Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

Instituto de Sistemas
e Robótica | LISBOA

Semi-supervised Semantic Segmentation of Smoke and Fire from Airborne Images with Generative Adversarial Networks to support Firefighting Actions

Master Thesis

Name of the Study Programme

International Media Informatics

Faculty 4

from

Lisa Kuhlmann

Date:

Berlin, 08.03.2021

1st Supervisor: Prof. Dr. Gefei Zhang

2nd Supervisor: Prof. Dr. Alexandre Bernardino

Acknowledgements

I would like to thank my supervisor Prof. Alexandre Bernardino, who gave me the opportunity to write this thesis about a topic I care about, welcomed me to the ISR and Firefront project and guided me during this work. I am very grateful for his patience, dedication and positivity during this time. Many thanks also to the members of the bi-weekly master thesis meetings, students and professors, for their inputs and advice. These meetings were a very helpful and essential part of this work.

To my supervisor Prof. Gefei Zhang in Berlin, thank you for the advice given in advance of this project and the availability for questions.

Thanks to my friends in Portugal, who made sure I got to know their country and culture in the best way possible, as far as the pandemic situation allowed it. Thank you for your company, for sharing memorable moments and the many attempts of distracting me from work. I will stay with 'saudades de Portugal' for sure.

Special gratefulness to my fellow students and friends from HTW for going through the studies together and the invaluable support and motivation given.

Lisa Kuhlmann - Teilüberwachte semantische Segmentation von Rauch und Feuer anhand von luftgestützten Aufnahmen mit Generative Adversarial Networks zur Unterstützung der Bekämpfung von Waldbränden

Kurzzusammenfassung

Die semantische Segmentierung von Rauch und Feuer kann eine Basis für die Anwendung nützlicher Tools zur Unterstützung der Bekämpfung von Waldbränden sein. Die Menge an annotierten Datensets, die für das Training von Deep Learning Modellen benötigt wird, ist sehr limitiert und das Erstellen aufwendig. Deshalb versucht diese Arbeit, nicht annotierte Bilder von Rauch und Feuer zusammen mit annotierten Bildern in einem teilüberwachten Lernansatz zu nutzen. Dazu wird ein Generative Adversarial Network (GAN) genutzt. Da GANs zur Instabilität während des Trainings neigen, werden verschiedene Einstellungen von Hyperparametern und ein kurzes, anfängliches vollüberwachtes Training getestet. Die erhaltenen Ergebnisse zeigen, verglichen mit einer vollüberwachten Methode als Basis, eine leichte Verbesserung der Genauigkeit der Segmentierungen. Dies weist darauf hin, dass die angewendete Methode eine Basis für weitere Verbesserungen in der Anwendung von teilüberwachtem Lernen für die Segmentierung von Rauch und Feuer sein kann.

Stichworte

Semantische Segmentation, Generative Adversarial Networks, Teilüberwachtes Lernen, Rauch, Feuer

Lisa Kuhlmann - Semi-supervised Semantic Segmentation of Smoke and Fire from Airborne Images with Generative Adversarial Networks to support Firefighting Actions

Abstract

The semantic segmentation of smoke and fire can be a basis for the application of useful tools to support firefighting actions. The amount of annotated training datasets required to train a deep neural network for this task is very scarce and the creation is expensive. This is why this work tries to leverage unlabeled training images used together with annotated images in a semi-supervised learning approach with the application of a Generative Adversarial Network (GAN). As GANs are prone to training instabilities, different hyperparameter settings and a short, fully-supervised pre-training are tested in this thesis when being applied to the segmentation of fire and smoke. The results obtained show a small improvement in segmentation accuracy compared to a fully-supervised baseline. This indicates that this method could be a basis for further improvements using additional unlabeled data to train a semi-supervised segmentation model.

Keywords

Semantic Segmentation, Generative Adversarial Networks, Semi-supervised Learning, Smoke, Fire

Contents

List of Figures

List of Tables

Acronyms

ASPP	Atrous Spatial Pyramid Pooling
BCE	Binary Cross Entropy
CNN	Convolutional Neural Network
CRF	Conditional Random Field
FCD	Fully-connected Discriminator
GAN	Generative Adversarial Network
ISR	Institute of Systems and Robotics Lisbon
mIoU	Mean Intersection over Union
SSL	Semi-supervised Learning
UAV	Unmanned Aerial Vehicle
WGAN	Wasserstein Generative Adversarial Network

Chapter 1

Introduction

1.1 Motivation

The risks of wildfires in Portugal and worldwide have been showing an increasing trend over the past decades. Figure 1.1 visualizes this trend in Portugal, where the decadal average of burned area increased "from under 75.000 ha during the 1980ies, to 100.000 ha in the 1990ies, to over 150.000 ha since 2000", as stated in a report about wildfire management by Beighley and Hyde [BH18].

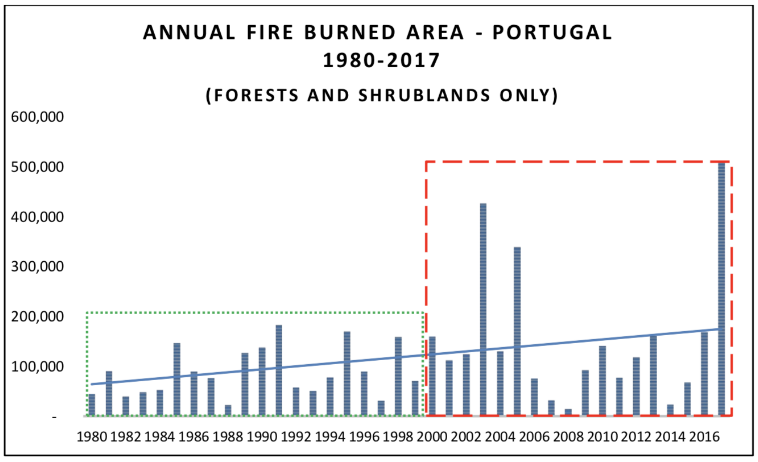
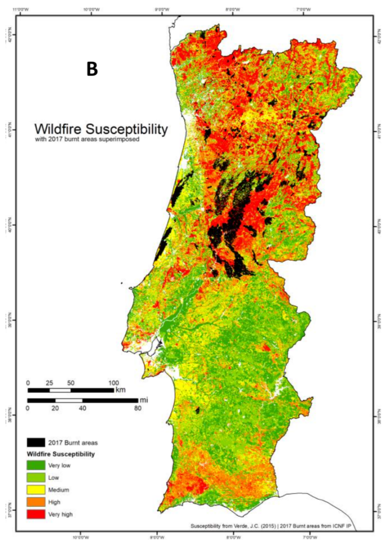
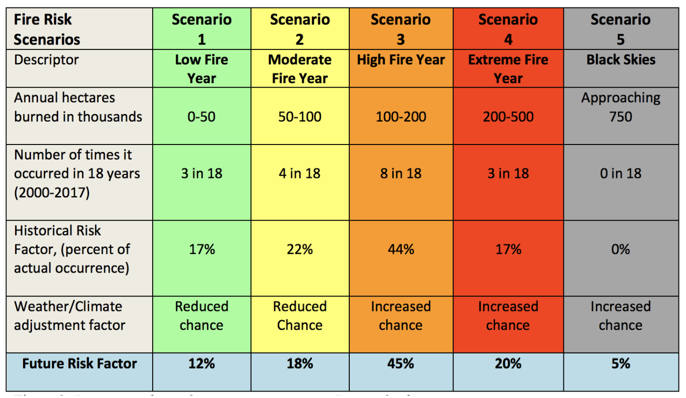


Figure 1.1: The annual burned area by wildfires in hectare in Portugal from 1980 to 2017, adapted from [BH18].

Noticeable is, that there is an inter-annual variability in the severity of the fires, but during the last two decades the intensities of the peak wildfire years have gotten much higher compared to the years before 2000. During these peaks, the risk of firefighting forces being overwhelmed is high. This scenario happened for example during the wildfire period in 2017, where over 100 people lost their lives. Figure 1.2a shows the dimension of the burned areas in 2017 colored in black and in Figure 1.2b Beighley and Hyde report the risks of different fire scenarios happening during the next decade.



(a)



Fire Risk Scenarios	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5
Descriptor	Low Fire Year	Moderate Fire Year	High Fire Year	Extreme Fire Year	Black Skies
Annual hectares burned in thousands	0-50	50-100	100-200	200-500	Approaching 750
Number of times it occurred in 18 years (2000-2017)	3 in 18	4 in 18	8 in 18	3 in 18	0 in 18
Historical Risk Factor, (percent of actual occurrence)	17%	22%	44%	17%	0%
Weather/Climate adjustment factor	Reduced chance	Reduced Chance	Increased chance	Increased chance	Increased chance
Future Risk Factor	12%	18%	45%	20%	5%

(b)

Figure 1.2: Wildfire susceptibility with burned areas in 2017 in 1.2a and different wildfire scenarios and their risk during the next decade in 1.2b. Figures adapted from [BH18].

To handle upcoming wildfire periods, the authors suggest a variety of measures. These include renewing fuel management efforts, fire planning and prevention, improving firefighter performance and their pay and career opportunities, rethinking cultural attitudes as well as adjusting the fire suppression skill set. These measures can also be supported by technology.

This thesis will focus on the aerial support for firefighters with footage from manned and unmanned aerial vehicles, such as drones. Essential for the automatic detection and segmentation of smoke and fire from aerial images is a good quality of the segmentations, as these are the basis for further tasks and need to be reliable. Common problems in this field are the definition of the object borders, the detection of small areas far away from the camera and the confusion with similar objects like fog and clouds.

1.2 Topic overview

The detection and segmentation of smoke and fire has been a widely researched topic in the past, beginning with traditional image processing and machine learning techniques working with handcrafted features such as colors and shapes. A more recent approach is the use of Deep Learning and Convolutional Neural Networks in the field of Computer Vision. These methods have in general shown good results, often outperforming traditional image processing methods because of their power to handle multi-dimensional datasets and to learn combinations of non-linear functions that can approximate very complex data distributions. However, Deep Learning needs a big amount of labeled training data to perform its best. Creating this labeled training data is a time-consuming task. It requires a significant amount of human effort to annotate an image on pixel level for segmentation masks acting as a ground truth reference for network training. Regarding the segmentation of smoke and fire and many other tasks there is a scarcity of labeled training data, which led to different approaches like using additional unlabeled training data in semi-supervised learning. Using only unlabeled data in unsupervised learning or data labeled only on image level or with bounding boxes in weakly-supervised learning are other methods trying to tackle the same problem. This work concentrates on a semi-supervised method using a Generative Adversarial Network (GAN) to leverage unlabeled training images together with labeled images for the task of smoke and fire segmentation.

1.3 Objectives

The goal of this thesis is to study experimentally the potential use of a semi-supervised GAN applied to the segmentation of smoke and fire while only using very small labeled training datasets of rgb channel images with additional unlabeled data. Firstly, the hyperparameter tunability of an existing GAN from the work of Mittal et al. [MTB19] is analysed when being applied to a wildfire dataset published by the Université de Corse

Pasquale Paoli [Noab]. Then the potential benefits of a fully-supervised pre-training before the start of semi-supervised learning are evaluated. Subsequently, the best GAN settings are applied to slightly bigger smoke and fire datasets with newly labeled and unlabeled images gathered from different sources. The results will be compared to fully-supervised models that are only trained with annotated data. In the end, the quality of the segmentation results regarding the designated use in real-time fire fighting actions is evaluated.

Limitations of this work are, that apart from rgb images, there is no additional data like infrared imagery or spatio-temporal information obtained from video frames used to predict the fire and smoke segmentations. Additionally, the use of very small training and testing datasets together with a relatively small number of three validation runs for each experiment should be considered when interpreting the results obtained in this thesis.

1.4 Firefront project

This work was developed at the Institute of Systems and Robotics (ISR) in Lisbon which belongs to a consortium working on a system to support firefighting actions in Portugal, namely the Firefront project [Noac].The project aims to develop a system based on unmanned aerial vehicles (UAVs) and cameras aquiring visible and infrared imagery for real-time detection and tracking of wildfires. This information will, for example, be used in a graphical interface for firefighters that gives a broad and fast overview of the affected areas and helps with planning the next firefighting actions or for predictive models when combined with further information like meteorological data and georeferences. Another important part of the project is the creation of a dataset of airborne sequences and associated telemetry of real forest fire scenarios to support further research.

1.5 Thesis outline

Chapter 2 starts with a theoretical overview about the type of deep learning networks used in this thesis. In Chapter 3, closely related works regarding the application of GANs for semantic segmentation are going to be described in more detail. The methods used for the experiments done are then presented in Chapter 4. The obtained results are presented and analysed in Chapter 5. In the end there is a summary and discussion about the main results and about possible future improvements in Chapter 6.

Chapter 2

Theoretical Background

This chapter will introduce the basic concepts of Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs), as these kinds of networks are the underlying components of the semi-supervised semantic segmentation method applied in this work.

2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of deep neural network that are optimized to work with complex input data such as images or audio. Most semantic segmentation methods using deep learning are based on CNNs and additional modules.

CNNs have the ability to identify the underlying features of an image automatically and in a hierarchical order. This ability distinguishes them from traditional image processing methods that rely on handcrafted features such as colors and shapes that are defined by a human with previously obtained domain knowledge. The definition of these features is very difficult and complex, as they need to be robust to variations, for example occlusions, deformations, lighting conditions, scaling and viewpoints. An example of a feature hierarchy learned by a CNN is shown in Figure 2.1, where the low level features are detected in earlier layers and the high level features in deeper layers of the network.

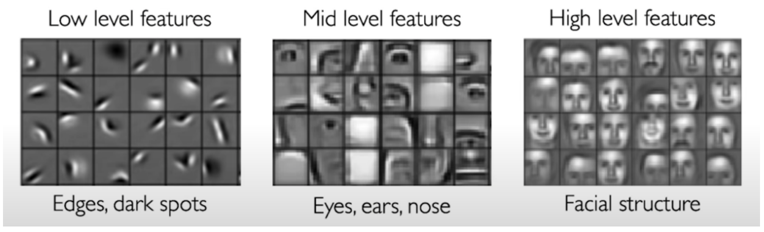


Figure 2.1: An example of hierarchical features extracted by a CNN for human face detection, adapted from [Fea].

An overview of an example architecture of a CNN is depicted in Figure 2.2 for a CNN used for image classification. Characteristic are the subsequent layers of convolution with non-linear processing units, namely activation functions, and pooling. In the case of a classification task, fully-connected layers are added in the end. One of the first CNNs, LeNet5, was already introduced in 1989 by LeCun et al. [Lec98] and consisted of only five layers. Improvements in hardware and network structure have enabled the use of very deep CNNs with more than 100 layers, such as ResNets [He16]. The details of CNN layers are described in the following.

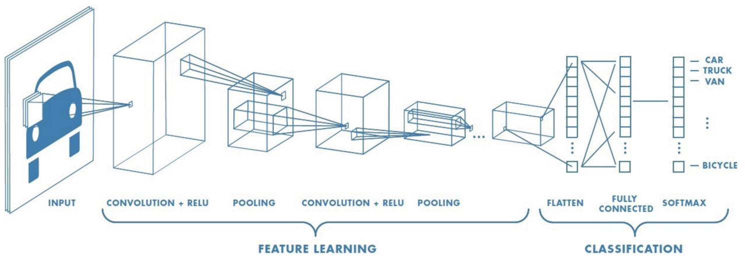


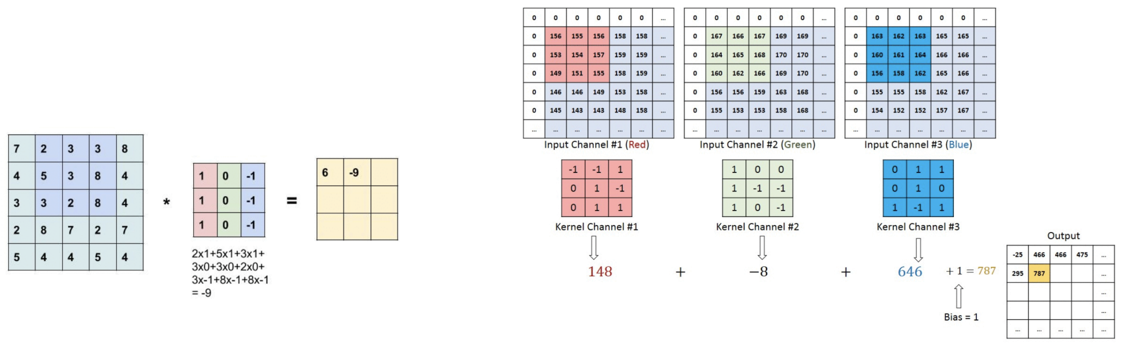
Figure 2.2: An exemplary overview of a CNN used for a classification task, adapted from [Cnn].

2.1.1 Layers

To extract the features of an image, its spatial structure should be considered, because image pixels that are spatially close to each other are more likely to be related to each other. To preserve the spatial structure during deep learning, convolutional layers are applied in CNNs. Convolutional operations contain an element-wise multiplication of the values of an image matrix with the values of a smaller filter kernel matrix, which slides as a patch over the image. The filter matrices can have different values, that are used for example for the task of detecting edges within the patches of an image. During the learning process of a CNN, the network learns what kind of features it needs to extract by applying different kinds of filters and assigning higher values to the important filters. The convolution operation is defined as follows:

$$G_{[\mathrm{m,n}]} = (f * h)[m.n] = \sum_j \sum_k h[j,k] f[m-j, n-k], \tag{2.1}$$

where G is the resulting feature map after the convolution and m and n are its row and column indices. The input image is denoted by f and the filter matrix by h with m and j as the the row and column indices of the filter. Figure 2.3 shows examples for a one-channel convolution and a convolution over multiple input channels, which is the case when using rgb images. Apart from the filter kernel type and size, there are two additional parameters that define a standard convolution operation, which are padding and stride. The stride denotes the distance of two consecutive filter positions while the filter kernel slides over the image and has an influence on the output size of the resulting feature map. Padding adds rows and columns of zeros to the outer sides of an input, so that each element of the real borders of an input can be in the center of a filter kernel. This is used to avoid information loss at the input border regions.



(a) (b)

Figure 2.3: An example of a convolution operation on one channel 2.3a and multiple channels with zero padding 2.3b. Figures adapted from [Cnn].

After applying a convolution, a bias value is added to the resulting value and these values are passed through an activation function. A common activation function used in CNNs is the ReLU function [Aga19] $f(x) = max(0, x)$, which maps negative values to zero but prevents the saturation of gradients in the positive dimension.

Convolutional layers are usually followed by a pooling layer, which applies a downsampling operation on the output feature maps. Downsampling is important to reduce the computational load of processing the data and also helps to extract more prominent features. By doing this, the spatial invariance of the feature maps should be kept, which is why the pooling operation works, similar to the convolution operation, with filters of a defined size and stride that slide over the feature maps. The most common pooling operation is max pooling, where the maximum value of the feature map at the current patch covered by the filter gets extracted. Another possible operation is average pooling, where the average of all values covered by the filter is returned. Figure 2.4 shows an example of both operations mentioned.

