Robustness of Concept-Based Models

Master thesis by Christian Stoll (Student ID: 2626398)
Date of submission: October 21, 2021

1. Review: Prof. Dr. Kristian Kersting
2. Review: M. Sc. Wolfgang Stammer
Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Computer Science
Department

Artificial Intelligence and
Machine Learning Lab

Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 und §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Christian Stoll (Student ID: 2626398), die vorliegende Masterarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß §23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 21. Oktober 2021

Christian Stoll (Student ID: 2626398)

Abstract

State-of-the-art computer vision models perform badly, when they are tasked to predict learned classes from out-of-distribution data or data perturbed by adversarial attacks. Both cases can appear in a day-to-day usage of such models and mostly arise unintentionally but can still be harmful. We investigate and evaluate multiple models, some of which allow for richer human-model interaction, by testing different models using out-of-distribution data and 4 adversarial attacks: FGSM, BIM, PGD and SparseFool. We propose and test a model which combines a high-level object detection and instance segmentation model called Mask R-CNN with a Concept Bottleneck model, which we call Mask Bottleneck model. We show the results of both, out-of-distribution attacks, and adversarial attacks on different parts of the models, all trained on the CUB-200-2011 dataset. The new model is inherently more robust, on these tasks, outperforming baseline, and other comparison models in most cases on classification accuracy and relative loss of accuracy, when being attacked. The Code is available at: `https://github.com/ChristianStoll/Robustness-of-Concept-Based-Models`.

Contents

List of Figures

List of Tables

1 Introduction

Current research in computer vision focused on the development of new and better models for tasks such as classification of images, object detection and segmentation. Lately this research moved towards explainable AI (Stammer, Schramowski, and Kersting 2020) and AI that is developed towards making decisions by using a human-like approach at feature detection and extraction (Suglia et al. 2020; T. Kim, S. Kim, and Y. Bengio 2020; Yi et al. 2018). Scientists in this area are strongly interested in the question about how and why the models make a certain decision, how to interpret their proceeding and how to improve their decision making.
But what happens if such a model faces a situation in the area of its' task, it has not faced in its training yet? What happens if there are intentional and possibly malicious attacks on the model or the model's input? While the possibilities, opportunities and use cases for the usage of AI and computer vision are largely increasing, answers to these questions get increasingly important. More research is carried out in this area, since both these questions imply that there is a large possibility and danger that humans or animals get hurt, when such situations arise due to the increasingly wide applications of machine learning models. A few of such applications are autonomous driving and assisted surgeries, but it might also include all stages of the production of goods and medicine, which are increasingly controlled by AI and machine learning and many other areas. These models and partial steps of processing data oftentimes at least partially relies on computer vision models for a visual analysis. Additionally, these steps of processing data often also rely on AI and humans deciding and working hand in hand to achieve a common goal.
In this thesis we will try to fill such a gap by investigating the behavior of recently proposed models under the attack of out-of-distribution data and adversarial attacks to find models, which are intrinsically more robust to one or both of these two tasks.
These models will include a model, which predicts human-based concepts as an intermediate step (Koh et al. 2020). Additional models will be object centric models, such as Scouter (Li et al. 2020) and as part of a larger pipeline Mask R-CNN (He, Gkioxari, et al. 2017). We will also investigate Vision Transformer (Dosovitskiy et al. 2020) as a model, which is unique among the models we used, since it does not use any convolutions.
We tested these models on two tasks. First an out-of-distribution attack, which used the TravelingBirds dataset from Koh et al. (2020). The second task included several adversarial attacks, such as FGSM (Goodfellow, Shlens, and Szegedy 2015), BIM (Kurakin, Goodfellow, and S. Bengio 2016), PGD (Madry et al. 2019) and SparseFool (Modas, Moosavi-Dezfooli, and Frossard 2018).
By using these tests and during this work we want to answer multiple larger questions, which all contain smaller more detailed questions.
The first large question is that we want to know, how these models perform on out-of-distribution data. This includes the related, but subordinate questions about the influence of different backgrounds, some of which are more uniform while some others are more complex. Another subordinate question is to evaluate if there is a model, which is able to independently remove parts of the background and if this removal of the background influences the following decisions.
The second large question is, how these models behave, when being attacked by adversarial attacks.

Subordinate questions to this very open question are questions on the behavior of different strengths of attacks, different kinds of attacks and if there is a difference for the models, if the full input is perturbed or if only parts of the input are perturbed. The final large question we want to answer in this work is, if any one of the models we tested and propose, do contain an intrinsic robustness to both of these attacks and to then evaluate, where this intrinsic robustness is coming from.

