88	45.88	0.00	0.00	0.00	0	0.001
gaussian blur	2	53.87	53.87	0.00	0.00	0.00	0	0.001
gaussian blur	1	60.35	60.35	0.00	0.00	0.00	0	0.001
brightness	3	61.32	61.34	0.02	1.37	0.05	3	0.0005
brightness	2	61.90	62.20	0.29	41.43	0.77	9	5e-05
brightness	1	62.52	62.75	0.23	239.77	0.60	1	0.001
contrast	3	57.60	57.60	0.00	0.00	0.00	0	0.001
contrast	2	59.52	60.15	0.63	20.28	1.55	1	0.001
contrast	1	60.39	61.04	0.65	29.13	1.63	1	0.001

Table B.10: SAM on Cityscapes: SAM-Intersection-over-Union-Based Test-Time Training (sIoU). Batch size varies according to the number of instances in an image. Sweep over learning rate values of [1e-3, 5e-4, 1e-4, 5e-5, 1e-5] .

corruption		IoU		error difference			best setting	
type	level	initial	best	abs	base (%)	total (%)	it	lr
all	3, 2, 1	52.09	54.60	2.52	25.12	5.26	10	0.0001
none	-	62.11	63.45	1.34	-	3.53	4	0.0001
frost	3	43.16	49.32	6.16	32.49	10.83	8	0.0001
frost	2	49.73	52.35	2.62	21.18	5.22	10	0.0001
frost	1	56.45	59.64	3.19	56.29	7.32	10	5e-05
fog	3	58.58	58.84	0.26	7.44	0.64	8	5e-05
fog	2	58.98	59.17	0.19	6.03	0.46	4	0.0001
fog	1	59.48	59.85	0.37	14.15	0.92	8	0.0001
gaussian noise	3	9.47	24.76	15.28	29.03	16.88	10	0.0001
gaussian noise	2	36.86	45.67	8.81	34.90	13.96	10	5e-05
gaussian noise	1	48.86	52.03	3.17	23.95	6.20	8	0.0001
shot noise	3	12.71	28.04	15.33	31.03	17.56	10	0.0001
shot noise	2	40.87	47.63	6.75	31.79	11.42	10	0.0001
shot noise	1	50.01	52.81	2.81	23.19	5.62	10	0.0001
spatter	3	51.24	51.44	0.20	1.85	0.41	3	5e-06
spatter	2	56.89	59.77	2.87	55.06	6.67	5	0.0001
spatter	1	61.77	62.77	1.00	293.91	2.62	3	5e-05
defocus blur	3	41.30	44.48	3.18	15.26	5.41	3	0.0001
defocus blur	2	51.15	52.68	1.53	13.99	3.14	3	0.0001
defocus blur	1	55.42	57.05	1.64	24.45	3.67	10	1e-05
glass blur	3	52.71	55.89	3.18	33.82	6.72	6	0.0001
glass blur	2	56.53	58.36	1.83	32.81	4.21	3	0.0001
glass blur	1	59.22	59.93	0.71	24.55	1.74	10	1e-05
gaussian blur	3	45.97	48.46	2.49	15.45	4.62	4	5e-05
gaussian blur	2	53.40	54.85	1.45	16.63	3.11	6	0.0001
gaussian blur	1	60.21	61.16	0.95	50.08	2.39	6	5e-05
brightness	3	61.12	62.05	0.93	93.99	2.39	10	1e-05
brightness	2	61.79	62.73	0.94	289.32	2.46	4	5e-05
brightness	1	62.13	63.55	1.41	-6455.88	3.73	7	1e-05
contrast	3	57.30	58.62	1.32	27.44	3.09	5	0.0001
contrast	2	59.11	60.11	1.01	33.56	2.47	7	5e-05
contrast	1	60.08	61.34	1.26	62.03	3.16	4	5e-05

Table B.11: SAM on Cityscapes: Mask-Refinement-Based Test-Time Training (REF) results. Batch size varies according to the number of instances in an image. Sweep over learning rate values of [1e-3, 5e-4, 1e-4, 5e-5, 1e-5] .