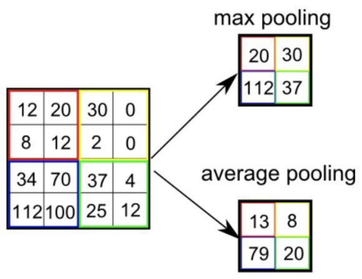
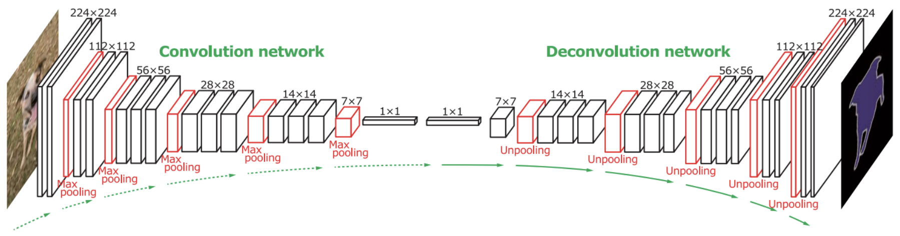


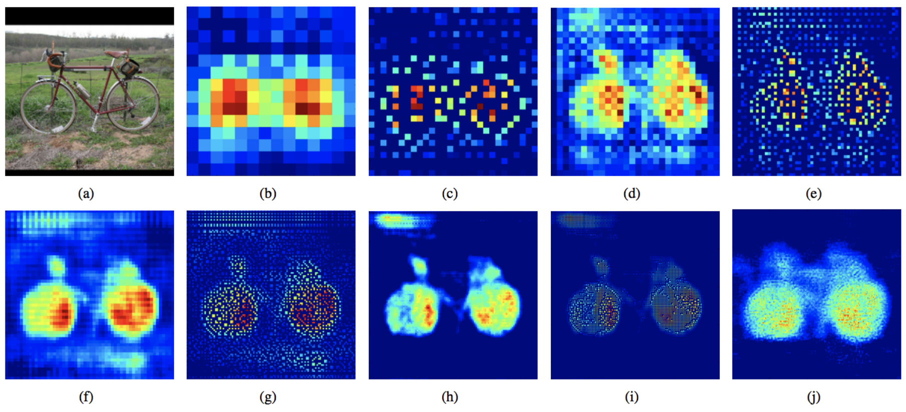
Figure 2.4: Examples of average and max pooling, adapted from [Cnn].

Convolution and pooling layers output high-level features of the input. When the task of a CNN is classification, a fully-connected layer with an input of these features can be used to output a probability of an image belonging to a specific class. This is usually done by applying activation functions that output a probability between zero and one for each possible class using the output of a fully-connected layer. A CNN for classification will later be used in this thesis as an auxiliary network for semi-supervised learning within a GAN, which will be described in more detail in the following sections.

When it comes to the task of semantic segmentation, the output of a CNN is supposed to be a prediction of pixel-wise probabilities indicating whether a specific pixel of an input image belongs to a specific class. To achieve this, the feature maps that were downsampled during the convolutional process of feature extraction need to be upsampled again to match the original size of the input image and form a segmentation map. This process of downsampling and upsampling is also called an encoder-decoder architecture, for which an example is shown in Figure 2.5a.



(a)



(b)

Figure 2.5: Overall architecture of the encoder-decoder network used in [NHH15] in 2.5a. The deconvolution network uses series of unpooling, deconvolution and rectification operations to obtain a dense pixel-wise class prediction map of the input image. In 2.5b the visualization of feature maps throughout the layers of transposed convolutions, where (a) is an input image and (b) to (j) are outputs of consecutive deconvolution (b), (d), (f), (h), (j) and unpooling layers (c), (e), (g), (i). This illustrates the type of information that is extracted throughout the layers, as the deconvolution of earlier convolution layers adds improved class-specific shape information whereas the deconvolution from small feature maps from late convolutional layers hold class information without finer details. Figures adapted from [NHH15].

There are different options to do the upsampling of feature maps. First approaches used bilinear interpolation and, optional, conditional random fields (CRFs) [KK12] as a post-processing step on the segmentation map. A problem of this approach is that information about small and fine objects could get lost in the process of down- and upsampling, if the objects are smaller than the receptive field of a network. The receptive field is "the size of the region in the input that produces the feature" [ANS19] and is, amongst others, influenced by the filter kernel sizes and strides. There have been attempts to reduce the problem of losing information throughout the layers, for example by using different kinds of skip connections between the layers to recover finer details from earlier layers in the predictions. Examples using skip connections are DenseNets [Hua18], U-Nets [RFB15] and ResNets [He16]. Another method is learnable upsampling, which is often referred to as deconvolution or transpose convolution, combined with unpooling layers proposed by Noh et al. [NHH15].

2.1.2 Training

A CNN is trained by using a loss function to calculate the error between its output and ground truth labels or other references. The goal of the CNN is to minimize a loss function by adapting the weights and filter kernels inside the layers to predict an output that causes a lower loss value. The update of the weights and filter kernels is done with the help of optimizers like stochastic gradient descent or Adam [Rud17]. The method of updating the weights and filters of a CNN in a forward direction is called forward propagation. The method of backpropagation based on a loss value is used to calculate the gradients from the output to the input of a neural network using the chain rule. Figure 2.6 depicts the directions of forward- and backpropagation.

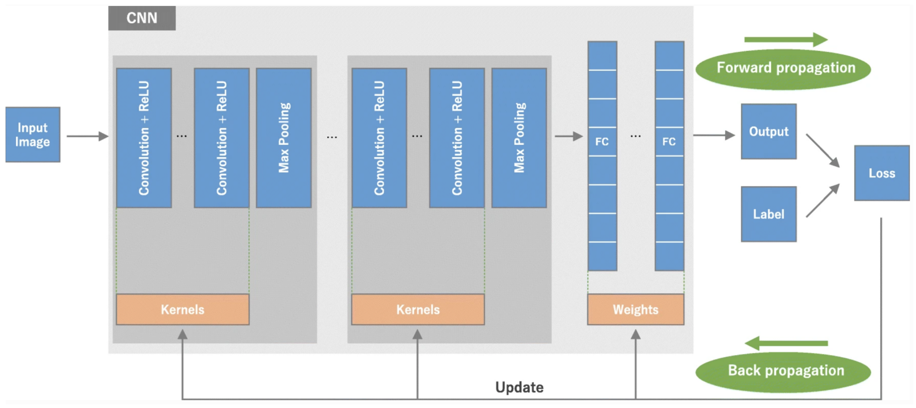


Figure 2.6: Forward- and backpropagation in CNNs, adapted from [Noaa].

2.1.3 Common regularization techniques

A common problem in deep learning is overfitting, where a trained network is too specified on the data it was trained with and does not perform well when introduced to data it has not seen before. To avoid this problem and train a model that is able to generalize well, different regularization techniques can be applied.

One of them is called dropout. At each training iteration it randomly selects nodes to be deactivated during this iteration. As a result, the network does not always have all nodes available for training and the complexity inside the layers is randomly reduced. This temporal random reduction has positive effects on the ability of a network to generalize [kF16]. An example depicting the removal of nodes in a neural network is shown in Figure 2.7. The dropout rate, which sets the percentage of nodes to be temporarily removed, is a hyperparameter in neural networks.

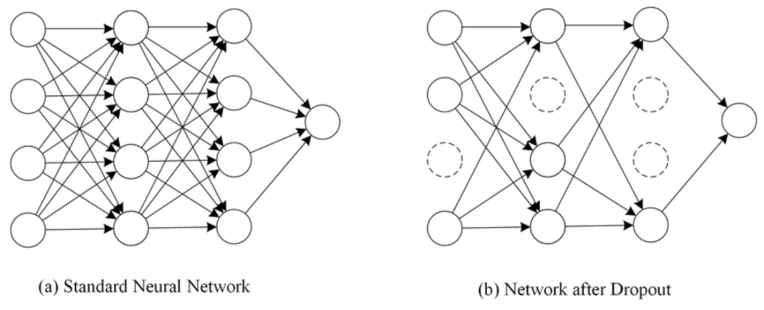


Figure 2.7: An example of a dropout operation applied in a neural network, adapted from [kF16].

Another way to avoid overfitting is augmenting the training dataset. The more different data a network sees during training the easier it can generalize. Apart from manual data augmentation with labeled data, there are methods to automatically augment training data when dealing with images. Adding transformations that preserve annotations, such as rotations, scaling, flipping and shifting to training images can improve the results of a model [SK19].

Early stopping is another method used and tries to stop the training process before a model overfits on the training data. To monitor when the point of overfitting is reached, a training dataset is split into training and validation parts. The training part is used to calculate the loss of the model and correct it, while the smaller validation dataset can be used as a reference about how well the model is performing on data apart from the training set. As soon as the loss of the validation dataset does not decrease anymore while the loss of the training dataset still does, early stopping terminates the training. The validation set is often also used to tune hyperparameters during training, which is why it can not be considered as a true test set, but it can be an indicator to evaluate the generalization ability of a model.

In the following, GANs are going to be introduced. GAN components are often CNNs. The GAN for semantic segmentation used in this thesis also consists of a CNN for segmentation and a CNN for classification which are connected to learn together.

2.2 Generative Adversarial Networks

GANs belong to the field of Deep Generative Modeling. The goal of generative modeling is to take input training examples from a data distribution and learn a model that represents this distribution and is able to generate new examples of the distribution learned. Originally, GANs take a low-dimensional input, like a random noise vector, and learn its transformation to a desired data distribution. Ian Goodfellow et al. [Goo14] first introduced the concept of GANs in 2014. The basic idea is to use two networks, where one network generates an ouput which the other network takes as an input and rates the quality of the generated input based on its own understanding. The two networks are referred to as generator and discriminator, based on their respective roles. In Figure 2.8 a standard GAN network is depicted. It shows the inputs and outputs of generator and discriminator for the task of generating new images belonging to a certain data distribution. The discriminator is trained to learn, whether a certain input is coming from a real world data distribution or was synthetically generated by the generator. The output of the discriminator is in turn used to correct the generator. In this setting, the discriminator takes a role that can be considered as a learned loss function for the generator.

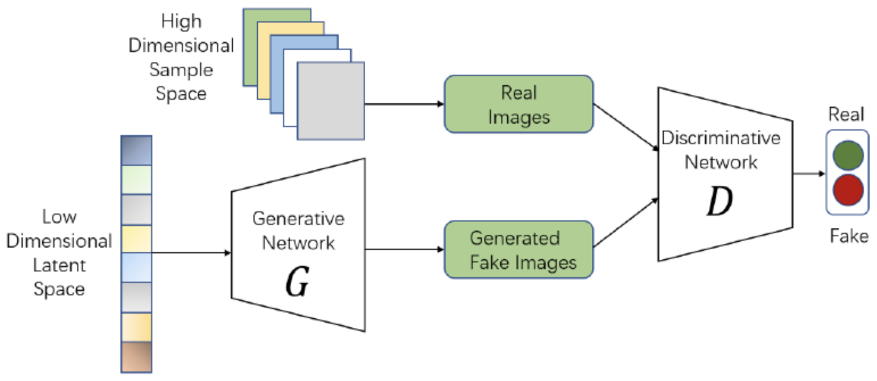


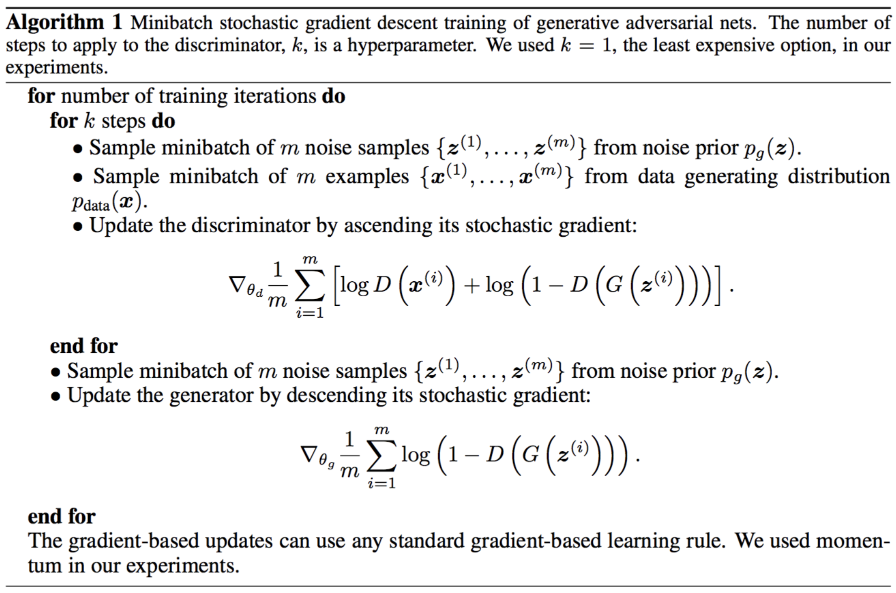
Figure 2.8: Standard GAN architecture for image generation, extracted from [Cai20].

2.2.1 Training

Discriminator and generator are trained alternatingly to get better at their respective task with the goal of outperforming the other network. In standard GANs, the networks are corrected by the binary cross-entropy (BCE) loss, which the discriminator wants to minimize and the generator wants to maximize. It is therefore also referred to as a minimax-loss L_{MinMax}, which is described in Equation 2.2.

$$L_{\text{MinMax}} = E_{\text{x}}[log(D(x))] + E_{\text{z}}[log(1 - D(G(z)))], \tag{2.2}$$

where $D(x)$ is the output of the discriminator for a real data instance x, E_{x} and $D(G(z)$ the discriminator output for a fake data instance generated by the generator G. E_{x} and E_{z} denote the expected values over all real data instances or over all generated inputs, respectively. The higher the BCE loss is, the worse the discriminator is classifying between real and fake inputs. Figure 2.9 shows a pseudo code example of the original GAN training process.



Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

```
for number of training iterations do
  for $k$ steps do
    • Sample minibatch of $m$ noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
    • Sample minibatch of $m$ examples $\{x^{(1)}, \dots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
    • Update the discriminator by ascending its stochastic gradient:
        $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \right].$
  end for
  • Sample minibatch of $m$ noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
  • Update the generator by descending its stochastic gradient:
      $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right).$
end for
```

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Figure 2.9: Pseudo code of the training of the discriminator and generator with stochastic gradient descent optimizer. The 'data generating distribution' means the distribution of real samples, for which the generator is trained to produce similar samples. Figure extracted from [Cai20].

Ideally, the abilities of the generator and discriminator equally improve over the training iterations, until the generator is able to produce perfect generations, which the discriminator is not able to distinguish from real inputs anymore and will always output a probability of 0.5 for both, generated and real samples. Figure 2.10 shows how an ideal training process of a GAN would look like. In reality, this training dynamic is very hard to achieve, because the abilities of the discriminator and generator usually do not stay on the same level. This is mainly due to the fact that the discriminator has an easier task learning to distinguish compared to the generator learning to generate. When the discriminator improves too much, the function approximated by the BCE loss will contain flat regions with gradients close to zero and the generator will not know how to improve. This is why GANs in general are very difficult to train and require extensive hyperparameter optimization. To reduce some training problems like the vanishing gradient, different approaches have been suggested, where some will be introduced in the next sections.

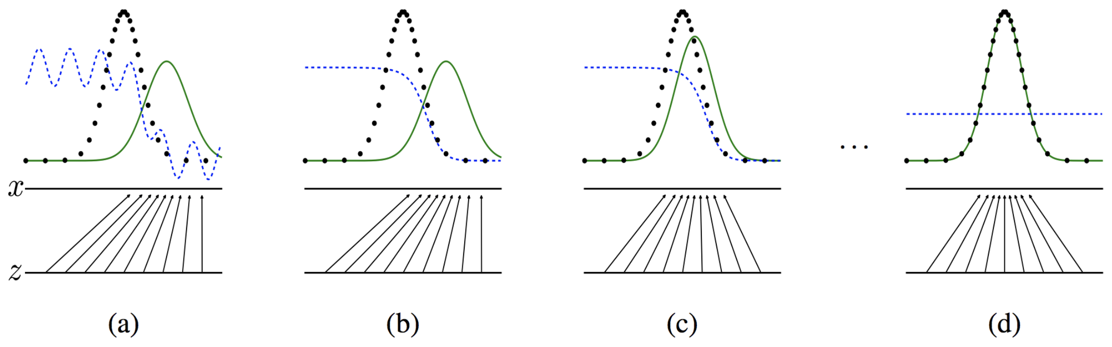


Figure 2.10: An ideal training progress of a GAN, where the black dotted distribution is the real data distribution sampled from x and the green distribution the generated distribution sampled from z. The blue dotted line visualizes the discriminative distribution of the discriminator which distinguishes between real and fake data. From (a) to (d) the generated distribution gets closer to the real distribution, until they are not distinguishable anymore. Figure extracted from [Cai20].

2.2.2 Methods to alleviate training difficulties

An essential task when using GANs is balancing out the abilities of the discriminator and generator networks, where the discriminator usually has a tendency to be stronger. A first approach was training the discriminator less frequently compared to the generator [Cai20]. Consequently, another problem can arise, which is, that the generator learns to only produce a limited amount of very similar or the same samples that are successful in fooling the discriminator. This dynamic is also referred to as mode collapse.

Another approach, suggested by Roth et al. [Rot17] is the addition of noise, like a gaussian noise, to the inputs of the discriminator, to aggravate its learning progress. Salimans et al. [Sal16] proposed the addition of noise in the beginning of the training progress and slowly adding weaker noise during later training iterations or a one-sided label smoothing. Salimans et al. also proposed a feature-matching loss, which will be used in this work as well and explained in more detail in Chapter 3 and 4.

Arjovski et al. [ACB17] proposed an alternative to the traditional GAN training by using a measure of the distance between two probability distributions, namely the earth-movers distance, for the loss function. This GAN variant is called a Wasserstein GAN (WGAN) and has the advantage, that it trains more stable than traditional GANs and in addition yields a loss function that is actually correlated with the quality of the output of the generator.In traditional GANs this does not have to be true, as the loss only measures how good the generator is able to fool the discriminator, whereas the quality of the decisions made by the discriminator is unknown. Because of this difference, the discriminator of a WGAN is referred to as a critic, as it does not distinguish between real or fake inputs but measures the distance between output features of the data distributions of generated and fake data.

To conclude, the components of a GAN need to be balanced out carefully to achieve an effective training progress. Since the introduction of GANs, there have been developed many different architectural variants for different use cases. The following chapter will focus on GANs used in the context of semantic segmentation tasks.

Chapter 3

State of the Art

This chapter will concentrate on recent work regarding semantic segmentation using GANs. The works of Hung et al. [Hun18] and Mittal et al. [MTB19], that are closely related to this thesis, will be described in more detail. There are two main strategies for using GANs to benefit semantic segmentation. The first strategy is to use a GAN to produce new training data to augment a dataset, which is then used for training a semantic segmentation network. The other approach is to use a GAN directly to produce semantic segmentations with the generator and to use the discriminator for evaluation of the segmentations of labeled and unlabeled input images.

3.1 GANs for data augmentation

In the work of Neff et al. [Nef] the authors use a GAN to augment training datasets for medical image segmentation of lungs and also for the bigger segmentation benchmark dataset Cityscapes [Cor16]. To do this, they use a GAN that is first trained to generate image-segmentation pairs. They train the GAN until the quality of the generated image-segmentation pairs does not further improve and then use the fully trained generator of the GAN as an additionial input to a fully-supervised segmentation network based on a U-Net [RFB15]. The generator is then used to produce an unlimited amount of synthetic image-segmentation pairs "on-the-fly", which are, alongside real image-segmentation pairs, used as an input for the U-Net.

Figure 3.1 shows the components of the network described. The authors use a WGAN with gradient penalty based on [Gul17]. For training the GAN, existing segmentation maps and their corresponding images are concatenated along the channel axis, so that the generator is trained to produce image-segmentation pairs and the discriminator is trained to decide, whether a given image-segmentation pair is real or synthetic.