- to carry out adversarial and out-of-distribution attacks on several computer vision models, some of which can be assigned to AI, whose decisions are more easily interpretable and influenceable through human intervention,
- investigating these models and evaluate, whether there is an intrinsic robustness of these models to either one or both of these attacks,
- propose and investigate a model, which uses a combination of Slot Attention and Set Transformer together with human like concepts to carry out its predictions,
- propose and investigate a model, which performs high level object detection before a lower-level classification of the images
- extensive evaluation and discussion of the results.

We will proceed as follows. Firstly, we will present related work to our topic and introduce definitions and abbreviations we will use throughout this thesis. Afterwards, we will introduce different datasets, models and the experimental attack setups. We will then describe and evaluate the experimental results and eventually finish this thesis with an extensive discussion of the results including an answer to our research question and a conclusion.

2 Related Work

2.1 Robustness in general

The robustness of machine learning models is an important area of research. It involves research for many tasks not only in computer vision and deep learning, but also for natural language processing, and many other fields. One example for natural language processing would be different dialects of languages as well as metaphors (Stowe, Ribeiro, and Gurevych 2020) and multiword expressions (Rohanian et al. 2020), which most often include ambiguity in the data (Hirschberg and Manning 2015). In computer vision, the robustness of models is often required in many directions. When we take the field of autonomous driving, it is required for the car to get a representation of the environment it is in. Common approaches for this are using object detection or instance segmentation to get a broad view on the environment. In these cases, the environment undergoes large changes during each day. At some time of the day, it is daylight. This means all the environments are bright, many road users are on the streets and sidewalks when talking about cities. At night, there are few people and road users outside and most streets are empty. But we do not only have daylight at nighttime, but we also have representations such as in cities or built-up areas, and we also have large highways or small streets in the nature. All these characteristics have to be represented to get a broad overview of the environment of a car, but these environments do not only change in a smaller local area, but also, when traveling a larger distance for example into a mountain area or when the vegetation changes to due changes in the climate across different countries. To tackle this task, which requires large datasets, Cordts et al. (2016) constructed the Cityscapes dataset.
Additionally, to get a more detailed view, autonomous cars have to detect and reliably classify signs on the street. While these signs remain constant within a certain country, many cars pass the borders of countries every day and each country has different traffic signs. Some of those are closely related for example for large highways, which cross borders within Europe, but others are more differently but no less important for save traveling.
All these factors have to be considered and represented in the training data, but also in the decision making of the models. While there is continuous progress and improvements of these areas, there are still a lot of different factors and individual cases, which might lead to bad decisions of such cars. One example might be, when we now also take the weather into account for example if it rains or just rained and the sun is shining, even then the input to such networks is perturbed.

2.2 Adversarial Attacks

Adversarial attacks are smaller area in machine learning, but a nonetheless important one in computer vision and deep learning for example. This area of research focuses on the question of what happens if the

input to a model is intentionally perturbed to cause faulty decisions.
There are two kinds of attacks, which can be used on the models. Black-box and white-box attacks. Bhambri et al. (2019) and Guo et al. (2019) provide definitions of these two setups. Summarized, white-box attacks are attacks, where the adversary knows everything about the target model, which includes learned weights, parameters and in some cases even the training data and gradient flow of the model. Black-box attacks on the other hand have very limited knowledge of the model, sometimes even without any knowledge of the model the adversary is attacking.
Regarding these two definitions, there are different works regarding adversarial attacks. Starting with the individual development of adversarial attacks, such as FGSM (Goodfellow, Shlens, and Szegedy 2015) and SparseFool (Modas, Moosavi-Dezfooli, and Frossard 2018) which are white-box attacks perturbing different amounts of pixels of an input image (Vargas and Kotyan 2019) and are either iterative, such as BIM (Kurakin, Goodfellow, and S. Bengio 2016) or single step approaches (Goodfellow, Shlens, and Szegedy 2015). These works often include additional research on the adversarial robustness of models (Yu et al. 2020) and possible solutions to find a way to overcome the vulnerability to adversarial attacks by using adversarial training (Madry et al. 2019).
Additionally, there is research in different metrics to evaluate the robustness of neural networks (Buzhinsky, Nerinovsky, and Tripakis 2020; Bastani et al. 2016).
Another part of work is the development of frameworks, which are providing numerous examples of adversarial attacks to use (H. Kim 2020).

2.3 Concept Bottleneck

Recent research showed that models using human-like concepts can be very helpful and supporting when working hand in hand with humans. This research showed, that *Concept Bottleneck models* (Koh et al. 2020) do not only achieve competitive results on classification tasks, but that they additionally enable an easy way of test-time intervention, when human-like concepts are used by the models. This way humans and machine learning models can work on certain tasks together and not only reduce the errors each would make when working individually, but while working together, they can also correct each other and significantly improve their results.