Appendix C

Attachments

Attachment D	Exploring different architectures and finetuning setups with Masked Autoencoders
Attachment E	Joint finetuning of segmentation and reconstruction - loss curves and insights
Attachment F	Deep Intersection over Union (IoU) surrogate training - loss curves

Bibliography

[1] D. Valevski, M. Kalman, Y. Matias, and Y. Leviathan, "Unitune: Text-driven image editing by fine tuning an image generation model on a single image", *arXiv preprint arXiv:2210.09477*, 2022.

[2] C. Gao, Y. Shih, W.-S. Lai, C.-K. Liang, and J.-B. Huang, "Portrait neural radiance fields from a single image", *arXiv preprint arXiv:2012.05903*, 2020.

[3] A. Lukezic, J. Matas, and M. Kristan, "D3s-a discriminative single shot segmentation tracker", in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 7133–7142.

[4] S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, and M. Bethge, "Improving robustness against common corruptions by covariate shift adaptation", *Advances in Neural Information Processing Systems*, vol. 33, pp. 11 539–11 551, 2020.

[5] Z. Nado, S. Padhy, D Sculley, A. D'Amour, B. Lakshminarayanan, and J. Snoek, "Evaluating prediction-time batch normalization for robustness under covariate shift", *arXiv preprint arXiv:2006.10963*, 2020.

[6] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell, "Tent: Fully test-time adaptation by entropy minimization", *arXiv preprint arXiv:2006.10726*, 2020.

[7] R. Volpi, P. De Jorge, D. Larlus, and G. Csurka, "On the road to online adaptation for semantic image segmentation", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 19 184–19 195.

[8] V. Prabhu, S. Khare, D. Kartik, and J. Hoffman, "Augco: Augmentation consistency-guided self-training for source-free domain adaptive semantic segmentation", *arXiv preprint arXiv:2107.10140*, 2021.

[9] Q. Wang, O. Fink, L. Van Gool, and D. Dai, "Continual test-time domain adaptation", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 7201–7211.

[10] M. Cordts, M. Omran, S. Ramos, *et al.*, "The cityscapes dataset for semantic urban scene understanding", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[11] C. Sakaridis, D. Dai, and L. Van Gool, "Acdc: The adverse conditions dataset with correspondences for semantic driving scene understanding", in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 765–10 775.

[12] N. Karani, E. Erdil, K. Chaitanya, and E. Konukoglu, "Test-time adaptable neural networks for robust medical image segmentation", *Medical Image Analysis*, vol. 68, p. 101 907, 2021.

[13] G. Valvano, A. Leo, and S. A. Tsaftaris, "Re-using adversarial mask discriminators for test-time training under distribution shifts", *arXiv preprint arXiv:2108.11926*, 2021.

[14] E. A. Brempong, S. Kornblith, T. Chen, N. Parmar, M. Minderer, and M. Norouzi, "Denoising pretraining for semantic segmentation", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4175–4186.

[15] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 000–16 009.

[16] Y. Gandelsman, Y. Sun, X. Chen, and A. A. Efros, "Test-time training with masked autoencoders", *arXiv preprint arXiv:2209.07522*, 2022.

[17] Y. Liu, P. Kothari, B. van Delft, B. Bellot-Gurlet, T. Mordan, and A. Alahi, "Ttt++: When does self-supervised test-time training fail or thrive?", *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 808–21 820, 2021.

[18] A. Bartler, A. Bühler, F. Wiewel, M. Döbler, and B. Yang, "Mt3: Meta test-time training for self-supervised test-time adaption", in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2022, pp. 3080–3090.

[19] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations", *Proceedings of the International Conference on Learning Representations*, 2019.

[20] A. T. Nguyen, T. Nguyen-Tang, S.-N. Lim, and P. H. Torr, "Tipi: Test time adaptation with transformation invariance",

[21] A. Kirillov, E. Mintun, N. Ravi, *et al.*, "Segment anything", *arXiv preprint arXiv:2304.02643*, 2023.

[22] G. Csurka, "A comprehensive survey on domain adaptation for visual applications", *Domain adaptation in computer vision applications*, pp. 1–35, 2017.