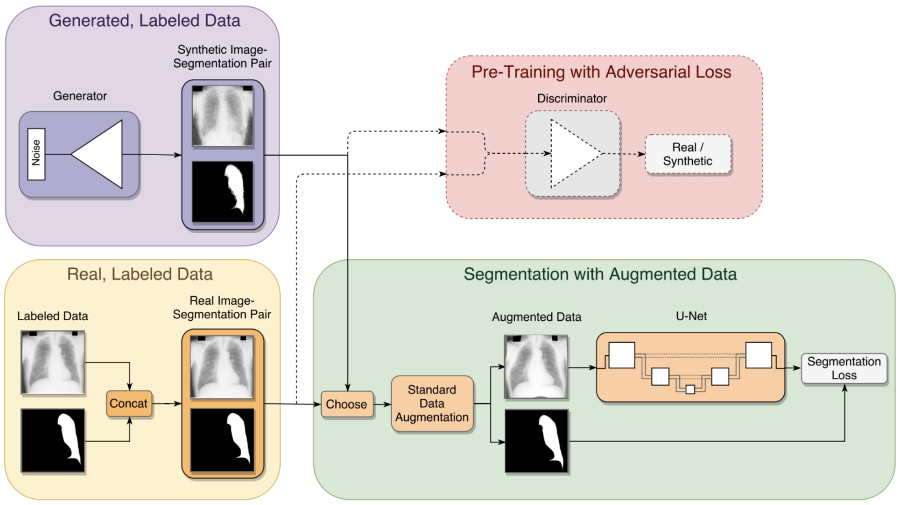

Generated, Labeled Data
Synthetic Image-Segmentation Pair
Generator
Noise
Pre-Training with Adversarial Loss
Discriminator
Real / Synthetic
Real, Labeled Data
Real Image-Segmentation Pair
Labeled Data
Concat
Segmentation with Augmented Data
Augmented Data
U-Net
Segmentation Loss
Choose
Standard Data Augmentation

Figure 3.1: Overview of the GAN architecture for data augmentation by [Nef]. The dashed lines and boxes denote steps and components that are only used during pre-training of the GAN generator.Figure extracted from [Nef]

For the evaluation of their method, Neff et al. [Nef] tried different ratios of real and synthetic data as input to the segmentation network and also trained the network together with standard data augmentation methods like horizontal flipping and without. Their results show a slight improvement when the network is trained with additional synthetic data compared to using only standard data augmentation. But, as the authors write, this slight improvement averaged over three data splits is not meaningful enough to conclude that the data augmentation with a GAN is more beneficial than using standard augmentation techniques. However, there is still an advantage of using GANs for data augmentation, because it "does not require extensive data analysis to find out optimal augmentation parameters" [Nef], as this is done by the GAN itself and therefore can reduce fine-tuning efforts. For further improvement of their method, the authors suggest trying to increase the resolution and representation power of the GAN to be able to produce more detailed synthetic image-segmentation pairs.

3.2 Semi-supervised GANs for direct segmentation

Hung et al. [Hun18] proposed one of the first works considering semi-supervised learning for semantic segmentation. Here, the generator outputs probability maps of semantic labels given an input image and the discriminator learns to produce a spatial confidence map, indicating the quality of the predicted regions, which is then used as a supervisory signal for unlabeled training samples. Similar to that, other probabilistic graphical models such as CRFs [KK12] often get used as a post-processing step. The advantage of using the GAN method proposed by Hung et al. is, that it does not require an extra post-processing module during testing and application of the trained segmentation model. Figure 3.2 shows an overview of the network.

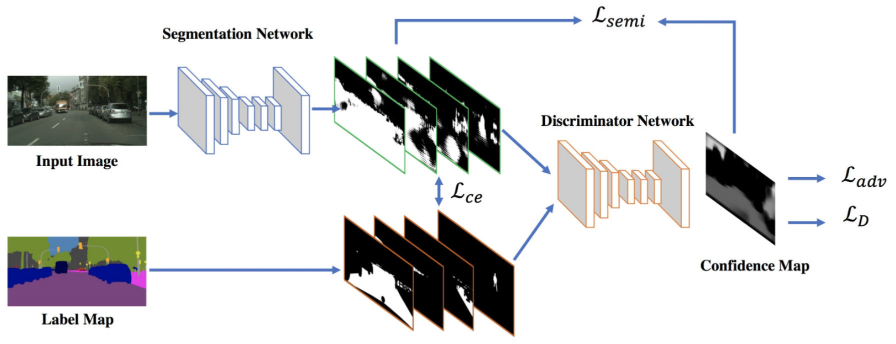


Figure 3.2: Overview of the semi-supervised GAN architecture from Hung et al. The losses shown are explained in more detail in the following. Figure adapted from [Hun18].

As the segmentation network they use a DeepLabv2 [Che17] framework with a pre-trained ResNet-101 [He16] model as a backbone trained on the ImageNet [Den09] and MSCOCO [Lin15] dataset. Due to memory constraints they do not use the multi-scale fusion of DeepLabv2. The CRF post-processing is not applied either. The discriminator is a fully convolutional network with five convolutional layers where the last layer is an upsampling layer that rescales the output to the size of the input map. The networks do not use batch normalization as it is unstable, when, like in this case, small batch sizes are used.

During training, the generator and discriminator are updated jointly at each iteration, where the discriminator is updated only based on labeled images and their predicted segmentation maps. Furthermore, the authors suggest a fully-supervised pre-training of the networks "to prevent the model suffering from initial noisy masks and predictions" [Hun18] and start semi-supervised training afterwards.

For training the discriminator D, there are two possible inputs: segmentation prediction maps from the generator or a one-hot encoded ground truth annotation map. The following loss L_{D} , which is similar to the standard binary cross-entropy loss used in traditional GANs, is used for training the discriminator:

$$L_{\mathrm{D}} = -\sum_{h,w}(1 - y_{\mathrm{n}})log(1 - D(S(X_{\mathrm{n}}))^{(h,w)}) + y_{\mathrm{n}}log(D(Y_{\mathrm{n}})^{(h,w)}), \tag{3.1}$$

where $y_{\mathrm{n}} = 0$ if the training sample is a generated segmentation map from the generator or $y_{\mathrm{n}} = 1$ if the sample is coming from the ground truth distribution. The letters h and w denote the height and width indices of the input maps and $S(X_{\mathrm{n}})$ and Y_{n} are the pixel values of the predicted segmentation map and the ground truth annotation map at the current $[h, w]$ pixel location for class c, respectively. The authors mention that there could be a potential issue regarding the differences of the inputs, where the ground truth maps hold one-hot probabilities with which the discriminator could easily detect whether the probability maps come from the ground truth or not. Nevertheless, Hung et al. report that they do not encounter this problem during training phase and speculate that the reason is that they "use a fully-convolutional scheme to predict spatial confidence, which increases the difficulty to learn the discriminator"[Hun18]. Additionally the ground truth values were slightly diffused before being used as an input to the discriminator, but this did not show any difference in the results.

For the training of the generator, different supervisory methods were applied, depending on whether the training sample used has an annotated ground truth map or is unlabeled. For training with labeled samples, the losses L_{ce} in Equation 3.2 and L_{adv} in Equation 3.3 are used.L_{ce} is a spatial multi-class cross-entropy loss and L_{adv} is the typical GAN adversarial loss using the output of the discriminator for predicted segmentation maps.

$$L_{\mathrm{ce}} = -\sum_{h,w}\sum_{c\in C} {Y_{\mathrm{n}}}^{(h,w,c)}log(S(X_{\mathrm{n}})^{(h,w,c)}) \tag{3.2}$$

$$L_{\text{adv}} = -\sum_{h,w} log(D(S(X_{\text{n}}))^{(h,w)}) \tag{3.3}$$

For generator training with unlabeled samples, the adversarial loss L_{adv} is applied as well, as it only requires the output of the discriminator. In addition, Hung et al. use the following loss L_{semi}, where the discriminator produces a binarized confidence map of trustworthy regions based on a threshold. This binarized confidence map is then used as a ground truth to train with a masked spatial cross entropy loss. In Equation 3.4 $I(.)$ denotes the indicator function to decide, whether the discriminator output of a predicted segmentation at location $[h, w]$ $D(S(X_{\text{n}}))^{(h,w,c)}$ is bigger than a set threshold T_{semi}. If this condition is true, $\hat{Y}_{\text{n}}$, a binarized confidence map, is generated with $\hat{Y}_{\text{n}}{}^{(h,w,c)} = 1$ if $c = argmax_{\text{n}} S(X_{\text{n}}{}^{(h,w,c)})$. The authors report, that a T_{semi} between 0.1 and 0.3 worked best for their case.

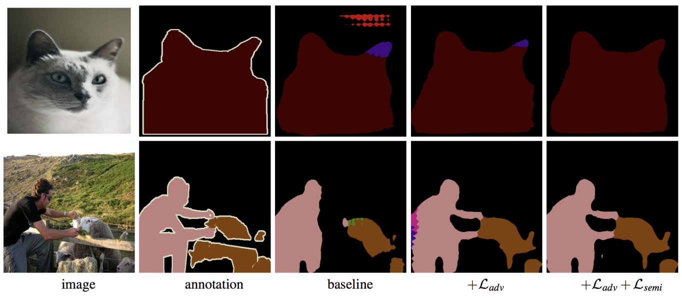
$$L_{\text{semi}} = -\sum_{h,w}\sum_{c \in C} I(D(S(X_{\text{n}}))^{(h,w,c)} > T_{\text{semi}}) \cdot \hat{Y}_{\text{n}}{}^{(h,w,c)} log(S(X_{\text{n}})^{(h,w,c)}) \tag{3.4}$$

Equation 3.5 summarizes all losses that are used for training the generator by trying to minimize it.

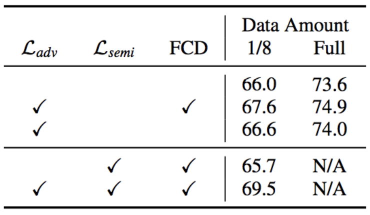
$$L_{\text{seg}} = L_{\text{ce}} + \lambda_{\text{adv}} L_{\text{adv}} + \lambda_{\text{semi}} L_{\text{semi}}, \tag{3.5}$$

where λ_{adv} and λ_{semi} are weights to control the influence of their corresponding losses to the total generator loss L_{seg}. The authors state that "it is crucial to choose a smaller λ_{adv} than the one used for labeled data" [Hun18] when training with unlabeled samples because of the risk of over-correcting the generator.

Hung et al. tested their approach with two benchmark datasets, namely PASCAL-VOC [Eve09] and Cityscapes [Cor16]. They report an improvement of their results from 1.6% to 3.3% mean intersection-over-union (mIoU) for the Cityscapes dataset and 3.5% to 4.0% mIoU compared to their fully-supervised DeepLabv2 baseline and also report improved results with respect to earlier approaches using semi- together with weakly-supervised GANs [SSS17]. Their qualitative results, such as in Figure 3.3a show improvements especially in the border regions of segmented objects. The table in Figure 3.3b shows the results of an ablation study considering the two losses used for semi-supervised learning and the use of a fully convolutional discriminator. Considering these results, L_{adv}, L_{semi} and also the fully convolutional discriminator seem to have a positive influence.



(a)



$\mathcal{L}_{adv}$	$\mathcal{L}_{semi}$	FCD	Data Amount 1/8	Data Amount Full
			66.0	73.6
✓		✓	67.6	74.9
✓			66.6	74.0
	✓	✓	65.7	N/A
✓	✓	✓	69.5	N/A

(b)

Figure 3.3: Visual example segmentations from Hung et al. in 3.3a and the results of an ablation study for the PASCAL-VOC dataset with MSCOCO pre-training about the influences of the semi-superived losses L_{adv} and L_{semi} as well as the use of a fully convolutional discriminator (FCD) against a discriminator that only outputs a single probability value in 3.3b. The data amount means the ratio of labeled samples used for training. Figures extracted from [Hun18].

Based on the work of Hung et al. [Hun18], Mittal et al. [MTB19] developed a semi-supervised GAN for semantic segmentation with several modifications and additions. Opposed to Hung et al., they use a discriminator that outputs a single probability value between 0 and 1 for the whole input image and not a fully convolutional one with an output for every pixel. Furthermore, the inputs for the discriminator are either a concatenation of a ground truth label map and the corresponding rgb image or of a predicted segmentation map and the corresponding rgb image along the channel axis. Instead of the adversarial loss L_{adv}, Mittal et al. use a feature-matching loss. They also do not pre-train the network with full supervision and start semi-supervised learning at the first training iteration. In addition to semi-supervised segmentation they also add a semi-supervised classification branch to their framework. The classification branch outputs an image-wise classification of objects present in an input image. It is then used to correct the segmentation output in case there exists a segmented region of a class that was not classified by the classification branch. Figure 3.4 shows an overview of the architecture.

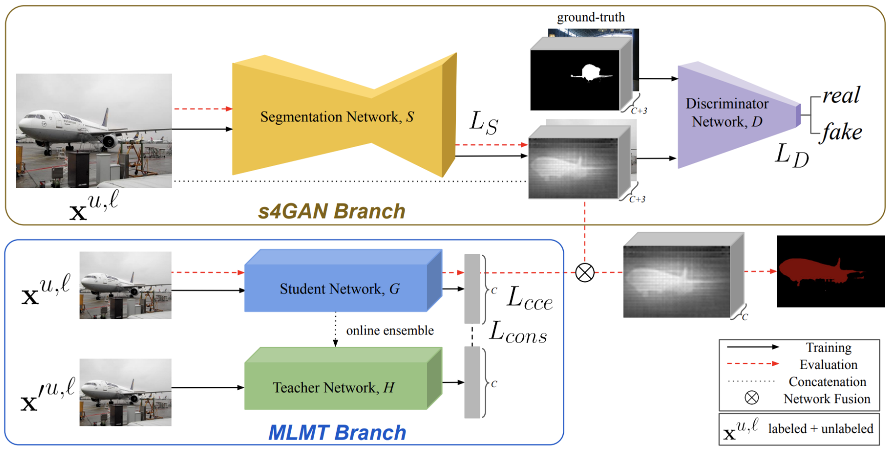


Figure 3.4: Overview of the GAN architecture from Mittal et. al., where the segmentation GAN is the upper s4GAN branch and the classification GAN is denoted as MLMT Branch. Figure extracted from [MTB19].

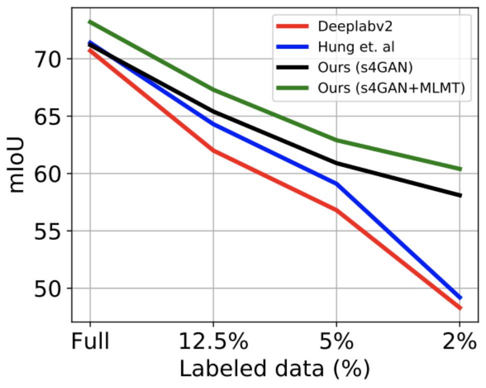
For training the discriminator of the segmentation GAN, the traditional GAN binary cross-entropy loss is used similar to the one used by Hung et al. in Equation 3.1, but here, Mittal et al. also use unlabeled training samples.

For the training of the segmentation generator with labeled images, the standard cross entropy loss is applied. As Mittal et al. do not use a fully-supervised pre-training of the segmentation GAN, they apply a self-training procedure with a threshold of 0.7 to balance out the abilities of the generator and discriminator during training. This means that good generator results for unlabeled images are picked out to be reused as a ground truth for supervised training with cross-entropy loss. This self-training loss aims to prevent the discriminator from becoming too confident too soon, as it generally learns faster than the generator. As an alternative to the adversarial loss used by Hung et al., Mittal et al. apply a feature-matching loss L_{fm} introduced in [Sal16]. L_{fm}, which compares the discriminator outputs of an intermediate layer for a batch sampled from labeled training data and a batch sampled from unlabeled training data. The proposed advantage of this loss is that it compares actual features that the discriminator learned about the ground truth and generated data distributions and not only a single output value. To summarize, the training of the generator network of the segmentation branch is based on the following combined loss L_{S}:

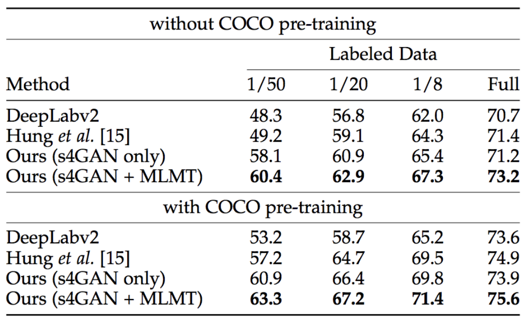
$$L_S = L_{ce} + \lambda_{fm} L_{fm} + \lambda_{st} L_{st}, \tag{3.6}$$

where λ_{fm} and λ_{st} are weights to control the influence of their corresponding losses to L_S. The details of the particular losses will be displayed in the next chapter, as they will be applied in this work as well.

Mittal et al. also tested their approach with the PASCAL-VOC [Eve09] and Cityscapes [Cor16] datasets, among others. With their method they were able to achieve an improvement over their baseline DeepLabv2 model as well as the previously mentioned work of Hung et al. [Hun18]. The authors speculate that the reason for this might be that they not only use labeled images to update the discriminator so their model is less prone to overfitting on the labeled data given. In addition, they found it crucial for the stability to train using the feature-matching loss and not the standard GAN adversarial loss, where the aim is to directly maximize the output of the discriminator. Figure 3.5 shows some of their results. As Table 3.5b shows there is especially an improvement over the method of Hung et al. when no pre-training is applied, which is interesting for domains where pre-training is not helpful.



(a)



without COCO pre-training				
	Labeled Data			
Method	1/50	1/20	1/8	Full
DeepLabv2	48.3	56.8	62.0	70.7
Hung *et al.* [15]	49.2	59.1	64.3	71.4
Ours (s4GAN only)	58.1	60.9	65.4	71.2
Ours (s4GAN + MLMT)	**60.4**	**62.9**	**67.3**	**73.2**
with COCO pre-training				
DeepLabv2	53.2	58.7	65.2	73.6
Hung *et al.* [15]	57.2	64.7	69.5	74.9
Ours (s4GAN only)	60.9	66.4	69.8	73.9
Ours (s4GAN + MLMT)	**63.3**	**67.2**	**71.4**	**75.6**

(b)

Figure 3.5: Figure 3.5a with results from Mittal et al. showing improvements over the baselines of DeepLabv2 and the previously mentioned work of Hung et al. [Hun18] for the PASCAL-VOC dataset and 3.5b comparing the results with and without pre-training with the MSCOCO dataset. Figures extracted from [MTB19]

To summarize, the approach of Neff et al. using a GAN to augment an existing dataset seemed to result in slight improvements, but the usage of semi-supervised GANs for direct segmentation, as applied in the works of Mittal et al. [MTB19] and Hung et al. [Hun18], yielded more promising improvements regarding the segmentation accuracy. Considering these approaches, the work from Mittal et al. seems to result in higher improvements compared to the work from Hung et al. Apart from the reasons mentioned earlier, another factor could be that Mittal et al. use a concatenation of rgb images and the generated segmentation map or ground truth as an input to the discriminator. The additional information coming from the rgb image that was segmented could be important for the evaluation of the segmentation quality by the discriminator.

Chapter 4

Methodology

For the matter of this thesis, the semantic segmentation of smoke and fire, a semi-supervised GAN is applied. This approach has shown good results for training with several benchmark datasets [MTB19] [Hun18]. The work from Mittal et al. [MTB19], which was introduced in Chapter 3, was chosen to be the basis for this work. The reason for choosing this basis is the improved performance with semi-supervised learning which occurred especially when using a very small ratio of labeled training data in the whole training dataset. As their method was tested with big benchmark datasets, the application for small datasets might require some modifications which will be suggested in this chapter.

4.1 Datasets

Two independent datasets for the segmentation of smoke and fire are used for training and testing. In the following, both datasets will be described.

4.1.1 Fire

As the labeled fire dataset, a densely annotated, publicly available dataset for wildfire detection developed by the Université de Corse Pasquale Paoli [Tou17], which will be referred to as the "Corsican" fire dataset in the following, is used for this work. In addition to these labeled images there have been collected 416 unlabeled wildfire images from Google and Bing image search. Table 4.1 characterizes both datasets. A significant difference between the labeled Corsican dataset and the unlabeled dataset is that there is a bigger portion of images from an aerial perspective in the unlabeled dataset. The intention behind that is that the firefront project aims to segment aerial images and through the introduction of these unlabeled images the model has the chance to capture more information for the aerial perspective. Figure 4.1 shows examples of the dominant perspectives of fire within the Corsican dataset and Figure 4.2 shows examples of aerial perspectives added to the unlabeled dataset.

	Fire labeled (Corsican)	Fire unlabeled
Pixel ratio (fire/background) in %	22/78	-
Ground perspective / slightly elevated	577	295
Aerial perspective	15	121
Total number of images	592	416

Table 4.1: Details of the fire dataset.