2.4 Object centric models

Object centric models learn, extract and use special features and representations which they later use to solve different tasks. Such features oftentimes differentiate one class of objects from another – or show the similarities of these classes.
One widely used task for object centric models is semantic segmentation, where a model associates a class label to each pixel of an image. Examples for such models are for example ResNet (He, X. Zhang, et al. 2015) and MobileNet (Howard et al. 2019). An additional task to semantic segmentation is object detection and instance segmentation, where the models, not only assign class labels to each pixel, but they also return information about the location of the objects in an image (Ren et al. 2015; He, Gkioxari, et al. 2017). This procedure allows for further usage of the respective parts of an image.
Recent research also showed that the concept of *attention* (Vaswani et al. 2017) can be adapted to be used

for computer vision in several different ways. There is the direct adaptation of a standard transformer model (Vaswani et al. 2017) to computer vision, a so-called Vision Transformer (Dosovitskiy et al. 2020). This adaptation does not use any convolutions, which makes it a rare occurrence regarding deep learning models for computer vision, which most often use convolutions to process the input images.
Another shape of object centric models is the usage of attention to extract representations of each object in an image, without losing the possibility to generalize to unseen compositions. This idea of using attention in object centric models is introduced by Locatello et al. (2020) and it is called slot attention.
Using this idea of slot attention, Li et al. (2020) developed a model, which learns class-specific representations using this kind of attention mechanism. These representations are used to find approving or rejecting explanations for each class to make a final decision.
In some of the research approaches in the area of object detection and semantic segmentation, there is also already research in adversarial attacks on such models. First, there is research in adversarial attacks and the robustness of Vision Transformers (Shao et al. 2021) on large datasets. Secondly, there is research on adversarial attacks on Faster-RCNN (Ren et al. 2015; Xie et al. 2017), a predecessor of Mask R-CNN (He, Gkioxari, et al. 2017) and different semantic segmentation networks, where the research is focused towards the safety and security in autonomous driving (Arnab, Miksik, and Torr 2017; Bär et al. 2021).

3 Definitions

We will use several expressions and abbreviations throughout this thesis and especially in graphs and figures. To give a good overview on them and to improve the readability of this thesis, we will provide the definitions we use in this section.

- **Concept model**: a model, which is taking an image as input to predict human-understandable concepts
- **End model, Independent model, Sequential model**: the second part of a *Concept Bottleneck* model. Depending on the training variant, the second part is either named *independent* or *sequential*
- **Concepts**: predictions of a concept model, concepts are the predictions of characteristics of a bird
- **Attributes**: the annotated labels to the characteristics of a bird
- **Cropbb**: cropping an image around the edges of a predicted bounding box
- **Segbb**: cropping an image around the edges which are extracted from a predicted segmentation mask
- **Apply a segmentation mask to an image**: multiply the given image with a segmentation mask, which in our case is consisting of bird pixels and background pixels
- **Useseg**: a postfix to some models (the ones having *cropbb* or *segbb* in their name) to indicate that these models use images, where a segmentation mask was applied to an image
- **XtoC**: the forward pass of a concept model using an image X as input and returning predicted concepts C
- **CtoY**: the forward pass of a fully connected layer using predicted concepts as input and returning class predictions Y
- **XtoY**: an end-to-end model taking an image X as input and returning class predictions Y. This might also include an intermediate step of predicting concepts, but in this case this intermediate step has no direct influence on our results.
- **ViT**: abbreviation for *Vision Transformer*
- **CB**: abbreviation for *Concept Bottleneck models*, models from Koh et al. (2020).
- **MB Stage 2 adv**: abbreviation for *Mask Bottleneck model, stage 2, adversarial attacks*

4 Datasets

4.1 Caltech-UCSD Birds-200-2011

In this thesis we used the Caltech-UCSD Birds-200-2011 (CUB) dataset (Wah et al. 2011) for most of our experiments. The dataset contains 11,788 images off 200 different bird classes. Additionally, each image in the dataset is annotated with the locations of 15 parts of a bird, 312 binary attributes and one bounding box. For all our experiments we use the information about the binary attributes to predict the concepts of all concept-based models. We use the preprocessed attributes of Koh et al. (2020), who limit the number of binary attributes to 112 for the comparability between different models. These attributes represent features like wing color, beak shape, bill shape, back color and others and they are class-level attributes, which are received by majority voting for each attribute and bird class. For example, if more than 50% of crows have black wings, then all crows must have black wings. In total there are 27 unique attribute classes, which are represented by the 112 binary attributes.
The CUB dataset is split in 5994 train images (51% of the dataset) and 5794 test images (49% of the dataset). This split is kept intact by Koh et al. (2020), but they additionally split the training set into a train and a validation set, with the validation set containing 20% of the original training images. In our experiments and for the models we developed and trained by ourselves, we kept this split into training, validation and test set. For models we retrained or finetuned, the split of the training set into training and validation set is different.