[23] A. L. Rodriguez and K. Mikolajczyk, "Domain adaptation for object detection via style consistency", *arXiv preprint arXiv:1911.10033*, 2019.

[24] K. Saito, D. Kim, S. Sclaroff, T. Darrell, and K. Saenko, "Semi-supervised domain adaptation via minimax entropy", in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 8050–8058.

[25] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7167–7176.

[26] Y. Zhang, S. Miao, T. Mansi, and R. Liao, "Task driven generative modeling for unsupervised domain adaptation: Application to x-ray image segmentation", in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II*, Springer, 2018, pp. 599–607.

[27] Y. Liu, W. Zhang, and J. Wang, "Source-free domain adaptation for semantic segmentation", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1215–1224.

[28] N. R. Aquino, M. Gutoski, L. T. Hattori, and H. S. Lopes, "The effect of data augmentation on the performance of convolutional neural networks", *Braz. Soc. Comput. Intell*, 2017.

[29] A. Steiner, A. Kolesnikov, X. Zhai, R. Wightman, J. Uszkoreit, and L. Beyer, "How to train your vit? data, augmentation, and regularization in vision transformers", *arXiv preprint arXiv:2106.10270*, 2021.

[30] Y. Sun, X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt, "Test-time training with self-supervision for generalization under distribution shifts", in *International conference on machine learning*, PMLR, 2020, pp. 9229–9248.

[31] D. Chen, D. Wang, T. Darrell, and S. Ebrahimi, "Contrastive test-time adaptation", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 295–305.

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database", in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[33] B. Li, Y. Wang, T. Che, *et al.*, "Rethinking distributional matching based domain adaptation", *arXiv preprint arXiv:2006.13352*, 2020.

[34] I. Misra and L. v. d. Maaten, "Self-supervised learning of pretext-invariant representations", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6707–6717.

[35] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction", in *International Conference on Machine Learning*, PMLR, 2021, pp. 12 310–12 320.

[36] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding", *arXiv preprint arXiv:1810.04805*, 2018.

[37] L. Floridi and M. Chiriatti, "Gpt-3: Its nature, scope, limits, and consequences", *Minds and Machines*, vol. 30, no. 4, pp. 681–694, 2020.

[38] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale", *arXiv preprint arXiv:2010.11929*, 2020.

[39] Y. Li, H. Mao, R. Girshick, and K. He, "Exploring plain vision transformer backbones for object detection", *arXiv preprint arXiv:2203.16527*, 2022.

[40] Z. Liu, Y. Lin, Y. Cao, *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows", in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.

[41] J. Xie, W. Li, X. Zhan, Z. Liu, Y. S. Ong, and C. C. Loy, "Masked frequency modeling for self-supervised visual pre-training", *arXiv preprint arXiv:2206.07706*, 2022.

[42] C. Zhang, C. Zhang, J. Song, J. S. K. Yi, K. Zhang, and I. S. Kweon, "A survey on masked autoencoder for self-supervised learning in vision and beyond", *arXiv preprint arXiv:2208.00173*, 2022.

[43] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection", in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[44] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation", in *2016 fourth international conference on 3D vision (3DV)*, Ieee, 2016, pp. 565–571.

[45] W. Gu, S. Bai, and L. Kong, "A review on 2d instance segmentation based on deep neural networks", *Image and Vision Computing*, p. 104 401, 2022.

[46] M. Zhuge, D.-P. Fan, N. Liu, D. Zhang, D. Xu, and L. Shao, "Salient object detection via integrity learning", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[47] L. Wang, H. Lu, Y. Wang, *et al.*, "Learning to detect salient objects with image-level supervision", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 136–145.

[48] C. Yang, L. Zhang, H. Lu, X. Ruan, and M.-H. Yang, "Saliency detection via graph-based manifold ranking", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 3166–3173.

[49] Q. Yan, L. Xu, J. Shi, and J. Jia, "Hierarchical saliency detection", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 1155–1162.

[50] G. Li and Y. Yu, "Visual saliency based on multiscale deep features", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5455–5463.

[51] Y. Li, X. Hou, C. Koch, J. M. Rehg, and A. L. Yuille, "The secrets of salient object segmentation", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 280–287.