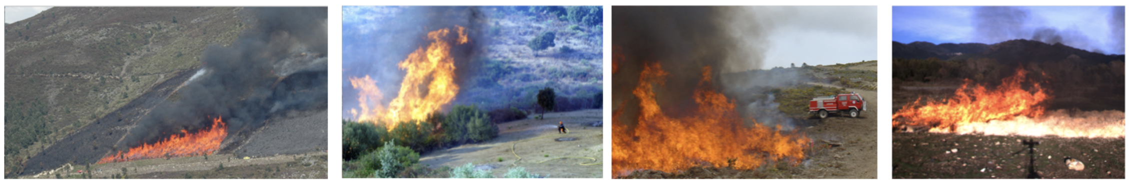


Figure 4.1: Examples of the ground and slightly elevated perspectives of fire images belonging to the Corsican dataset.

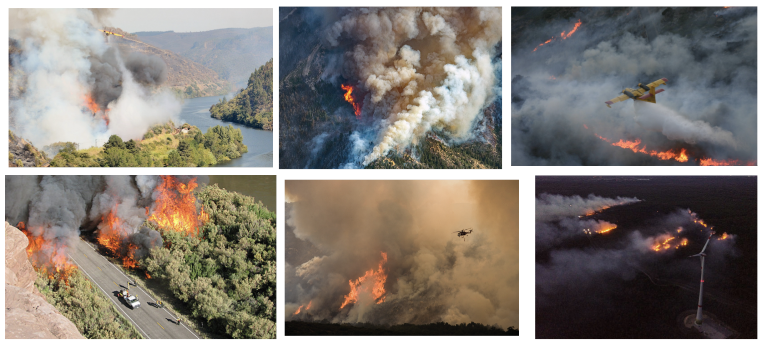


Figure 4.2: Examples with aerial perspectives from the unlabeled fire dataset.

4.1.2 Smoke

To alleviate the problem of inaccurate, strict smoke borders, an "ignore-label" was added to the annotations, where the smoke concentration is very low or if an area is not certainly identifiable as smoke or non-smoke. This way of annotating was also applied in a small, publicly available smoke database [Noad], whose images are also a part of the labeled dataset used in this work. From the total of 200 labeled smoke images, 105 come from the previously mentioned smoke database, 18 images are adapted from the Corsican fire dataset and annotated for smoke, and 77 images were collected and annotated from Google and Bing image searches. For more consistent labels, some of the 105 smoke database images were re-labeled to match the labeling style of the other images. The image areas annotated with the ignore-label are not considered in the calculation of losses and evaluation metrics. This method of labeling is still subjective, but could possibly avoid the confusion of the network during training in smoke border zones and yield more reliable evaluation results. For the task of labeling, the tool DarwinV7 [Noae] was used. In addition to the labeled images there are 148 unlabeled images collected from Google and Bing image search. Table 4.2 characterizes the dataset used and figure 4.3 shows example annotations with the ignore-label.

	Smoke labeled	Smoke unlabeled
Pixel ratio (smoke/ignore-label/background) in %	27/1/72	-
Ground perspective / slightly elevated	50	39
Aerial perspective	125	109
Non-smoke	25	-
Total number of images	200	148

Table 4.2: Details of the smoke dataset.

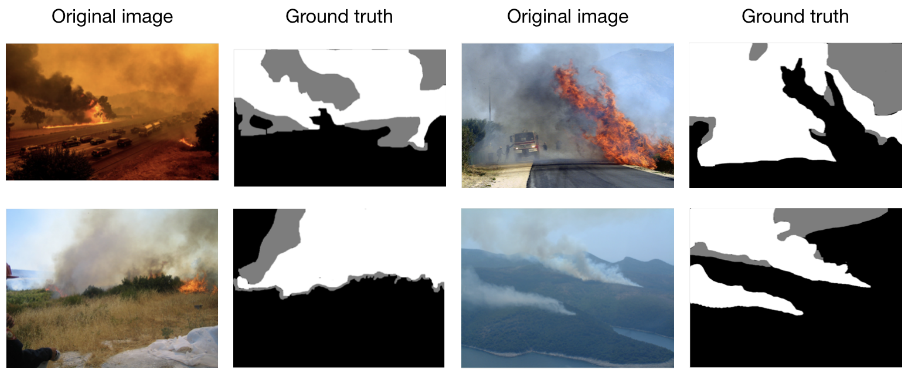


Figure 4.3: Examples of the annotation of smoke images using white labels for smoke and grey labels for uncertain areas that might be smoke and smoke borders.

4.2 Network Architecture and Implementation

The GAN architecture used in this work is based on the segmentation branch of the GAN proposed by Mittal et al. [MTB19]. Figure 4.4 shows an overview of the network components and possible inputs and outputs. The details of the training procedure will be explained in Section 4.2.2.

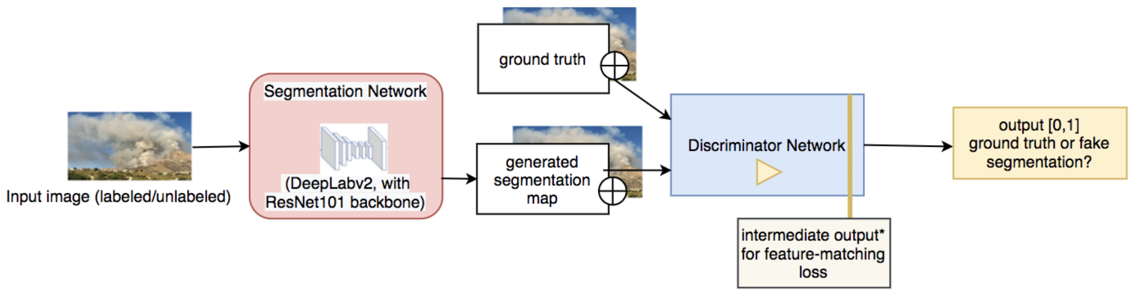

Input image (labeled/unlabeled)
Segmentation Network
(DeepLabv2, with ResNet101 backbone)
ground truth
generated segmentation map
Discriminator Network
output [0,1] ground truth or fake segmentation?
intermediate output* for feature-matching loss

Figure 4.4: Overview of the GAN components and their inputs and outputs used in this work. The ⊕ denotes the concatenation of ground truth or segmentation maps with their corresponding rgb image along the channel axis. The generated segmentation map holds probability values for each pixel belonging to a certain class. The discriminator output is a probability between 0 and 1. The closer the output is to 1 the more confident the discriminator is that the input is a ground truth segmentation. The intermediate output* denotes a vector that is obtained before the last fully-connected layer of the discriminator. The architecture is adapted from the segmentation branch in [MTB19].

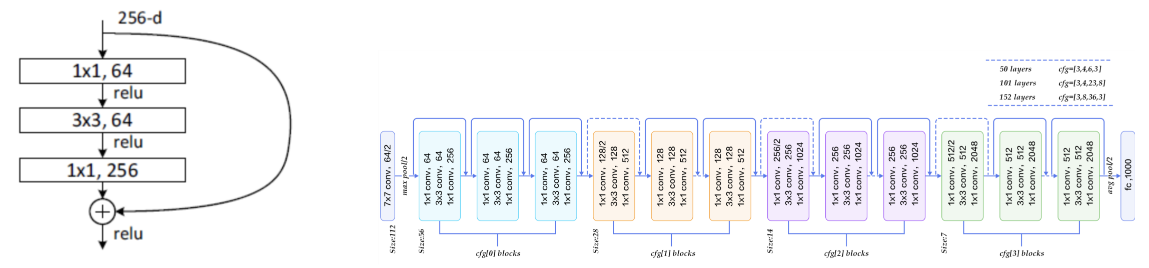
4.2.1 Implementation details

Mittal et al. made the Pytorch implementation of their semi-supervised segmentation GAN [sud21] publicly available, so this implementation will be used as the basis for this work.

Generator network

The segmentation network is a DeepLabv2 framework with a ResNet-101 network [He16] as a backbone, which is pre-trained on the ImageNet [Den09] dataset. DeepLabv3+ [Che18], although being a more recent version of DeepLab, is not applied in this work, because Mittal et al. note that "DeepLabv3+ is unstable in the low-data 'supervised only' setting" [MTB19], which could impair the fair comparison between fully-supervised and semi-supervised learning.

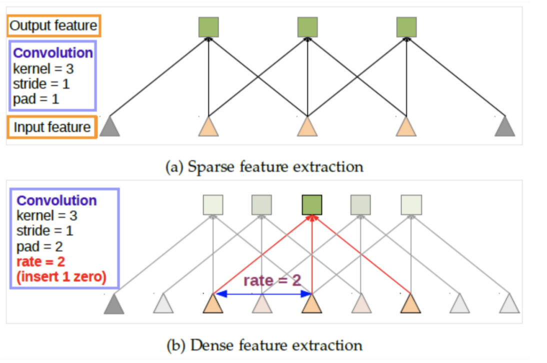
ResNet-101 is a deep network with 101 layers. To be able to learn with this number of layers, the network uses skip connections to minimize the vanishing or exploding gradient problem for very deep networks and to preserve information from previous layers. The skip connections for ResNets are called residual blocks, whereas ResNet-101 uses a specific bottleneck design, depicted in Figure 4.5a. Here, the output value of a previous block is added to the output of the current block. Inside a residual block the convolutions with a filter size of 1x1 are used to reduce the number of input feature maps [He16]. Figure 4.5b shows, how the bottleneck blocks are concatenated with each other through the layers of a default ResNet.



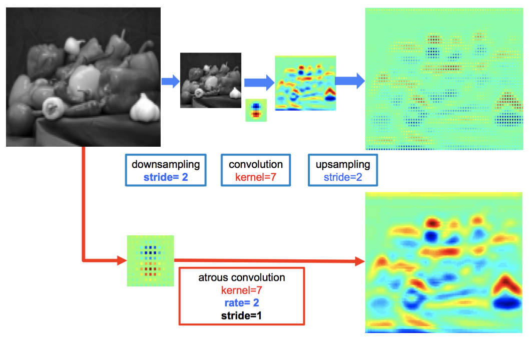
(a) (b)

Figure 4.5: Figure 4.5a shows an exemplary bottleneck residual block for a Resnet and Figure 4.5b shows the architecture of a ResNet with subsequent residual blocks. Figures extracted from [He16].

The DeepLabv2 model works with ResNet-101 as a basis, but uses a different type of convolution in deeper layers, namely atrous convolutions. Their purpose is to enlarge the receptive field of the network, which allows considering a bigger context for the classification decision of image regions. Atrous convolutions use upsampled filters with the resulting gaps filled with zeros. In Figure 4.6a the effect of atrous convolution is illustrated, where each output feature of a convolutional layer was affected by inputs from a wider region on the input feature map, compared with standard convolution. Figure 4.6b shows that atrous convolution does not require a previous downsampling and later upsampling to produce a high resolution output feature map. In addition, DeepLabv2 uses a method called atrous spatial pyramid pooling (ASPP), which can capture image information at multiple scales and output this information to a feature map of a fixed size. Figure 4.7 visualizes ASPP. The standard DeepLabv2 additionally uses a multi-scale structure, which means that it runs in parallel over differently downscaled images. Because the method is computationally very expensive, it is not used in this implementation. Also the CRF post-processing step is not applied in this work.



(a) One-dimensional atrous convolution in the lower image on a high resolution input feature map compared to standard convolution in the upper image on a low resolution input feature map.



(b) Two-dimensional standard convolution with down- and upsampling on the top pipeline with blue arrows compared to atrous convolution following the red arrow pipeline.

Figure 4.6: Atrous convolution in DeepLabv2. "Rate" denotes the distance of the original filter kernel elements with zero filled gaps in between. Figures extracted from [Che17].

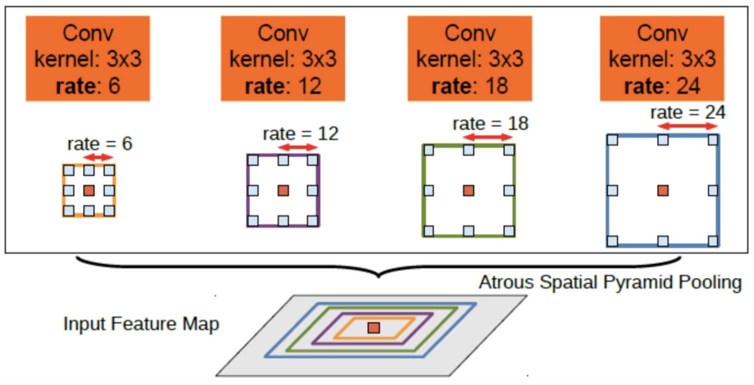


Figure 4.7: Atrous spatial pyramid pooling, extracted from [Che17].

Table 4.3 shows the details of the segmentation network layers. The input size is 312x321, to which input images are cropped. As a standard data augmentation, horizontal flipping is applied. The layer groups conv4 and conv5 use atrous convolution with a rate of 2 and 4, respectively. After the ASPP, an upsampling operation with bilinear interpolation and softmax activation is performed on the output to match the size of the input image. To obtain a binary segmentation mask for evaluation, an argmax funtion is applied to the softmax output.

Layer group	Filter kernel size, number of filters	Number of ResNet blocks	Stride
conv1	7x7x64	-	2
3x3 max pooling			2
conv2	{1x1x64, 3x3x64, 1x1x256}	3	2
conv3	{1x1x128, 3x3x128, 1x1x512}	4	2
conv4	{1x1x256, 3x3x256, 1x1x1024}	23	1
conv5	{1x1x512, 3x3x512, 1x1x2048}	3	1
Atrous spatial pyramid pooling (rate=6,12,18,24)			

Table 4.3: Details of the segmentation network layers.

Discriminator network

The discriminator network is a standard convolutional binary classification network. Table 4.4 shows the details of the network layers. Except the last layer, each convolution layer is followed by a Leaky-ReLU [Xu15] activation function parametrised by 0.2 and a dropout layer with a dropout rate of 0.5. Mittal et al. note that this dropout rate was "crucial for stable GAN training" [MTB19]. After the fully-connected layer, a sigmoid function is applied to output a probability between 0 and 1.

Layer	Filter kernel size, number of filters	Stride
conv1	4x4x64	2
conv2	4x4x128	2
conv3	4x4x256	2
conv4	4x4x512	2
Global average pooling		
Fully-connected layer		

Table 4.4: Details of the discriminator network layers.

4.2.2 Training procedure

The discriminator and generator are trained alternatingly at each iteration by calculating the losses described in the following subsections and updating the learnable weights of the networks.

Discriminator

For training the discriminator, the standard BCE-loss, as it is proposed in the original GAN paper [Goo14], is applied. Equation 4.1 describes the discriminator loss L_{D}, where $\mathbb{E}_{(\mathrm{x(l),y(l)})}$ denotes a batch of rgb images and ground truth segmentation maps sampled from the labeled training data distribution and $\mathbb{E}_{\mathrm{x}}$ a batch of rgb images sampled from the labeled or unlabeled training data distribution. $D(y(l) \oplus x(l))$ is the output of the discriminator D for the input of the ground truth segmentation map $y(l)$ concatenated with the corresponding rgb image $x(l)$. $D(S(x) \oplus x)$ is the output of D for the generated segmentation map by the segmentation network S concatenated with the corresponding rgb image x, where x can be samples from the labeled or unlabeled training data distribution. Figure 4.8 illustrates the inputs and outputs involved for calculating L_{D}.

$$L_{\mathrm{D}}{=}\mathbb{E}_{(\mathrm{x(l),y(l)})}[logD(y(l) \oplus x(l))] + \mathbb{E}_{\mathrm{x}}[log(1 - D(S(x) \oplus x))] \tag{4.1}$$

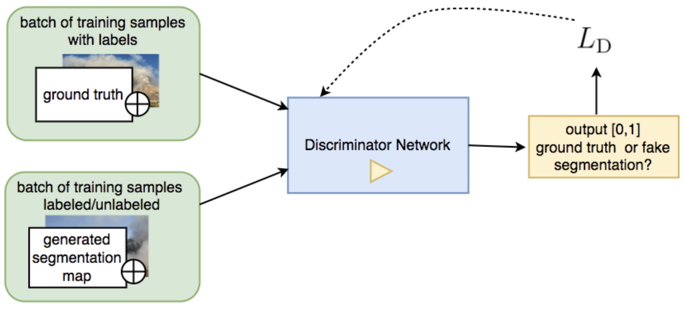

batch of training samples with labels
ground truth
batch of training samples labeled/unlabeled
generated segmentation map
Discriminator Network
output [0,1] ground truth or fake segmentation?
L_{D}

Figure 4.8: GAN components involved to compute the discriminator loss L_{D}. The $\oplus$ denotes the concatenation of ground truth or segmentation maps with their corresponding rgb image along the channel axis.

Generator

The generator uses different losses, depending on whether the input image comes from the labeled or unlabeled training data distribution. For labeled images, the standard cross entropy loss in Equation 4.2 is applied, where y^ℓ is the value of the ground truth segmentation map at location (h, w, c) and $S(x^\ell)(h, w, c)$ is the generated segmentation mask from the segmentation network. Figure 4.9 illustrates the cross-entropy loss.

$$L_{\text{ce}} = -\sum_{h,w,c} y^\ell(h, w, c) log(S(x^\ell)(h, w, c)), \tag{4.2}$$

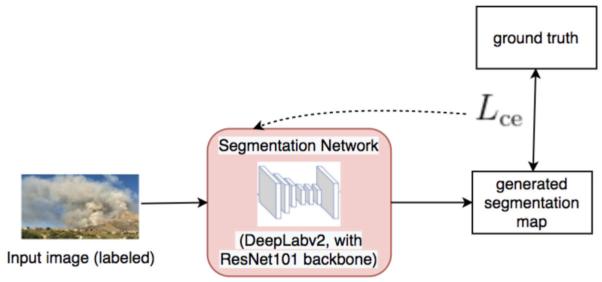


Figure 4.9: GAN components involved to compute the cross-entropy loss L_{ce} for training with samples coming from the labeled data distribution.

To calculate the loss for unlabeled training samples, two different losses are applied. The first one adopts a self-training approach, where generated segmentation maps that are able to fool the discriminator into thinking that they come from the ground truth data distribution are reused as pseudo ground truth segmentation maps to calculate the cross-entropy loss L_{ce}. The pseudo ground truth segmentation maps are obtained by applying an argmax function to the generated segmentation map. The threshold that decides whether a generated segmentation map is good enough to be reused as a ground truth is a hyperparameter in the training process, which is set to 0.7 in the work of Mittal et al. The intention behind the self-training loss is to encourage the segmentation network to predict segmentation maps that are able to fool the discriminator and balance out the training dynamics, where the discriminator tends to learn faster than the generator. Figure 4.10 illustrates the self-training loss.

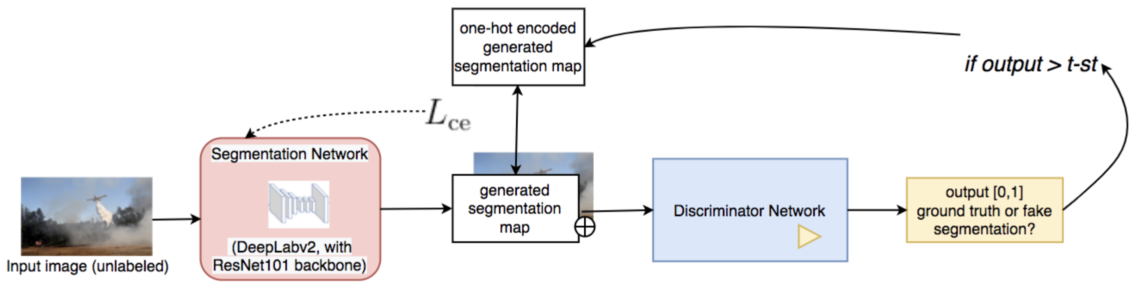

one-hot encoded generated segmentation map
if output > t-st
L_{ce}
Segmentation Network
(DeepLabv2, with ResNet101 backbone)
generated segmentation map
Discriminator Network
output [0,1] ground truth or fake segmentation?
Input image (unlabeled)

Figure 4.10: GAN components involved to compute the self-training loss L_{st} for training with samples coming from the unlabeled data distribution, t-st denotes the self-training threshold.