4.2 Places365-Standard

The Places365-Standard dataset is a subset of much larger Places dataset (Zhou et al. 2018). Places365-Standard contains 1.8 million training images of 365 classes. In the validation set there are 50 images per class, while there are 900 images per class in the test set. The places dataset contains images of different scenes. These scenes do not focus on one single or a few objects, but they capture larger areas, such as veterinarian offices, elevator doors or a cafeteria. While we do not use this dataset directly, we use parts of Places365-Standard (Zhou et al. 2018) to achieve a distribution shift in the background of the CUB images. How this distribution shift is achieved will be described in the following paragraph.

4.3 TravelingBirds

Koh et al. (2020) constructed and used the *TravelingBirds* dataset to test the robustness of their models to shifts in the underlying data distribution. The TravelingBirds dataset consists of the three smaller datasets, which we will refer to as *CUB-black*, *CUB-fixed* and *CUB-random*. Here the authors of Koh et al. (2020) modified the background of each image in the CUB dataset by using the provided segmentation masks to cut out the bird and place it on a different background. The colors of the bird are kept intact for all artificially generated images. While in CUB-black, each bird is placed on a black background, the changes of the images in the datasets CUB-fixed and CUB-random are much larger. The birds are placed into images from the places365 dataset (Zhou et al. 2018). In CUB-fixed, each bird class is assigned to one places365 class and the background class of each image of this certain bird species is always the same. In CUB-random the background for each bird is chosen randomly from the full places365 dataset. The split of the dataset for training and testing coming from CUB are kept intact. Koh et al. (2020) state, that TravelingBirds is constructed in a similar manner to the Waterbirds dataset introduced in Sagawa et al. (2019).

5 Models

In this section we will describe the models, we used and compared in this work.

5.1 Inception-v3 Baseline

We use an Inception-v3 model (Szegedy et al. 2015) as our baseline, which was trained on the CUB dataset (Wah et al. 2011) by Koh et al. (2020). This model will be the baseline for most other models, with one exception: Scouter (Li et al. 2020) cannot be trained on all the 200 classes of CUB, therefore we also trained a version of the Inception-v3 model on the first 25 classes of CUB, the same classes, the different Scouter models (Li et al. 2020) were trained on. We will the model trained on 25 classes *Inception-v3-25*. More details about Scouter and why training on 200 classes was not possible will follow later in sections 5.4, 7.2.

5.2 Inception-v3 CUB-black

For one of our settings we were interested, how the baseline model behaves on a dataset which provided only the birds of the CUB dataset (Wah et al. 2011), but without any background information. Therefore, we also trained an Inception-v3 model (Szegedy et al. 2015) on the full CUB-black dataset with all 200 classes from Koh et al. (2020). We will call this model *Inception-v3-black*.

5.3 Concept Bottleneck Models

Koh et al. (2020) introduced several variants of a so called “bottleneck” model, which shall follow a human-like approach at classifying birds of the CUB dataset by using different high-level attributes. The approach to such a model can be seen in figure 5.1. It uses an imput image X to predict human-specified concepts C and then uses the predicted concepts C to predict the output Y. For their baseline, Koh et al. (2020) resized one of the layers of the Inception-v3 (Szegedy et al. 2015) to match the 112 of concepts, which are provided for prediction. The 112 concepts result of the majority voting which Koh et al. (2020) introduced and we described earlier. Due to the majority voting Koh et al. (2020) reduced the number of attributes from 312 annotated attributes in CUB to 112.
Except for the adapted bottleneck baseline, each of Koh et al. (2020)’s models consisted of two parts. A concept model, which is an Inception-v3 model with a resized last layer, and a fully connected layer to predict the bird classes from the concepts. Splitting the model into two parts does have another beneficial