[52] V. Movahedi and J. H. Elder, "Design and perceptual validation of performance measures for salient object segmentation", in *2010 IEEE computer society conference on computer vision and pattern recognition-workshops*, IEEE, 2010, pp. 49–56.

[53] L. Qi, J. Kuen, Y. Wang, *et al.*, "Open world entity segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[54] W. Wang, M. Feiszli, H. Wang, J. Malik, and D. Tran, "Open-world instance segmentation: Exploiting pseudo ground truth from learned pairwise affinity", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4422–4432.

[55] X. Hu, C.-W. Fu, L. Zhu, and P.-A. Heng, "Depth-attentional features for single-image rain removal", in *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, 2019, pp. 8022–8031.

[56] C. Sakaridis, D. Dai, and L. Van Gool, "Semantic foggy scene understanding with synthetic data", *International Journal of Computer Vision*, vol. 126, no. 9, pp. 973–992, 2018. [Online]. Available: `https://doi.org/10.1007/s11263-018-1072-8`.

[57] A. Fournier, D. Fussell, and L. Carpenter, "Computer rendering of stochastic models", *Communications of the ACM*, vol. 25, no. 6, pp. 371–384, 1982.

[58] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation", in *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241.

[59] G. Máttyus, W. Luo, and R. Urtasun, "Deeproadmapper: Extracting road topology from aerial images", in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3438–3446.

[60] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples", *arXiv preprint arXiv:1412.6572*, 2014.

[61] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks", *arXiv preprint arXiv:1706.06083*, 2017.

[62] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world", in *Artificial intelligence safety and security*, Chapman and Hall/CRC, 2018, pp. 99–112.

[63] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization", *Advances in neural information processing systems*, vol. 17, 2004.

[64] D.-H. Lee *et al.*, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks", in *Workshop on challenges in representation learning, ICML*, vol. 3, 2013, p. 896.

[65] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", in *International conference on machine learning*, pmlr, 2015, pp. 448–456.

[66] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization", *arXiv preprint arXiv:1607.06450*, 2016.

[67] Y. Wu and K. He, "Group normalization", in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[68] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library", *Advances in neural information processing systems*, vol. 32, 2019.

[69] R. Wightman, *Pytorch image models*, `https://github.com/rwightman/pytorch-image-models`, 2019. DOI: `10.5281/zenodo.4414861`.

[70] L. Biewald, *Experiment tracking with weights and biases*, Software available from wandb.com, 2020. [Online]. Available: `https://www.wandb.com/`.

[71] G. Bradski, "The opencv library.", *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 25, no. 11, pp. 120–123, 2000.

[72] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "Scipy 1.0: Fundamental algorithms for scientific computing in python", *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.

[73] J. D. Hunter, "Matplotlib: A 2d graphics environment", *Computing in science & engineering*, vol. 9, no. 03, pp. 90–95, 2007.

[74] W. McKinney *et al.*, "Pandas: A foundational python library for data analysis and statistics", *Python for high performance and scientific computing*, vol. 14, no. 9, pp. 1–9, 2011.

[75] Y. Patel, T. Hodaň, and J. Matas, "Learning surrogates via deep embedding", in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX*, Springer, 2020, pp. 205–221.

[76] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines", in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[77] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks", in *International conference on machine learning*, PMLR, 2019, pp. 6105–6114.

[78] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization", *arXiv preprint arXiv:1711.05101*, 2017.

[79] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms", *arXiv preprint arXiv:1803.02999*, 2018.

[80] M. Riemer, I. Cases, R. Ajemian, *et al.*, "Learning to learn without forgetting by maximizing transfer and minimizing interference", *arXiv preprint arXiv:1810.11910*, 2018.

[81] E. J. Hu, Y. Shen, P. Wallis, *et al.*, "Lora: Low-rank adaptation of large language models", *arXiv preprint arXiv:2106.09685*, 2021.

[82] W. Wang, E. Xie, X. Li, *et al.*, "Pvt v2: Improved baselines with pyramid vision transformer", *Computational Visual Media*, vol. 8, no. 3, pp. 415–424, 2022.