The second loss that is used for unlabeled training samples is the feature-matching loss, originally from [Sal16]. Equation 4.3 describes this loss, where $\mathbb{E}_{(\mathrm{x(l)},\mathrm{y(l)})}$ denotes a batch of training samples from the labeled training data distribution and $\mathbb{E}_{\mathrm{x(u)}}$ is a batch of rgb images sampled from the unlabeled training data distribution. $D_{\mathrm{k}}(.)$ denotes an intermediate output of the discriminator after the last convolutional layer with global average pooling before the fully-connected layer is applied. The intermediate output is calculated for labeled training images concatenated with their corresponding ground truth $y(l) \oplus x(l)$ and unlabeled samples with their corresponding generated segmentation map $S(x(u)) \oplus x(u)$. Because the intermediate output holds a more detailed feature information about the input, the loss is called feature-matching loss, as it computes the difference between samples coming from the ground truth and unlabeled samples with generated segmentation maps. In Figure 4.11 the components involved for L_{fm} are illustrated.

$$L_{\mathrm{fm}} = ||\mathbb{E}_{(\mathrm{x(l)},\mathrm{y(l)})}[D_{\mathrm{k}}(y(l) \oplus x(l))] - \mathbb{E}_{\mathrm{x(u)}}[D_{\mathrm{k}}(S(x(u)) \oplus x(u))]|| \tag{4.3}$$

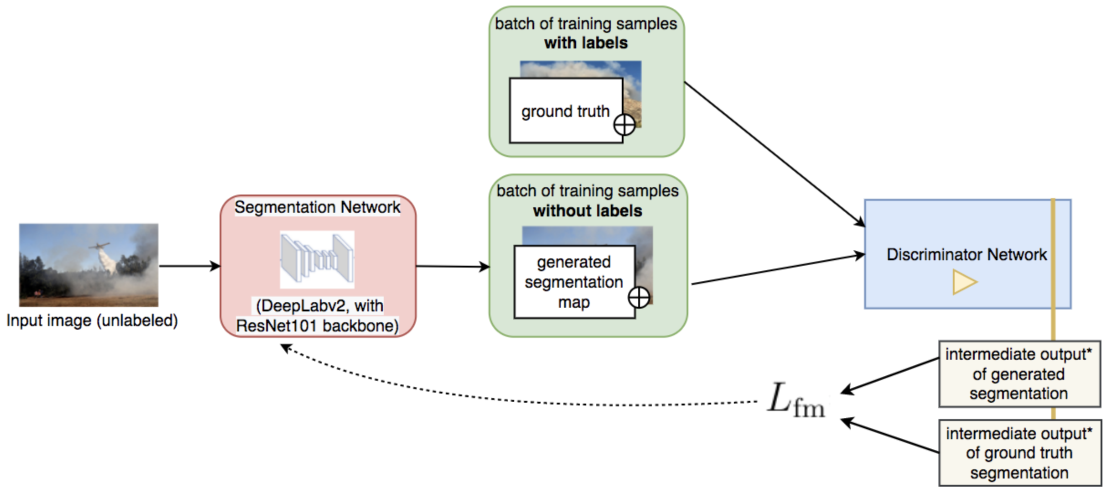

batch of training samples
with labels
ground truth
batch of training samples
without labels
generated segmentation map
Segmentation Network
(DeepLabv2, with ResNet101 backbone)
Input image (unlabeled)
Discriminator Network
L_{fm}
intermediate output* of generated segmentation
intermediate output* of ground truth segmentation

Figure 4.11: GAN components involved to compute the feature-matching loss L_{fm}. The intermediate output* is the output vector after the 4th convolutional layer and global average pooling of the discriminator.

4.3 Experiments

The goal of the following experiments is to evaluate whether a semi-supervised GAN, as described before, is applicable for the task of smoke and fire segmentation when using only very small labeled training datasets and if there are advantages compared to fully-supervised training. As the baseline of the experiments, the work from Mittal et al. [MTB19], was applied to big benchmark datasets like PASCAL-VOC [Cor16] and Cityscapes [Cor16], the GAN might need several modifications to perform well for the use case of this thesis. For the first three experiments only the Corsican fire dataset is used for training and split into labeled and unlabeled parts. This is done to avoid an interfering influence of training samples collected from other sources that might be too different when comparing fully-supervised with semi-supervised training. The insights obtained by training with this sanity check fire dataset will then be applied to smoke and fire datasets that use all training images available in the last experiment. The models for the segmentation of smoke and fire are trained separately, so that each model specialises either on the detection of smoke or on the detection of fire.

The experiments were conducted using a NVIDIA GeForce RTX 2080 Ti GPU with 12GB memory, CUDA version 10.2 and Pytorch 1.5.1.

For each of the following experiments the results will be computed for three random dataset splits. Each model is trained until 8000 training iterations with a batch size of 4.

4.3.1 Optimization of hyperparameters

GANs have shown to be hard to train because of training instabilities and their sensitivity to hyperparameters, especially when applied to different domains of training data. This is why different hyperparameter settings will be tested against the original baseline settings. Mittal et al. [MTB19] especially emphasized the role of the self-training and feature matching loss weights, λ_{st} and λ_{fm}, and the threshold for the self-training loss in the calculation of the total generator loss L_{S} in Equation 4.4 as important for balancing out the abilities of generator and discriminator. This is why different values of these hyperparameters are tested. For these experiments, the Corsican dataset is split as shown in Table 4.5.

$$L_{\mathrm{S}}=L_{\mathrm{ce}}+\lambda_{\mathrm{fm}}L_{\mathrm{fm}}+\lambda_{\mathrm{st}}L_{\mathrm{st}}, \tag{4.4}$$

Number of train images		Number of test images
labeled	unlabeled	
52	364	176

Table 4.5: Dataset split for hyperparameter optimization.

4.3.2 Fully-supervised pre-training

Mittal et al. [MTB19] did not adopt the fully-supervised pre-training approach suggested by Hung et al. [Hun18]. They speculate that this method leads to overfitting of the discriminator to the labeled training data distribution. This effect is even stronger, because the discriminator is only trained with labeled data, even after the beginning of semi-supervised learning of the generator. On the other hand, Hung et al. [Hun18] use the fully-supervised pre-training to gain training stability during the first training iterations. In this experiment the goal is to test, if a shorter pre-training phase than the one proposed by Hung et al. and the subsequent training of the discriminator with labeled as well as unlabeled samples can be beneficial for the training stability and balance between discriminator and generator, without the discriminator overfitting too much on the labeled training samples. To do so, the results of a fully-supervised pre-training until a certain starting point of semi-supervised training from 25, 500, 1000, 1500 and 2000 iterations will be compared. Figure 4.12 visualizes the difference between the previously mentioned works and this approach.

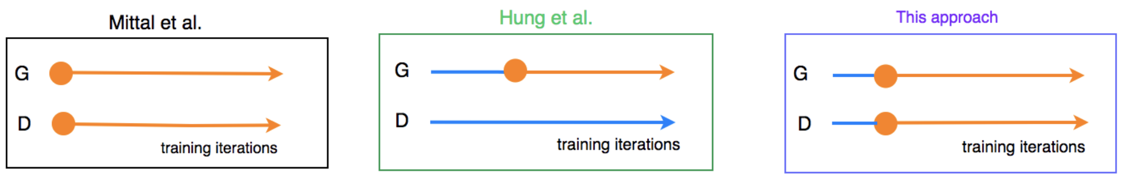


Figure 4.12: Starting points of semi-supervised learning (orange dot) of the works of Mittal et al. [MTB19], Hung et al. [Hun18] and this approach. G and D denote the generator and discriminator, respectively. The blue line denotes fully-supervised learning with training samples from the labeled training data distribution and the orange line denotes semi-supervised learning with both, the unlabeled and labeled training data.

After having obtained the results of hyperparameter optimization and fully-supervised pre-training, different combinations of these parameters will be evaluated, as they are likely to be interdependent.

4.3.3 Comparison of different labeled-ratios

To analyse the effect of semi-supervised learning related to different ratios of labeled and unlabeled training images, experiments with labeled-ratios of 2.5%, 5%, 12.5% and 50% are performed. Mittal et al. [MTB19] reported an increasing positive effect of semi-supervised learning for low labeled-ratios when using big datasets. This experiment wants to test, whether this relation is also true for a small dataset like the Corsican dataset. Table 4.6 shows how the dataset is split for each labeled-ratio.

Labeled-ratio in %	Number of train images		Number of test images
	labeled	unlabeled	
2.5	8	408	176
5	20	396	176
12.5	52	364	176
50	208	208	176
100	416	-	176

Table 4.6: Data splits of the Corsican dataset for different labeled-ratios.

4.3.4 Training with full size datasets

With the conclusions made from the previous experiments with the Corsican sanity check dataset, the best settings are applied on the fire and smoke datasets using all available labeled and unlabeled images. This is done to see whether the effects of semi-supervised learning are comparable to the effects on the Corsican dataset alone when using datasets with images from many different sources. Furthermore it would be beneficial to use all available labeled images for fully-supervised learning. The datasets are split as shown in Table 4.7.

	Number of train images		Number of test images
	labeled	unlabeled	
Fire dataset	416	416	176
Smoke dataset	140	148	60

Table 4.7: Dataset splits of smoke and fire when using the full size datasets.

Chapter 5

Evaluation

5.1 Evaluation Metrics

To evaluate the segmentation performance of the network tested, two common metrics are applied. The first one is the intersection over union (IoU) and the second the F1-score, which is the harmonic mean between precision and recall. The advantage of IoU over simple pixel accuracy is that it takes into account class imbalances in the dataset, as the portions of smoke and fire pixels with respect to the number of background pixels are not equal.

The IoU is calculated as follows:

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union} = \frac{TP}{(TP + FP + FN)}, \tag{5.1}$$

where TP, FP and FN are the number of true-positive, false-positve and false-negative classifications of pixels for a class, respectively.

The precision indicates how accurate the true positive predictions are with respect to all positive predictions. The higher the precision the less false positives are predicted by a model.

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

On the other hand, the recall indicates how many true positive predictions out of the actual number of positive class pixels a model is able to find. It is calculated as follows:

$$Recall = \frac{TP}{TP + FN}. \tag{5.3}$$

As for the segmentation of smoke and fire, both metrics, recall and precision, are equally important, so the F1-score as their trade-off is applied as well. IoU and F1-score are very similar, but the F1-score does not penalise single instances of bad classification as much as the IoU.

$$F1 = \frac{2TP}{2TP + FP + FN} \tag{5.4}$$

Another measure that will get used is the comparison of the number and size of connected components between ground truth segmentation maps and predicted segmentation maps. Knowing how many areas of a certain size are predicted, compared to the statistics of the ground truth, could give further insight about the segmentation characteristics of a model.

5.2 Results

5.2.1 Hyperparameter optimization and fully-supervised pre-training

Table 5.1 shows the mean IoU (mIoU) results for fire when training the GAN with the Corsican fire sanity check dataset with a 12.5% labeled-ratio for different hyperparameter settings. For each setting the results are averaged over three random data splits. The fully-supervised baseline of these data splits, which is only trained with the 12.5% labeled images of each split, reaches a mIoU of 86% with a standard deviation of 0.6. Table 5.2 shows the mIoU results for fire with different starting points of semi-supervised learning.

Hyperparameter	Value	mIoU fire in %	Standard deviation
Self-training loss weight	0.6	85	1.1
	0.7	85	1.4
	0.8	84	1.3
	1.0*	85	0.9
Self-training threshold	0.4	**87**	1.1
	0.5	**88**	0.8
	0.6	85	0.9
	0.7*	85	0.5
Feature-matching loss weight	0.1*	85	1.2
	0.2	**87**	0.9
	0.3	86	0.6
	0.4	**87**	0.7

Table 5.1: Results for different hyperparameter settings for the Corsican dataset with a labeled-ratio of 12.5%. The mIoU results reaching a higher value than the fully-supervised baseline are denoted in bold. The values annotated with * are the original settings used by Mittal et al. [MTB19].

Starting point of semi-supervised learning (training iterations)	mIoU fire in %	Standard deviation
0*	85	0.5
25	86	0.6
500	**87**	0.6
1000	**88**	0.4
1500	86	0.5
2000	85	0.8

Table 5.2: Results for different starting points of semi-supervised learning for the Corsican dataset with a labeled-ratio of 12.5%. The mIoU results that reach a higher value than the fully-supervised baseline are denoted in bold. The values with * are the original settings used by Mittal et al. [MTB19].

Noticeable is that the semi-supervised training with original parameter settings from

Mittal et al. [MTB19] yield a slightly worse result of 85% mIoU compared to fully-supervised training with only the labeled data portion of 12.5% with an mIoU of 86%. On the other hand, the semi-supervised method achieves slightly better results, when either the self-training threshold is lowered, the feature-matching loss weight is increased or if the GAN is trained fully-supervised for the first 500 or 1000 iterations. This could indicate that, with the original settings, the discriminator is much stronger than the generator and the generator might not be able to catch up with its progress. To see how the test results change along the training process, Figures 5.1 and 5.2 show the results for the first data split of each parameter setting at certain checkpoints during training.

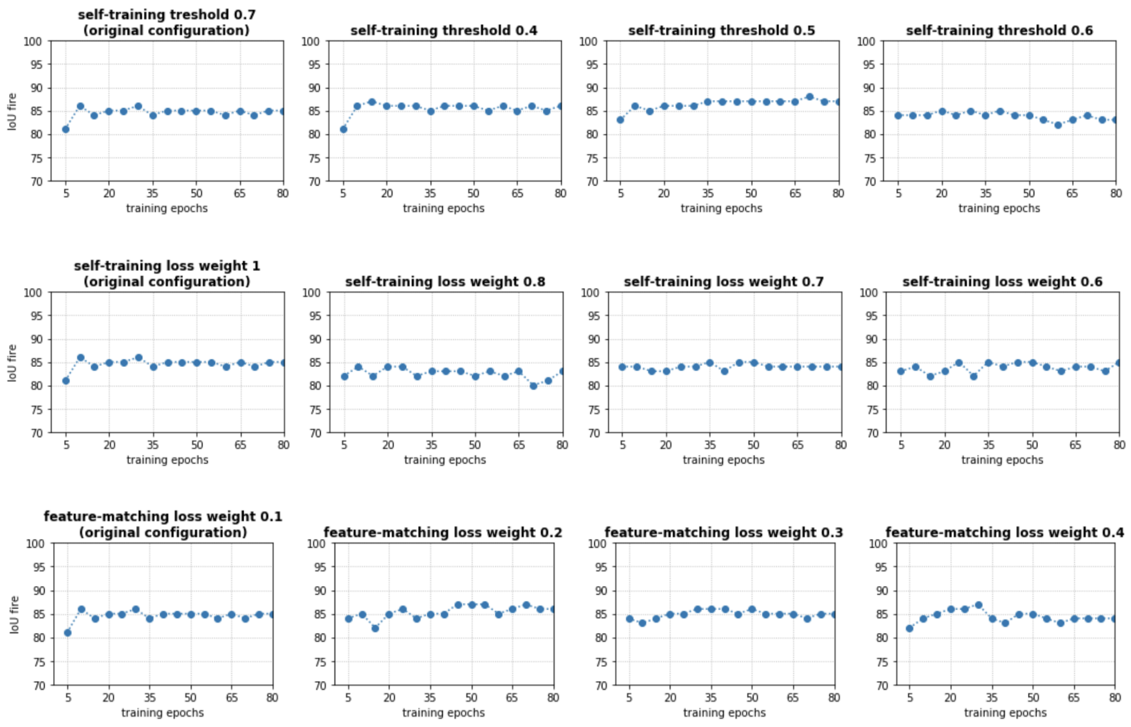


Figure 5.1: Test IoU fire results over training iterations for data split 1.

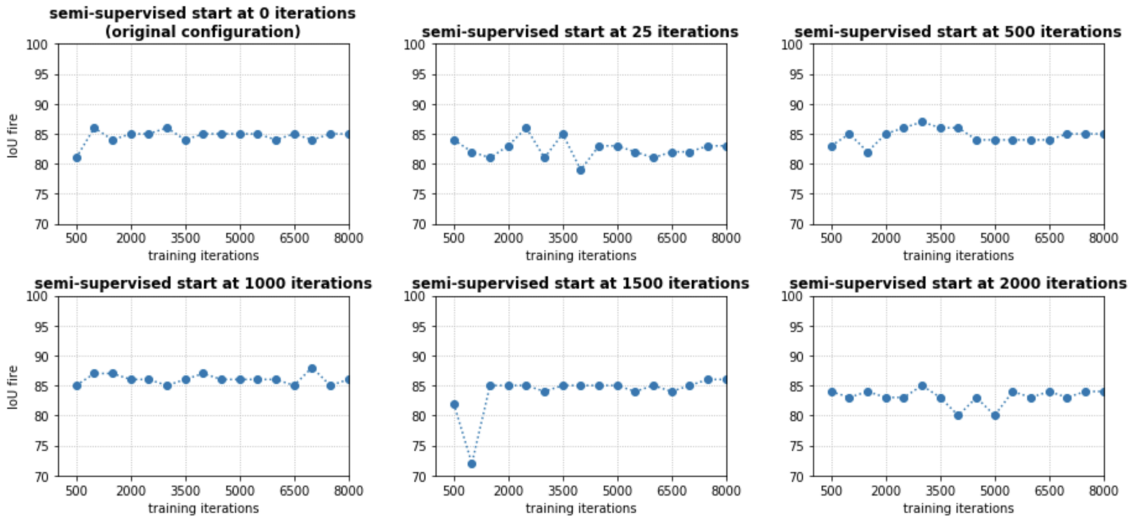


Figure 5.2: Test IoU fire results over training iterations for data split 1 after different starting points of semi-supervised learning.

The following paragraphs will discuss the possible reasons for each hyperparameter change that resulted in a better mIoU for this experiment.

A lower self-training threshold of 0.5 accepts generator segmentation predictions that are still able to fool the discriminator, but not as good as segmentation predictions with a discriminator output of originally 0.7. If the generator is not able to produce segmentation predictions higher than the original self-training threshold, the self-training loss is not applied and the generator does not receive a helpful learning signal from the discriminator. Lowering the self-training threshold could be a way for the generator to gain more helpful correction from the discriminator. When the definition of a good segmentation result is lowered from 0.7 to 0.5, the weaker generator has a higher chance to produce good segmentation maps and benefit from the self-training loss.

A higher feature-matching loss weight emphasizes the influence of the feature-matching loss, which corrects the generator to produce segmentation maps that have similar feature statistics as the ground truth segmentation maps. If the discriminator learned a good feature representation of the ground truth segmentation maps, the feature-matching loss is a valuable input for the slower learning generator.

The later starting point of semi-supervised learning seems to stabilize the generator and discriminator with fully-supervised training, which can then refine the segmentation quality with semi-supervised learning. Table 5.2 shows that there is also a point, at 1500 and 2000 iterations, when the later start of semi-supervised learning has no positive effect on the results anymore. This could be due to overfitting of both networks on the labeled training dataset, which is what Mittel et al. [MTB19] warned about with respect to the later semi-supervised start applied by Hung et al. [Hun18]. So it seems to be important to not train fully-supervised for too long when the labeled training data is scarce.

As some of the changed hyperparameter settings resulted in a slight improvement of the semi-supervised fire mIoU, combinations of those settings were tested in order to see if these combinations could have an even higher positive impact. The combinations tested were a later start of semi-supervised learning with different self-training thresholds, a higher feature-matching loss weight, and combinations where these parameters change their values at different iterations to possibly adapt to the changing abilities of the discriminator and generator during training. The results for the combinations tested did not further improve the mIoU. In Appendix A the test mIoU results of each parameter combination over the training iterations are shown for the first data split.

For the next experiments, a hyperparameter setting was chosen as an 'optimized' semi-supervised version. The self-training threshold of 0.5 and the later start of semi-supervised learning at 1000 iterations both lead to a mIoU of 88% and, looking at the results over the training iterations in figures 5.1 and 5.2, both do not show strong instabilities of results during the training process. Their combination did not further improve the results. Considering the standard deviation of each result, the later start of semi-supervised learning has a lower standard deviation of 0.4 compared to 0.8. Because a lower variance of the results is more favourable, it was chosen to be the 'optimized' parameter setting.