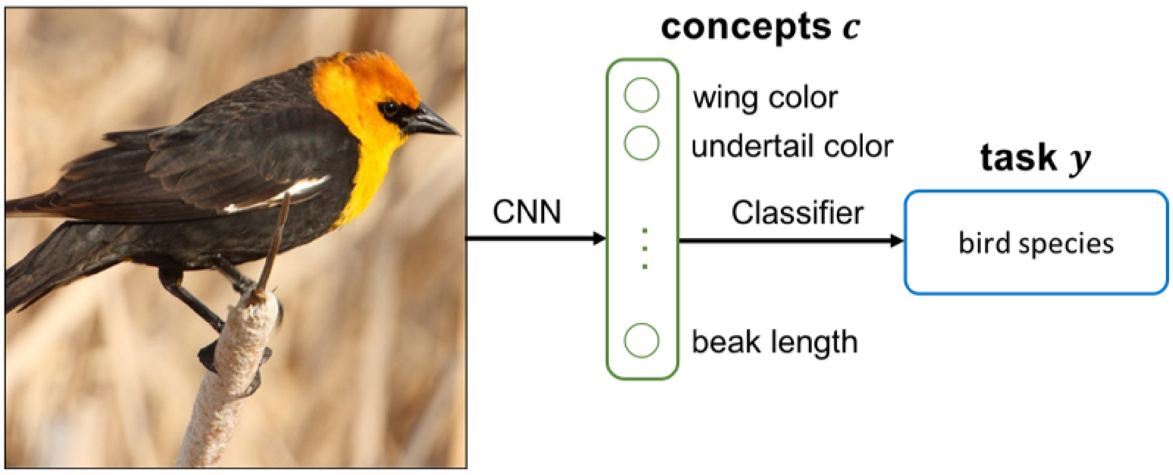

Figure 5.1: The classification pipeline of a Concept Bottleneck model. The first step is to predict a set of human-specified concepts *c*, then use *c* to predict the final output *y*. (Taken and adapted from Koh et al. (2020), figure 1.)

effect for the models: it allows human test-time intervention, which increased the accuracy of the final class prediction significantly if the original concept accuracy is high enough. Although these findings are interesting and important regarding joint operations and decision making between humans and artificial models, we will not examine this part in this thesis and focus on the robustness of the models.
All bottleneck models consisting of the already mentioned to parts are trained in three different ways: an independent, a sequential and a joint version of the model. First Koh et al. (2020) trained a so called *concept model* which represents the first part of the model and is reused for both independent and sequential predictions. For the independent version of the second part, they trained the fully connected layer on the annotated but reduced number of attributes of the CUB dataset and the bird classes. For the sequential version, they first predicted the concepts using the concept model and gave these predictions to the second part of the model as inputs and they used the class annotations of CUB as label. The third version is joint training, where Koh et al. (2020) trained an end-to-end version of the model and optimized a combined loss consisting of the concept loss as well as the classification loss. We will not consider the joint model in this thesis, since it does not allow adversarial attacks on only one part of the model due to the training process and is therefore not comparable to the other bottleneck models introduced in Koh et al. (2020). For more details on the training of the models, please refer to Koh et al. (2020).

5.4 Scouter

The second model we investigated was Scouter (Li et al. 2020). This model used an approached based on a version of slot attention (Locatello et al. 2020), which Li et al. (2020) called "xSlot". This approach implemented uses a certain number of slots per class as a hyperparameter to predict positive supports (Scouter+) or negative supports (Scouter-) to decide for or against a certain bird class.
The classification pipeline of Scouter is shown in fig. 5.2. Li et al. (2020) used a ResNeSt-50 (H. Zhang et al. 2020) as their backbone, placed their xSlot attention module directly behind the backbone and added

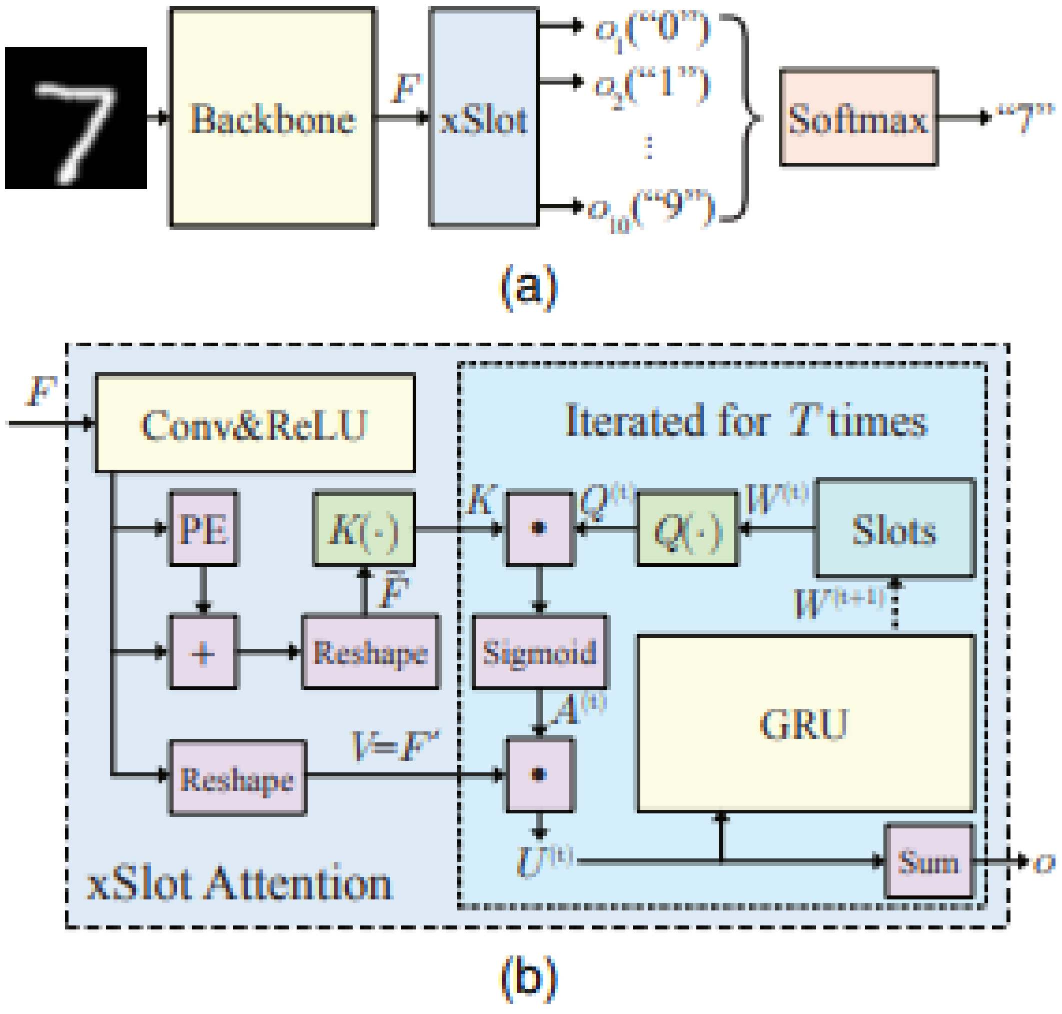

Figure 5.2: The classification pipeline of Scouter. (a) Overview over the model, (b) The xSlot Attention module in SCOUTER. (Taken from: Li et al. (2020), Figure 2.)

a softmax layer at the end for the final class prediction. We used the same training routine and the same hyperparameters to retrain the model, Li et al. (2020) used for training their model on the CUB dataset. First, trained the models with the provided hyperparameters for each one version of Scouter+ and one version of Scouter-. Both models were trained on the first 25 classes of CUB. Additionally, we trained Scouter on different hyperparameter settings to investigate the behavior of the different models regarding positive and negative supports as well as different numbers of supports per class. This left us with a total of five different versions of *Scouter*.

All hyperparameters we used can be seen in table 5.1. We use these hyperparameter settings to evaluate if the model behaves differently for positive and negative supports and for different numbers of supports per class.

All the hyperparameters serve a certain function. Starting with the *support*, this hyperparameter controls, the model learns features to decide for a certain class (positive support) or if the model learns features, which enables it to decide against a certain class (negative support).

Name	Support	Number of slots per class	λ
Scouter25+	positive	5	10
Scouter25-	negative	3	1
Scouter25+ s1 l10	positive	1	10
Scouter25- s1 l1	negative	1	1
Scouter25- s1 l1	negative	1	10

Table 5.1: Different hyperparameters used for training and testing variants of the Scouter models.

The second hyperparameter is the number of slots per class. This value indicates, how many different features of the image the model should use, which then serve as supports that lead to a decision of the model.
The λ value is the final hyperparameter. This value is used to limit the area and the importance of the support regions. Larger λ lead the model to select smaller regions by selecting fewer and smaller supports, while smaller λ lead the model to prefer larger areas.

5.5 Vision Transformer

The third large group of models we evaluated were the recently proposed Vision Transformer (Dosovitskiy et al. 2020). These models are based on the Transformer model coming from natural language processing (Vaswani et al. 2017) and they do not contain any convolutions. This characteristic makes Vision Transformers (Dosovitskiy et al. 2020) unique among the models we investigated because all other models use convolutions to process the input images. Since Vision Transformers up to this point were only trained on different versions of ImageNet (Russakovsky et al. 2014), on the in-house JFT-300M dataset of Google (Sun et al. 2017) and a paper about fine-tuning on small to mid-sized datasets was very recently published, we fine-tuned pretrained version of Vision Transformers on CUB. Here we trained 3 versions of different model and patch sizes: two versions of ViT small, one with patch size 16 and image size 224x224 (*ViT small p16*) and one with patch size 32 and image size 224x224 (*ViT small p32*). The third model was the ViT base version with patch size 16 and image size 224x224 (*ViT base*).