Figures 5.3 , 5.4 and 5.5 show some qualitative results of the segmentations comparing the fully-supervised baseline and semi-supervised learning with the optimized setting with the later starting point of 1000 iterations.

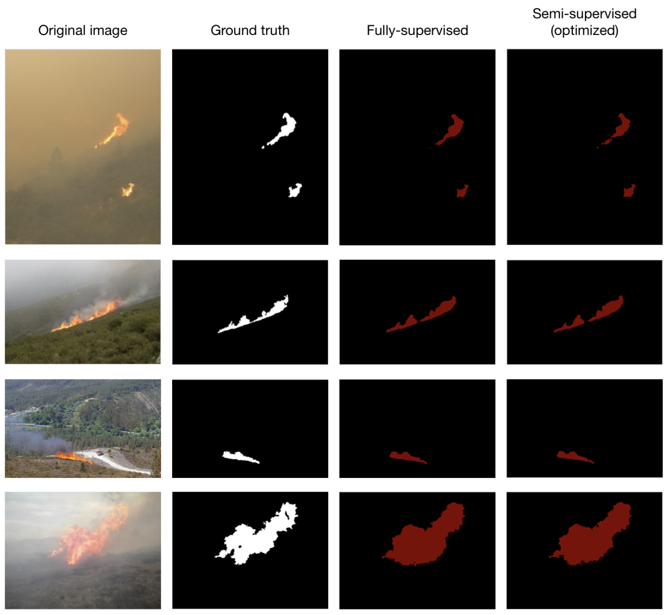


Figure 5.3: Examples of good segmentations from fully-supervised and semi-supervised (optimized) training for the 12.5% labeled-ratio Corsican fire dataset.

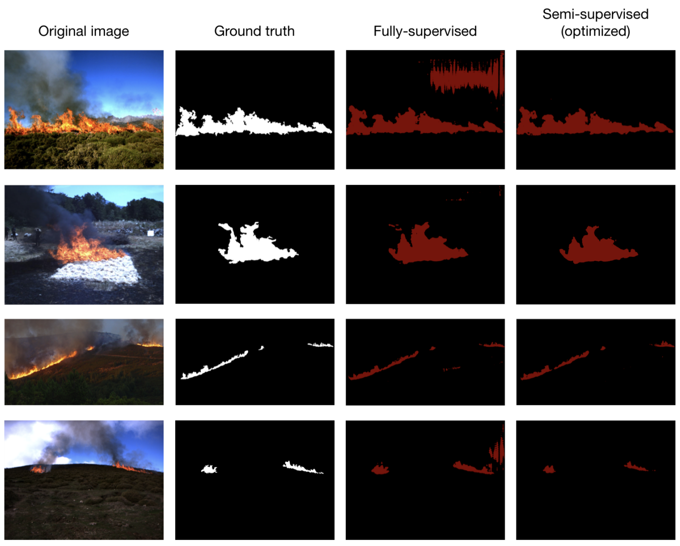


Figure 5.4: Examples of segmentations for the 12.5% labeled-ratio Corsican fire dataset where semi-supervised (optimized) training detected less false-positive regions compared to fully-supervised training.

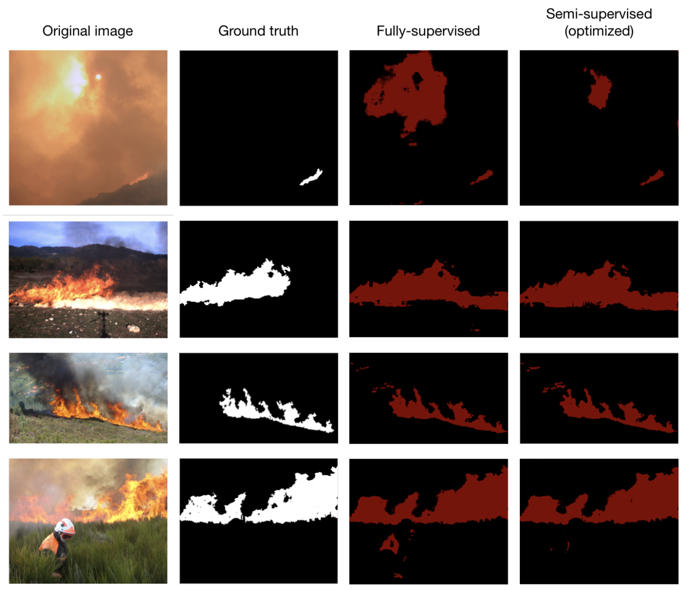


Figure 5.5: Failure modes for both, fully-supervised and semi-supervised training.

Noticeable is that fully-supervised learning seemingly detected more false-positive areas compared to semi-supervised learning. Some examples of this phenomenon are shown in Figure 5.4. An attempt to describe this difference quantitatively is the comparison of the number and size of connected components in the ground truth, semi-supervised and fully-supervised segmentation maps. If fully-supervised learning detects more and mostly small false-positive areas apart from the flame, the segmented components should be more numerous and smaller compared to the segmented components of semi-supervised learning. Table 5.3 shows how many connected components were found in the segmentation maps of each method. Because the numbers for fully-supervised and semi-supervised learning were very high, a morphological closing operation was applied on the segmentation maps, to eliminate scattered predictions where the unique components are so close to each other that they would belong semantically together. On the other hand, false-positive areas with a higher distance to the flame still continue to be an individual area. An example is shown in Figure 5.6 .

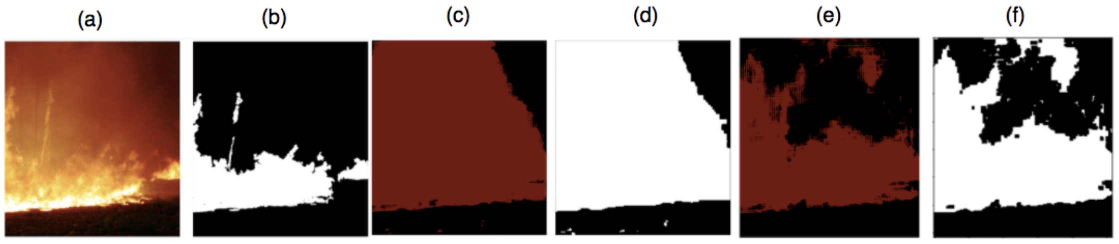
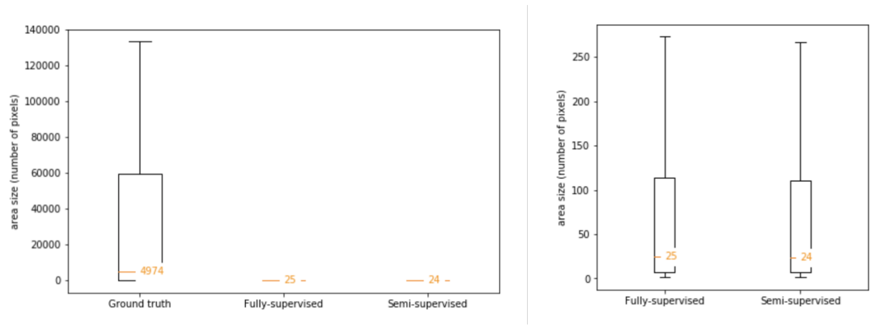


Figure 5.6: Example of the morphological closing operation applied on the segmentation maps of fully-supervised and semi-supervised results. (a) is the original image, (b) the ground truth segmentation, (c) the fully-supervised segmentation, (d) the fully-supervised segmentation after morphological closing, (e) the semi-supervised segmentation and (f) the semi-supervised segmentation after morphological closing.

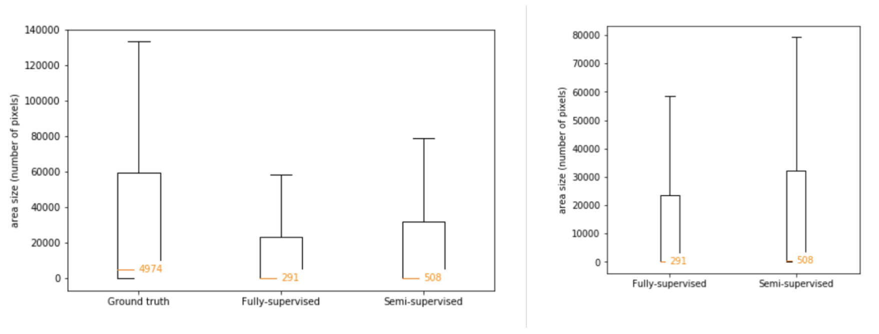
After the post-processing with morphological closing, the numbers of connected components for both methods are lower, while the semi-supervised approach is a bit closer to the ground truth. The boxplots in Figure 5.7 compare the distributions of segmented areas with regard to their size. Fifty percent of the areas are relatively small for both methods, but the median for semi-supervised learning is slightly higher than the median using fully-supervised learning, which indicates that the fully-supervised results contain slightly more small areas apart from a flame after applying the morphological closing operation. Also the values until the 75th percentile and the maximum of the boxplots shown are different, where the fully-supervised values are smaller than the semi-supervised ones which are a little closer to the distribution of the ground truth values. As the number and sizes of connected components are only computed for the test results of the first training data split, these results are not statistically significant but might explain better a possible difference between semi-supervised and fully-supervised segmentation results apart from the slight improvement of 2% for mIoU in the quantitative results. For completeness, the mIoU results after the morphological closing operation are slightly better for semi-supervised learning with 88.7% and slightly worse for fully-supervised learning with 85.6%.

	Number of connected components for fire		
	Ground truth	**Fully-supervised**	**Semi-supervised**
Without morphological closing	438	3691	3196
After morphological closing		705	591

Table 5.3: The numbers of connected components found in the ground truth segmentation maps and after fully-supervised and semi-supervised learning. Results shown for a 12.5% labeled-ratio of the fire dataset without and with morphological closing as post-processing step.



(a)



(b)

Figure 5.7: Boxplots for area sizes of fire comparing the ground truth with fully-supervised and semi-supervised learning with a labeled-ratio of 12.5%. Subfigure 5.7a shows the results without a morphological closing operation applied to the segmented areas and subfigure 5.7b shows the results after a morphological closing.

5.2.2 Different labeled-ratios

The results for mIoU, precision, recall and F1-score for fire compared at different labeled-ratios are shown in Figure 5.8 . Similar to the results of Mittal et al. [MTB19], the positive effect of semi-supervised learning to the mIoU is more significant when the labeled-ratio is low. Noticeable is that with semi-supervised learning the precision is more likely to be higher and the recall is more likely to be lower than for fully-supervised learning, whereas their trade-off, the f1-score, is very similar. This indicates that, in this case, the semi-supervised models are better at predicting less false-positives while the fully-supervised models are better at finding more true-positives. This is relevant, if the significance of one of those metrics is more important than the other. In the case of fire segmentation used in real-time fire fighting situations, both metrics are in general equally important and it can not be concluded, that one method is more suitable than the other for this use case.

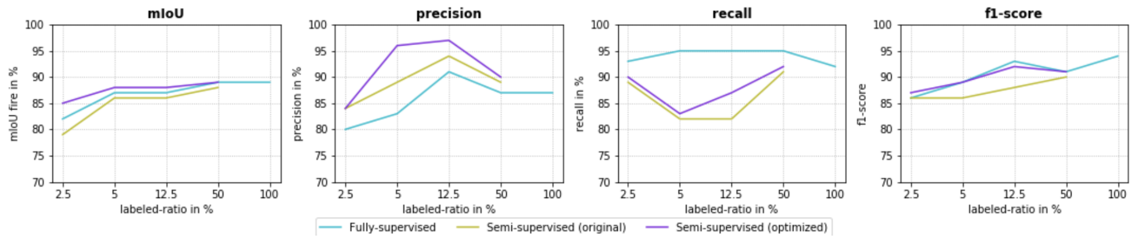


Figure 5.8: Fire mIoU results comparing fully-supervised learning, semi-supervised learning with original hyperparameter settings from Mittal et al. [MTB19] and semi-supervised learning with an optimized starting point at 1000 training iterations for different labeled-ratios.

5.2.3 Results for training with full size datasets

Using the full fire dataset with all available labeled images for fully-supervised learning and additionally the collected unlabeled images for semi-supervised learning, the following results in Table 5.9 were obtained. The differences between semi-supervised and fully-supervised learning are very small in this setting with respect to the evaluation metrics. A possible reason could be that the images used for testing belong to the Corsican dataset, which has a much smaller ratio of areal fire images compared to the unlabeled fire dataset. This is why it could be interesting to test whether the semi-supervised model is better at segmenting aerial fire images compared to the fully-supervised model. To do so, a test dataset with enough labeled aerial fire images would be required.

Similar to the previous results using the Corsican sanity check dataset, a later start of semi-supervised learning shows an improvement compared to the original settings applied by Mittal et al. [MTB19] for bigger benchmark datasets. Also the relations of the numbers and sizes of connected components for semi-supervised and fully-supervised learning in Table 5.4 and Figure 5.13 show a similar behaviour as in the previous experiment with a labeled-ratio of 12.5% .

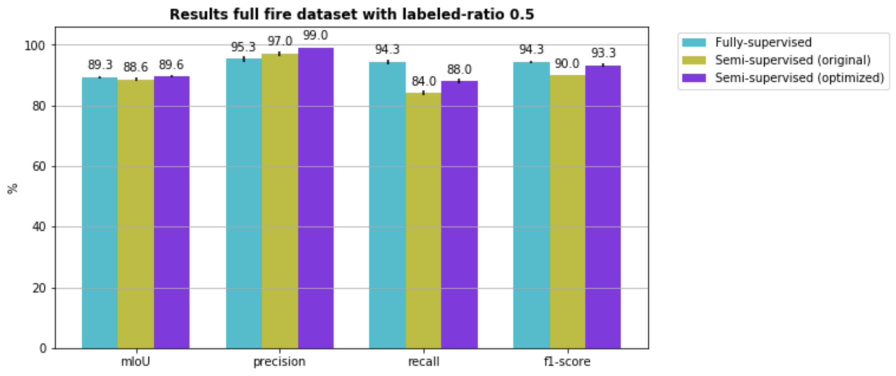
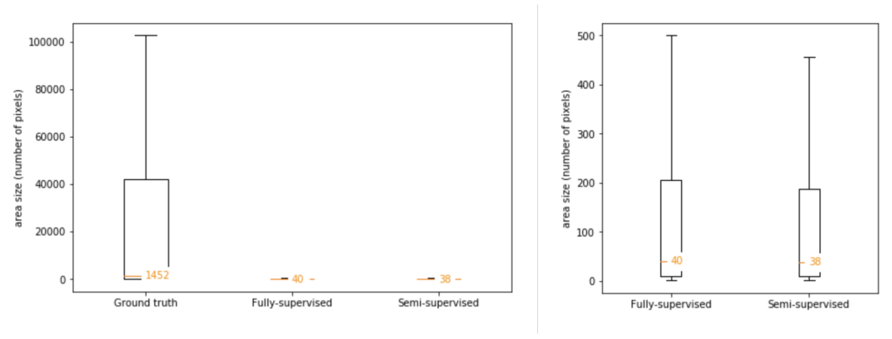


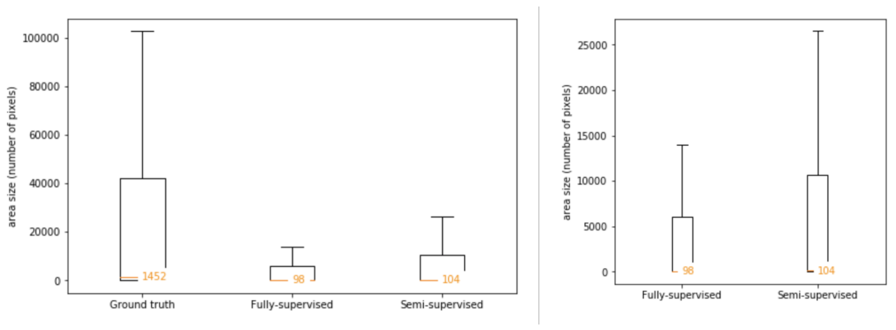
Figure 5.9: Fire results for a 50% labeled-ratio using the full fire dataset averaged over 3 random data splits. The black lines in top of the bars denote the standard deviation.

	Number of connected components for fire		
	Ground truth	**Fully-supervised**	**Semi-supervised**
Without morphological closing	544	2188	2156
After morphological closing		793	721

Table 5.4: The numbers of connected components found in the ground truth segmentation maps and after fully-supervised and semi-supervised learning. Results shown for a 50% labeled-ratio of the full fire dataset without and with morphological closing as a post-processing step.



(a)



(b)

Figure 5.10: Boxplots for area sizes of fire comparing the ground truth with fully-supervised and semi-supervised learning with a labeled-ratio of 50% using the full labeled fire dataset and unlabeled images from additional sources. Results are shown for the first training data split. Subfigure 5.10a shows the results without a morphological closing operation applied to the segmented areas and subfigure 5.10b shows the results after a morphological closing.

Figures 5.11 and 5.12 show examples of segmentations when the models were trained with the full size fire dataset. More examples can be found in Appendix B. A common problem is the confusion of fire with bright yellow, orange or red objects close to a flame.

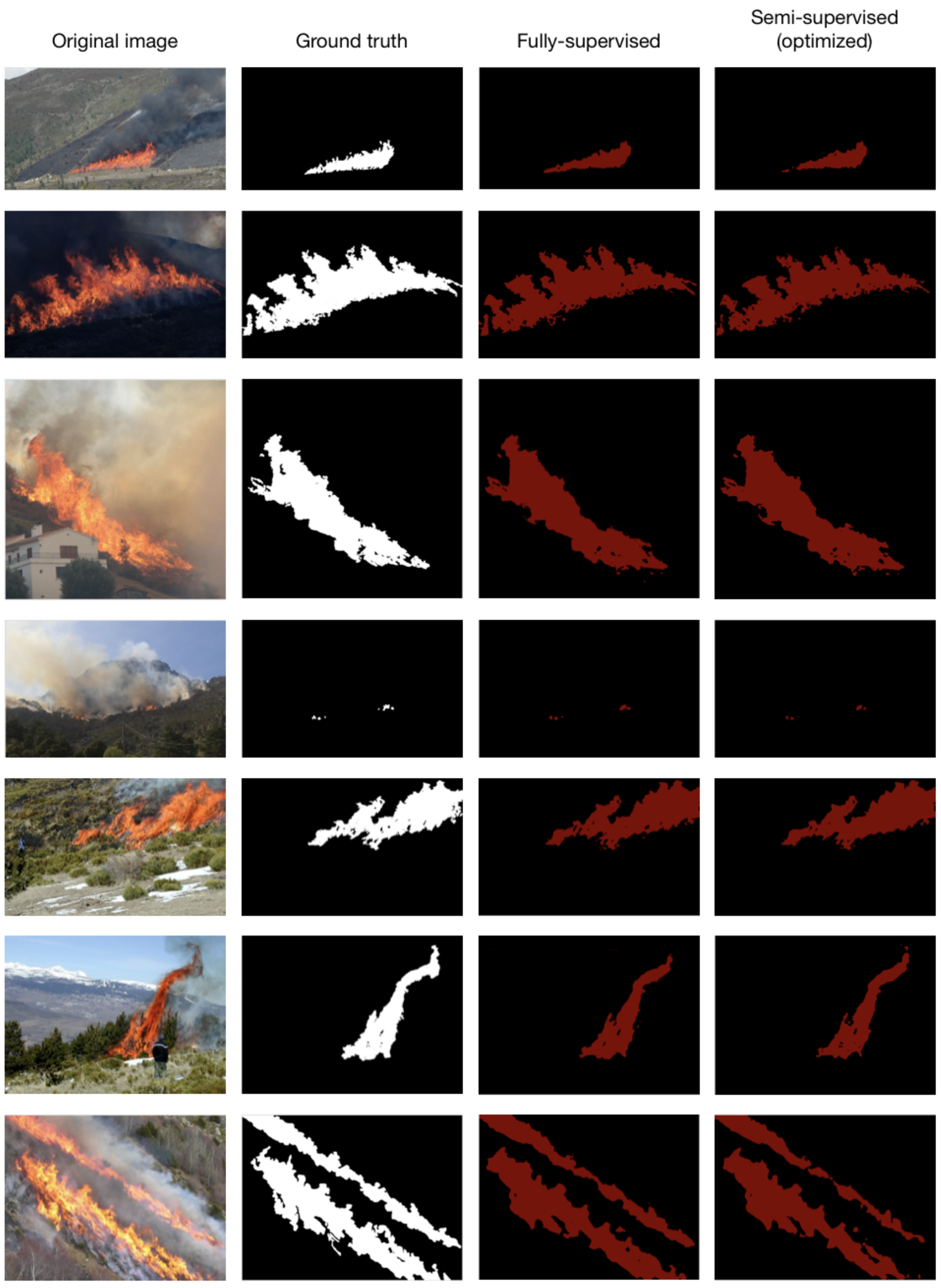


Figure 5.11: Example segmentations for fire using the full size dataset.