5.6 Slot Attention & Set Transformer

As a next step, we wanted to investigate if it is possible to create a model, which is using an adapted version of Slot Attention (Locatello et al. 2020) to predict the birds' concepts in the images and use a Set Transformer (Lee et al. 2018) as a second part to predict the bird classes from the concepts. The structure of the model based on the research of Stammer, Schramowski, and Kersting (2020). They trained their model on the CLEVR-Hans dataset, a self-generated toy dataset related to the CLEVR dataset (Stammer, Schramowski, and Kersting 2020), which is constructed to show confounding features in images.
Our general idea was to train a model which first extracts foreground and background information, based on the idea of textciteslotattention. In our experiment, we modified the model by adding a bottleneck layer similar to the one in Koh et al. (2020) at the end of the Slot Attention module to predict attributes for each of the slots. Therefore, we increased the size of the bottleneck layer to *number of attributes + 1* and used

the additional bit to indicate whether the respective slot is considered as foreground or background, and we set the *number of slots* per image to two.
The Set Transformer (Lee et al. 2018) should then use the output of the adapted Slot Attention model, which returns a matrix of shape

$$(batch\,size,\ number\,of\,slots,\ number\,of\,attributes\,+\,1)$$

to predict the final bird class. We remind here that the order of the slots is not fixed but determined by the strength of the features and the Set Transformer is a model, which is able to learn how to deal with this uncertainty.

5.7 Mask Bottleneck Model

Our next attempt to building a concept bottleneck-based model, which also includes information about foreground and objects in the images as well as background information followed a human like approach at scene understanding . Such an approach was introduced by Yi et al. (2018).
The Neural-Symbolic Visual Question Answering model, trained on the CLEVR toy dataset uses a pipeline of multiple steps, consisting of scene parsing, scene representation, question parsing and question answering. But only the scene parsing component of the pipeline was relevant for our work and we used this idea of scene parsing by object detection.
We will now describe the full *Mask Bottleneck pipeline*, which is displayed in fig. 5.3.

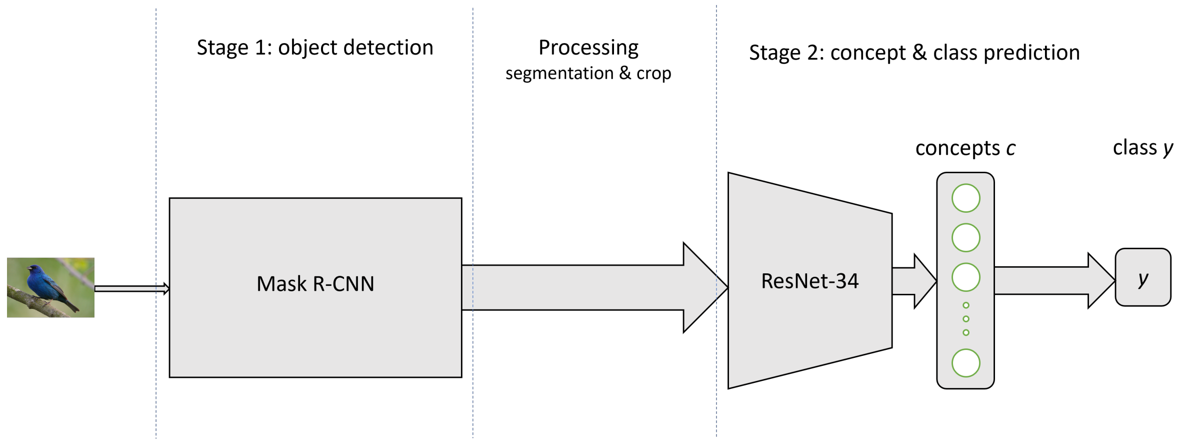

Figure 5.3: The pipeline of the Mask Bottleneck model consisting of high level object detection (stage 1) followed by a processing step for cropping (and segmentation) and a concept bottleneck model as the final stage 2 to predict the bird species.

First, we tested whether the pretrained Mask R-CNN was able to reliably detect birds in the CUB dataset and found out, that in total the Mask R-CNN was not able to detect a bird in 155 out of all 11,788 images with a certainty threshold of at least 0.5 and it was not able to detect a bird in 88 of 11,788 images with a certainty threshold of at least 0.3. These results on the pretrained Mask R-CNN were promising enough that we refrained from fine-tuning the model on CUB, since it already provided reasonable results by using a certainty threshold of 0.3.