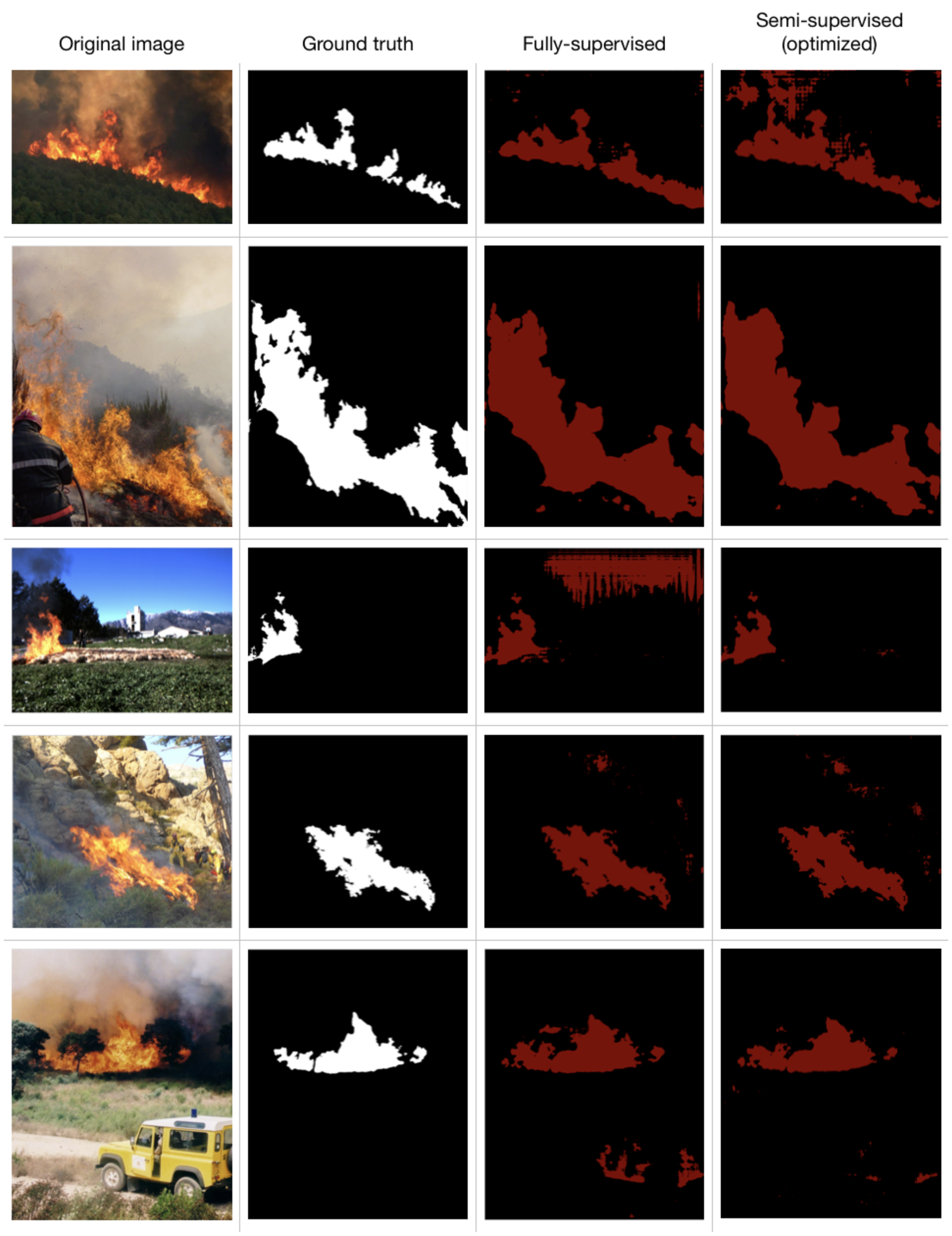


Figure 5.12: Example failure modes for fire using the full size dataset.

For the segmentation of smoke, the following results in Figure 5.13 were obtained. Here, the improvement from fully-supervised learning only to semi-supervised learning with additional images is a bit higher compared to the results with the full size fire dataset. This might be due to the higher portion of aerial images in the labeled smoke dataset, which is used for testing. Through the additional unlabeled training images, which contain mostly aerial perspectives of wildfire smoke, the semi-supervised model has seen more different images with this perspective which could explain the slightly better results of 1.2% mIoU and 1.6% F1-score.

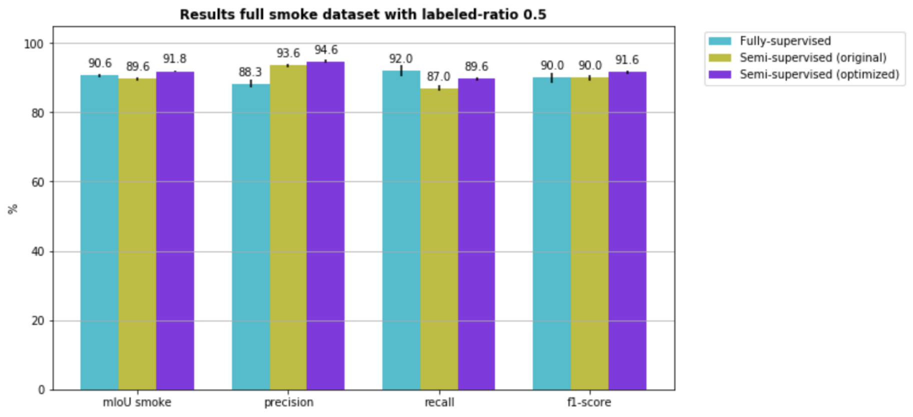
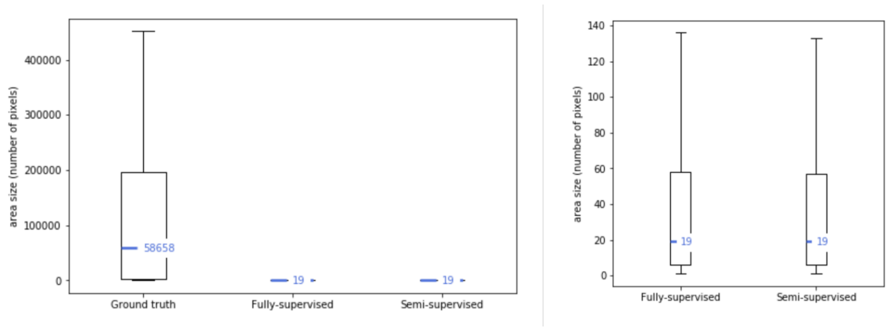


Figure 5.13: Smoke results for a 50% labeled-ratio using the full smoke dataset averaged over 3 random data splits. The black lines on top of the bars denote the standard deviation.

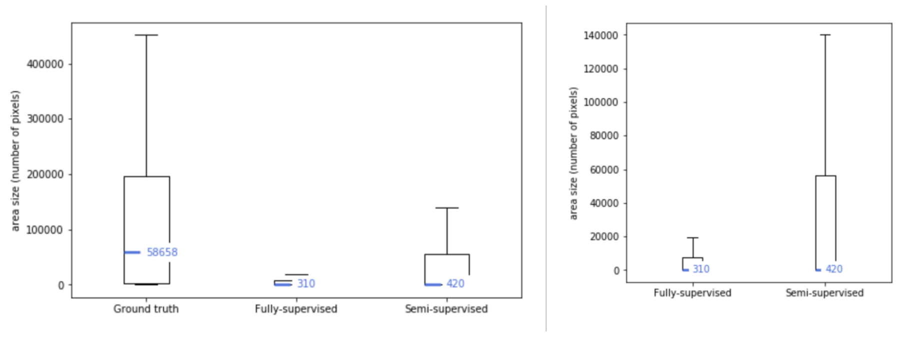
In Figure 5.14 and Table 5.5 the relation of sizes and numbers of connected components of the first training data split show a similar behaviour to the two other results for the Corsican fire sanity check dataset and the full size fire dataset.

	Number of connected components for smoke		
	Ground truth	**Fully-supervised**	**Semi-supervised**
Without morphological closing	151	1868	1462
After morphological closing		294	287

Table 5.5: The numbers of connected components found in the ground truth segmentation maps and after fully-supervised and semi-supervised learning. Results shown for the 50% labeled-ratio of the full smoke dataset without and with morphological closing as post-processing step.



(a)



(b)

Figure 5.14: Boxplots for area sizes of smoke comparing the ground truth with fully-supervised and semi-supervised learning with a labeled-ratio of 50%. Subfigure 5.14a shows the results without a morphological closing operation applied to the segmented areas and subfigure 5.14b shows the results after a morphological closing.

The following figures show some segmentation examples for smoke. In Figure 5.16 examples of failure modes that were observed are depicted. Problems arise for example, when there are objects behind semi transparent smoke. Also, small smoke areas are sometimes not detected and greyish objects like stones or dry grass are confused as smoke.

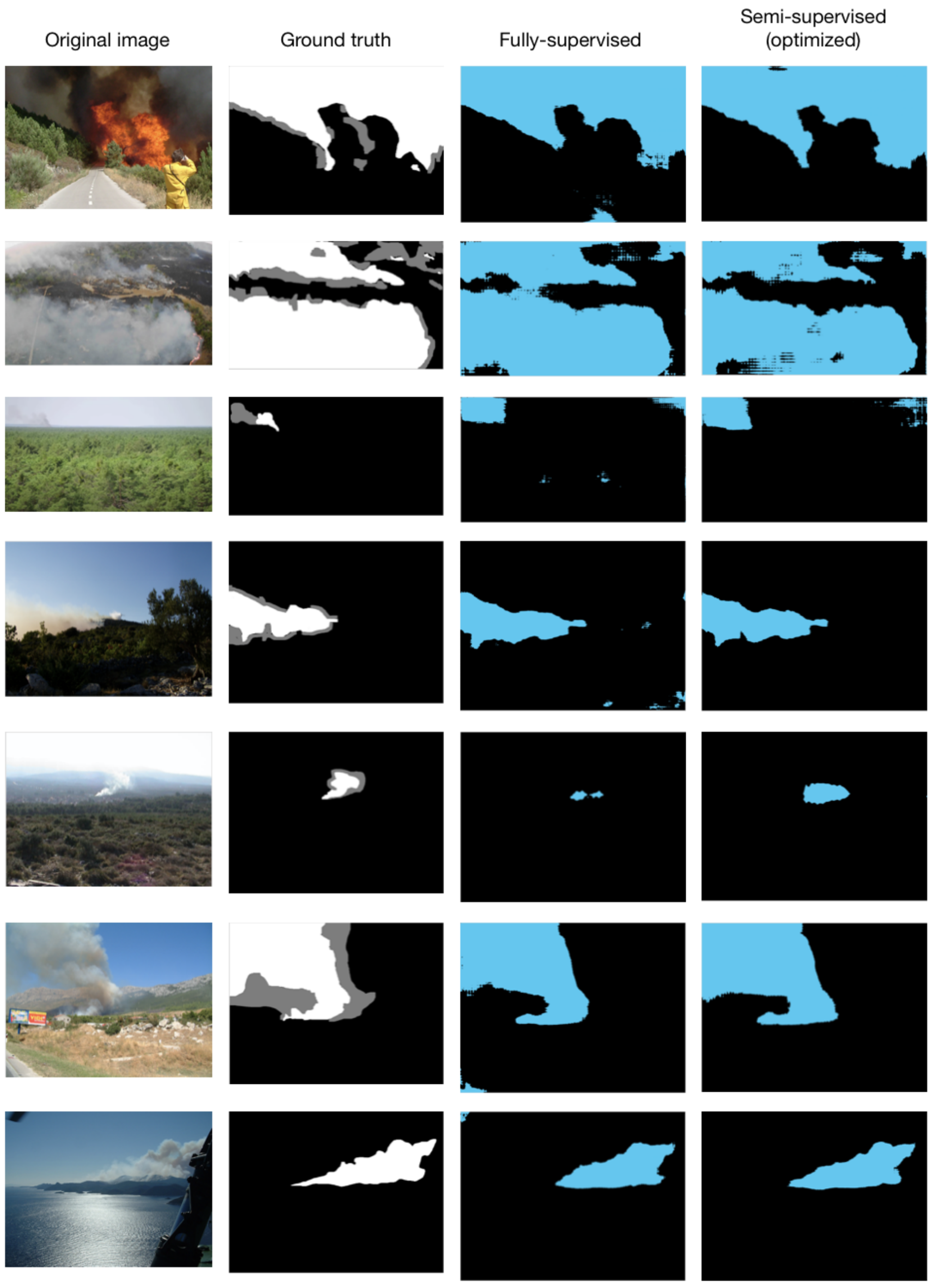


Figure 5.15: Example segmentations for smoke.

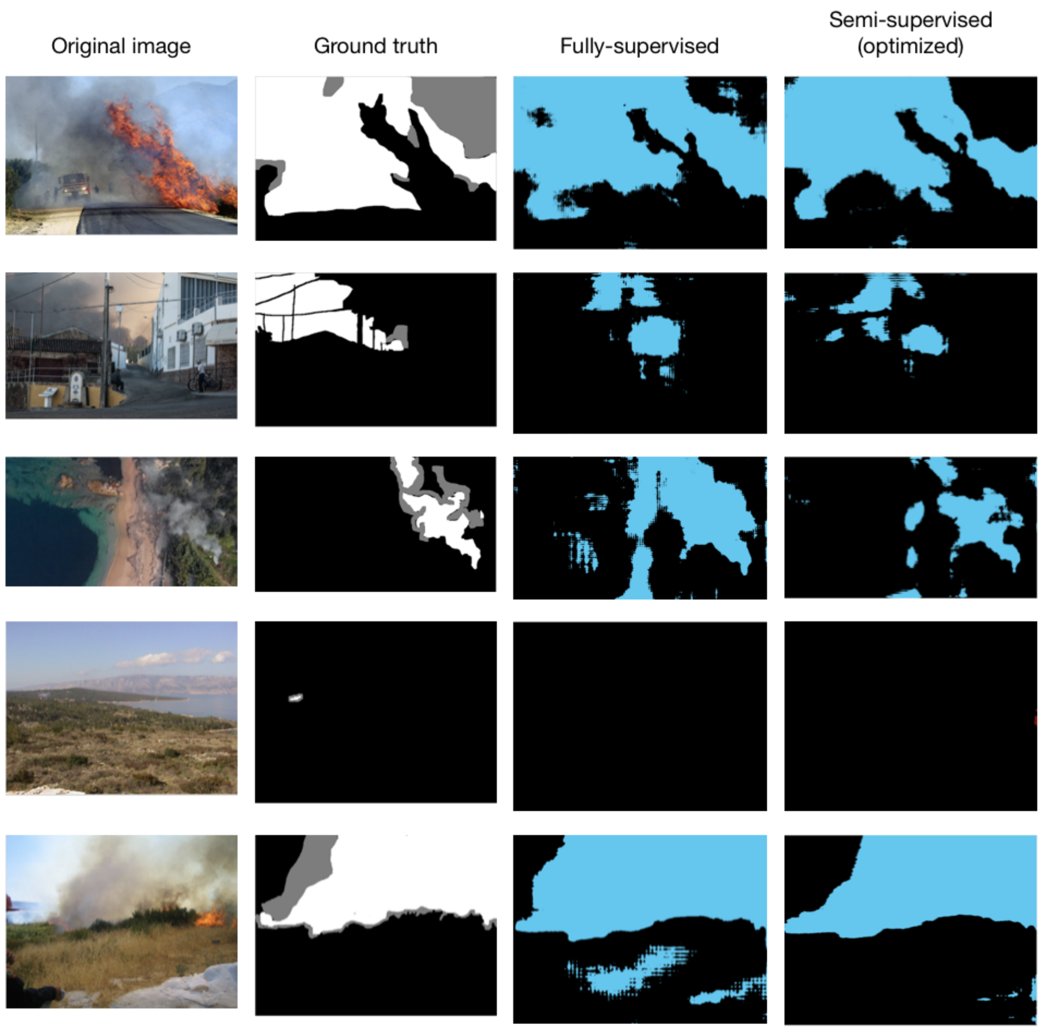


Figure 5.16: Example failure modes for smoke.

Summarizing the results of the previous experiments, the applied semi-supervised GAN has turned out to be very sensitive to the hyperparameters tested. Compared to being used for the training of big benchmark datasets, the training with smaller, specific datasets like the ones for smoke and fire, seems to require parameter settings that stronger balance out the abilities of the discriminator and generator. A strategy that worked well with both datasets was a short, fully-supervised pre-training of the discriminator and generator until 1000 iterations, which lead to a slight improvement of mIoU compared to fully-supervised learning only. Considering the qualitative segmentation results, there seems to be a tendency of less predicted false positive areas apart from a flame when using semi-supervised learning. Common failure modes for fire segmentation are reflections of fire, the sun behind smoke and bright yellow, orange or red objects close to an actual flame. For the segmentation of smoke, small smoke areas are problematic, as they sometimes do not get detected or objects with a similar colour like stones or dry grass are misclassified as smoke. In general, the advantages of semi-supervised learning have shown to be more prominent when using low labeled-ratios, even for the small smoke and fire datasets used in this work.

Chapter 6

Conclusion

The objective of this work was to evaluate the potential benefits of applying a semi-supervised GAN for semantic segmentation using only small labeled training datasets together with additional unlabeled images of smoke and fire. As GANs are known to be hard to train and sensitive to hyperparameters, different hyperparameter settings were tested when applied to the GAN proposed by Mittal et al. [MTB19]. For the same purpose of more training stability and establishing a balance between the discriminator and generator, a short, fully-supervised pre-training was tested as well. Some changes in hyperparameter settings and the fully-supervised pre-training yielded a slight improvement of the mIoU for fire over the fully-supervised baseline and also over the original hyperparameter settings by Mittal et al., when training with the Corsican fire dataset.

Based on that, the fully-supervised pre-training was chosen to be applied as an optimized semi-supervised GAN version to the training with different labeled-ratios of the Corsican fire dataset. Subsequently it was applied to slightly bigger fire and smoke datasets using different sources of training images. For all experiments, the optimized, semi-supervised method was able to achieve a small improvement considering the quantitative metrics, whereas the improvement when training with very small labeled-ratios was more prominent. The results obtained by the experiments indicate that a semi-supervised GAN can be beneficial for the task of smoke and fire segmentation, especially when there is not much labeled training data, but a larger amount of unlabeled data, available.

A possible advantage of semi-supervised learning regarding the segmentation of smoke and fire regions from aerial perspectives could be that available fire or smoke datasets containing images from a ground perspective could still be used as a basis for fully-supervised learning which then can be refined using unlabeled images from an aerial perspective.

Finding the best hyperparameter settings for the GAN, which work well with a different dataset, required an extensive hyperparameter search, which is one of the downsides of using a GAN. Therefore, the short, fully-supervised pre-training could be a simple option for more training stability, as this method worked well for the smoke as well as the fire dataset used in this work. Another approach that could lead to better training stability is the use of a Wasserstein GAN, which was not tested in this work. However, Wasserstein GANs have shown to be superior in many cases, especially the recent version of a Wasserstein GAN with gradient penalty [Gul17].

Considering possible improvements of the segmentation network used in the implementation of this work, a more recent version of DeepLab, DeepLabv3+ [Che18], could be beneficial, as it uses more advanced down- and upsampling techniques. This is especially important for the segmentation of small smoke areas, as these sometimes do not get detected with the network applied, which could be caused by information loss within the layers of the network.

In the future, the Firefront project aims to acquire a wildfire dataset of aerial imagery containing rgb and infrared data in images and video sequences. This kind of data could lead to an improvement in the accuracy of segmentations, as false positive segmentations can be prevented more easily when additional information like the temperature and the spatio-temporal behaviour of objects is available. It might also be beneficial to combine different approaches that were and are investigated within the Firefront project to find a suitable solution for smoke and fire segmentation.

Appendix A

Hyperparameter combinations

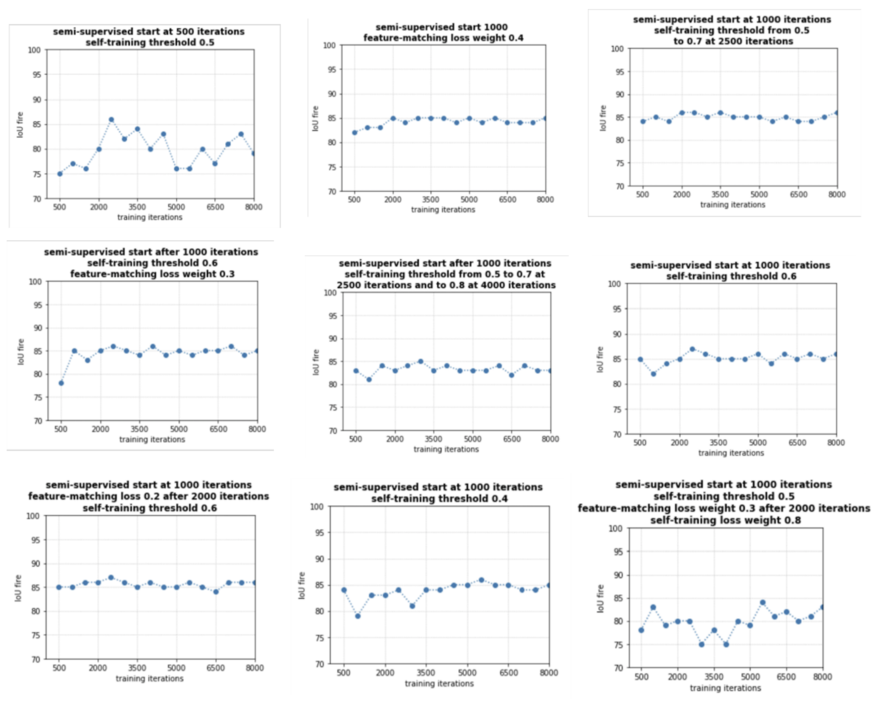


Figure A.1: The hyperparameter combinations tested for the Corsican fire dataset which did not result in further improvements of the mIoU of fire.

Appendix B

Additional qualitative results

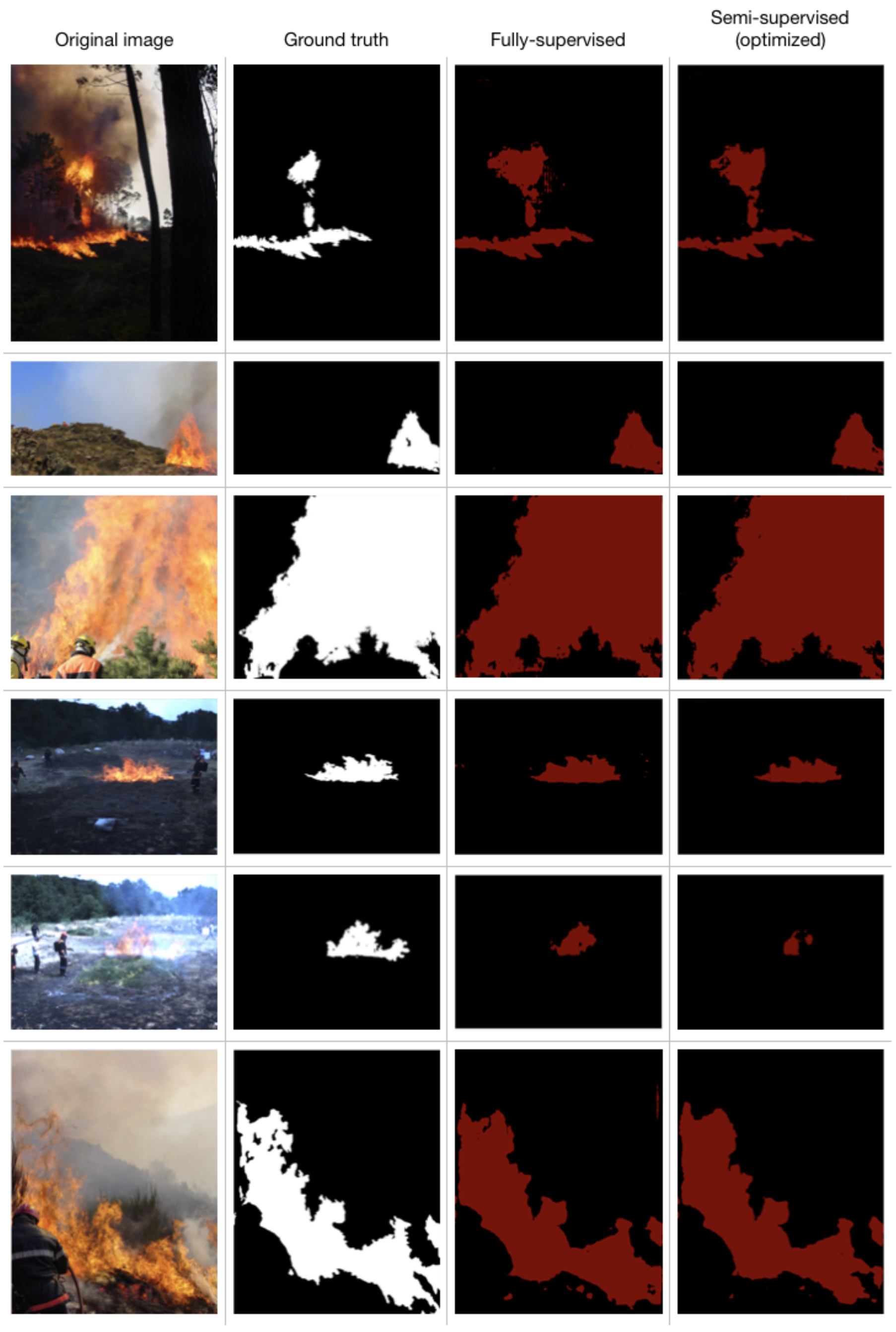


Figure B.1: Example segmentations for fire after using the full size dataset for training.

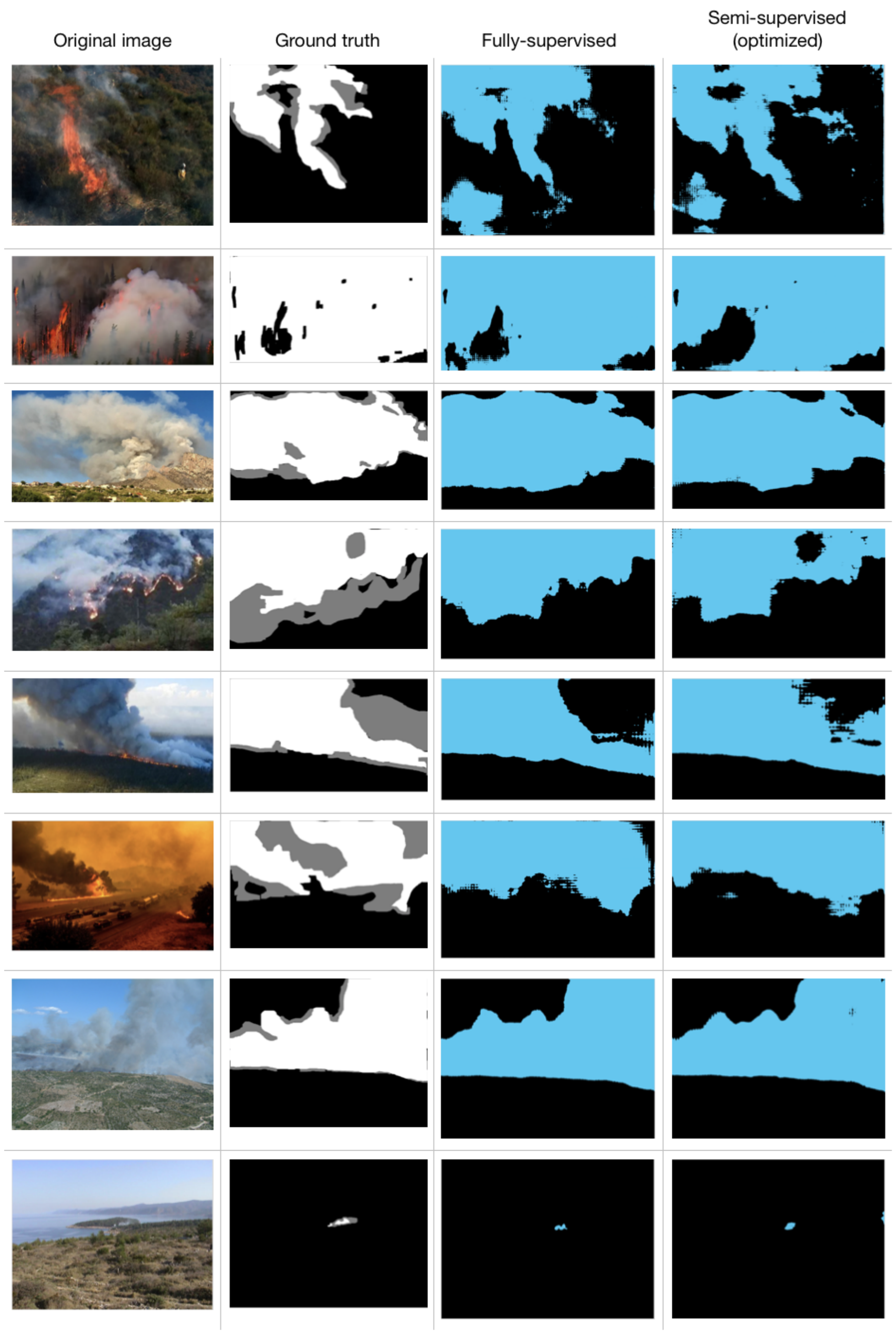


Figure B.2: Example segmentations for smoke after using the full size dataset for training.

Bibliography

[Cnn] *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (visited on 02/27/2021) (cited on pages 6, 7).

[Aga19] Abien Fred Agarap. "Deep Learning using Rectified Linear Units (ReLU)". In: *arXiv:1803.08375 [cs, stat]* (Feb. 7, 2019). arXiv: 1803.08375. URL: http://arxiv.org/abs/1803.08375 (visited on 03/06/2021) (cited on page 7).

[ANS19] Andr\&eacute Araujo, Wade Norris, and Jack Sim. "Computing Receptive Fields of Convolutional Neural Networks". In: *Distill* 4.11 (Nov. 4, 2019), e21. ISSN: 2476-0757. DOI: 10.23915/distill.00021. URL: https://distill.pub/2019/computing-receptive-fields (visited on 03/02/2021) (cited on page 9).

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein GAN". In: *arXiv:1701.07875 [cs, stat]* (Dec. 6, 2017). arXiv: 1701.07875. URL: http://arxiv.org/abs/1701.07875 (visited on 02/17/2021) (cited on page 13).

[BH18] Mark Beighley and Albert C. Hyde. *Portugal Wildfire Management in a New Era Assessing Fire Risks, Resources and Reforms*. Lisbon, Portugal: Instituto Superior de Agronomia, Feb. 2018, page 52. URL: https://tinyurl.com/wzk4lql (cited on pages 1, 2).

[Cai20] Likun Cai et al. "Utilizing Amari-Alpha Divergence to Stabilize the Training of Generative Adversarial Networks". In: *Entropy* 22 (Apr. 2020), page 410. DOI: 10.3390/e22040410 (cited on pages 11–13).

[Che17] Liang-Chieh Chen et al. "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs". In: *arXiv:1606.00915 [cs]* (May 11, 2017). arXiv: 1606.00915. URL: http://arxiv.org/abs/1606.00915 (visited on 03/03/2021) (cited on pages 16, 27, 28).

[Che18] Liang-Chieh Chen et al. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation". In: *arXiv:1802.02611 [cs]* (Aug. 22, 2018). arXiv: 1802.02611. URL: http://arxiv.org/abs/1802.02611 (visited on 03/06/2021) (cited on pages 25, 48).

[Noaa] *Convolutional neural networks: an overview and application in radiology | Insights into Imaging | Full Text*. URL: https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9 (visited on 02/21/2021) (cited on page 9).

[Cor16] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cited on pages 14, 18, 20, 31).

[Noab] *CORSICAN FIRE DATABASE | Projet Feux | Università di Corsica Pasquale Paoli | Université de Corse Pasquale Paoli*. URL: https://feuxdeforet.universita.corsica/article.php?id_art=2133&id_rub=572&id_menu=0&id_cat=0&id_site=33&lang=en (visited on 03/03/2021) (cited on page 3).

[Den09] J. Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pages 248–255. DOI: 10.1109/CVPR.2009.5206848 (cited on page 16).

[Den09] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009 (cited on page 25).

[Eve09] M. Everingham et al. "The Pascal Visual Object Classes (VOC) Challenge". In: *International Journal of Computer Vision* 88 (2009), pages 303–338 (cited on pages 18, 20).

[Noac] *Firefront*. URL: http://web.tecnico.ulisboa.pt/~ist178247/firefront/wordpress/ (visited on 02/27/2021) (cited on page 4).

[Goo14] Ian J. Goodfellow et al. "Generative Adversarial Networks". In: *arXiv:1406.2661 [cs, stat]* (June 10, 2014). arXiv: 1406.2661. URL: http://arxiv.org/abs/1406.2661 (visited on 03/06/2021) (cited on pages 11, 29).

[Gul17] Ishaan Gulrajani et al. “Improved Training of Wasserstein GANs”. In: *arXiv:1704.00028 [cs, stat]* (Dec. 25, 2017). arXiv: 1704.00028. URL: http://arxiv.org/abs/1704.00028 (visited on 02/12/2021) (cited on pages 15, 48).

[He16] K. He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). ISSN: 1063-6919. June 2016, pages 770–778. DOI: 10.1109/CVPR.2016.90 (cited on pages 6, 9, 16, 25, 26).

[Hua18] Gao Huang et al. “Densely Connected Convolutional Networks”. In: *arXiv:1608.06993 [cs]* (Jan. 28, 2018). arXiv: 1608.06993. URL: http://arxiv.org/abs/1608.06993 (visited on 03/02/2021) (cited on page 9).

[Hun18] Wei-Chih Hung et al. “Adversarial Learning for Semi-Supervised Semantic Segmentation”. In: *arXiv:1802.07934 [cs]* (July 24, 2018). arXiv: 1802.07934. URL: http://arxiv.org/abs/1802.07934 (visited on 02/28/2021) (cited on pages 14–20, 22, 33, 39).

[Noad] *Image database*. URL: http://wildfire.fesb.hr/index.php?option=com_content&amp;view=article&amp;id=49&amp;Itemid=54 (visited on 03/04/2021) (cited on page 23).

[kF16] Amine ben khalifa and Hichem Frigui. “Multiple Instance Fuzzy Inference Neural Networks”. In: (Oct. 2016) (cited on page 10).

[KK12] Philipp Krähenbühl and Vladlen Koltun. “Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials”. In: *arXiv:1210.5644 [cs]* (Oct. 20, 2012). arXiv: 1210.5644. URL: http://arxiv.org/abs/1210.5644 (visited on 03/02/2021) (cited on pages 8, 15).

[Lec98] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (Nov. 1998). Conference Name: Proceedings of the IEEE, pages 2278–2324. ISSN: 1558-2256. DOI: 10.1109/5.726791 (cited on page 6).

[Lin15] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV] (cited on page 16).

[Fea] *MIT 6.S191: Introduction to Deep Learning*. URL: http://introtodeeplearning.com (visited on 02/27/2021) (cited on page 6).

[MTB19] Sudhanshu Mittal, Maxim Tatarchenko, and Thomas Brox. "Semi-Supervised Semantic Segmentation with High- and Low-level Consistency". In: *arXiv:1908.05724 [cs]* (Aug. 15, 2019). arXiv: 1908.05724. URL: http://arxiv.org/abs/1908.05724 (visited on 03/03/2021) (cited on pages 3, 14, 19, 20, 22, 24, 25, 28, 31–34, 37–39, 42, 43, 47).

[Nef] Thomas Neff et al. "Generative Adversarial Networks to Synthetically Augment Data for Deep Learning based Image Segmentation". In: (). ISBN: 9783851256031 Publisher: Verlag der Technischen Universität Graz. DOI: 10.3217/978-3-85125-603-1-07. URL: https://openlib.tugraz.at/download.php?id=5b3619809d758&location=medra (visited on 02/19/2021) (cited on pages 14, 15).

[NHH15] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. "Learning Deconvolution Network for Semantic Segmentation". In: *arXiv:1505.04366 [cs]* (May 17, 2015). arXiv: 1505.04366. URL: http://arxiv.org/abs/1505.04366 (visited on 03/02/2021) (cited on pages 8, 9).

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *arXiv:1505.04597 [cs]* (May 18, 2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597 (visited on 03/02/2021) (cited on pages 9, 14).

[Rot17] Kevin Roth et al. "Stabilizing Training of Generative Adversarial Networks through Regularization". In: *arXiv:1705.09367 [cs, stat]* (Nov. 7, 2017). arXiv: 1705.09367. URL: http://arxiv.org/abs/1705.09367 (visited on 03/08/2021) (cited on page 13).

[Rud17] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv:1609.04747 [cs]* (June 15, 2017). arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747 (visited on 03/08/2021) (cited on page 9).

[Sal16] Tim Salimans et al. "Improved Techniques for Training GANs". In: *arXiv:1606.03498 [cs]* (June 10, 2016). version: 1. arXiv: 1606.03498. URL: http://arxiv.org/abs/1606.03498 (visited on 02/28/2021) (cited on pages 13, 19, 31).

[SK19] Connor Shorten and Taghi M Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning". In: *Journal of Big Data* 6.1 (2019), page 60. ISSN:

2196-1115. DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0 (cited on page 10).

[SSS17] Nasim Souly, Concetto Spampinato, and Mubarak Shah. "Semi and Weakly Supervised Semantic Segmentation Using Generative Adversarial Network". In: *arXiv:1703.09695 [cs]* (Mar. 28, 2017). arXiv: 1703.09695. URL: http://arxiv.org/abs/1703.09695 (visited on 03/03/2021) (cited on page 18).

[sud21] sud0301. *sud0301/semisup-semseg*. original-date: 2019-08-11T19:07:01Z. Mar. 2, 2021. URL: https://github.com/sud0301/semisup-semseg (visited on 03/05/2021) (cited on page 25).

[Tou17] Tom Toulouse et al. "Computer vision for wildfire research: An evolving image dataset for processing and analysis". In: *Fire Safety Journal* 92 (July 2017), pages 188–194. DOI: 10.1016/j.firesaf.2017.06.012 (cited on page 22).

[Noae] *V7 Darwin - Automated Image Annotation*. URL: https://www.v7labs.com/darwin (visited on 03/07/2021) (cited on page 24).

[Xu15] Bing Xu et al. "Empirical Evaluation of Rectified Activations in Convolutional Network". In: *arXiv:1505.00853 [cs, stat]* (Nov. 27, 2015). arXiv: 1505.00853. URL: http://arxiv.org/abs/1505.00853 (visited on 03/08/2021) (cited on page 28).

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Berlin, den 08.03.2021

Lisa Kuhlmann

Lisa Kuhlmann