With using Mask R-CNN for object detection, we now hat the first stage of our pipeline ready. This first stage can be seen in fig. 5.3 on the left side. After the object detection of the Mask R-CNN we had several possibilities of providing inputs to the second stage of the pipeline by using the outputs of the first stage. We summarize the usage of the results of the object detection step and the preparation for the second stage of the pipeline by naming it *processing*.
The first possibility was to crop the images based on the predicted largest bounding box, which was generated by using the outer coordinates of all bounding boxes arising from bird predictions of one image above the threshold of 0.3. The second possibility was to generate a bounding box of the outer coordinates of the sum of the segmentation masks of all bird predictions with a certainty above 0.3. Since the bounding boxes created of these two variants were not equal, we chose to use both variants. The third and fourth possibility of providing input to the second stage is to apply the predicted segmentation mask of Mask R-CNN to the input image and then follow it up one time by cropping around the predicted bounding box and one time by cropping around the bounding box generated by the borders of the segmentation masks. These two possibilities are quite like the idea of training the Inception-v3 model on the CUB-black dataset (*Inception-v3-blac*, section 5.2). We now have found four different kinds of inputs, we can provide to the second stage of the pipeline, which will result in four different concept models to train for this part of the pipeline. We will call the dataset arising from one forward pass and the following processing step ***CUB-cropped*** and use it later again to train the models of the second stage of the pipeline.
The second part of the prediction pipeline consisted of a ResNet-34 (He, X. Zhang, et al. 2015) as backbone where we resized the last layer of the ResNet-34 to the 112 attributes used, similar to the bottleneck layer in Koh et al. (2020). After predicting the concepts, we used one fully connected layer to predict the bird classes again. Here, we had to consider the options, we described in the previous paragraph, and we tested all of them: the cropping using predicted bounding boxes and using the edges of the segmentation masks as well as applying the predicted segmentation masks before cropping.

6 Experimental Setup

After describing the datasets we used, and the models we developed and examined, we will now describe the experimental setups in this section. Afterwards we will describe the observations and results of the experiments regarding the robustness of these models to out-of-distribution attacks and to adversarial attack. Afterwards, we will follow up the experimental results with a discussion and analysis. We carried out two large experiments for each model. First, we are testing the models on the TravelingBirds dataset and secondly, we ran multiple adversarial attacks on the images and models. We will state here for the full chapter that if we are writing about executing an attack on images or on a model, we always talk about using the images from the test set of the respective dataset, the model was trained on, and which was not seen during the training process.

6.1 TravelingBirds – out-of-distribution Attack

The TravelingBirds attack is an out-of-distribution attack. With this attack we want to test the robustness of the models in different and more difficult settings, which are generally not seen during the training process. In our case, we are using the TravelingBirds dataset (Koh et al. 2020) which displays birds in situations, which are not shown in the CUB dataset. While in CUB most birds have either water as background, especially when they are flying or if they are waterbirds. Otherwise, the birds oftentimes have a green background, mostly consisting of grass, trees, and flowers. In TravelingBirds the background is either black (CUB-black), class dependent (CUB-fixed) or randomly chosen (CUB-random). For more details, please refer to section 4.3.
For this attack, we first checked the accuracy on the in-distribution dataset, which was most of the times the CUB dataset. For Inception-v3 black (Szegedy et al. 2015) the in-distribution dataset is of course CUB-black and for the second part of the mask pipeline the in-distribution dataset is the test set resulting of the forward pass of stage one and the following processing step. We will call this dataset *CUB-cropped*. After we have run the model on the in-distribution dataset, we ran it on the three respective out-of-distribution datasets, which is three of the four of CUB (or CUB-cropped), CUB-black, CUB-fixed and CUB-random, to determine the accuracy of the models on these datasets too.

6.2 Adversarial Attacks

Testing the adversarial robustness of the different models was the second large experiment we carried out. For the adversarial attacks, we used four different white-box attacks. These four attacks are: fast gradient sign method (FGSM) (Goodfellow, Shlens, and Szegedy 2015), basic iterative method (BIM) (Kurakin, Goodfellow, and S. Bengio 2016), projected gradient descent (PGD) (Madry et al. 2019) and SparseFool

(Modas, Moosavi-Dezfooli, and Frossard 2018). Adversarial attacks are designed to fool neural networks by adding seemingly random and small noise to images. While this noise might not be perceived by humans it reduces the confidence for predictions of neural networks or the neural networks change their prediction completely due to the adversarial noise.
We will first provide two definitions for black-box and white-box attacks. (Guo et al. 2019; Bhambri et al. 2019). Black-box attacks have limited knowledge of the model they attack. For attacking the model, they might have the labeled output for some input they provide, but sometimes, they don't even have the label information. These attacks are designed and constrained around querying the model inputs and observing labels and confidence scores.
White-box attacks on the other hand have detailed information about the target model. Their knowledge includes the learned weights, parameters used and the gradient, which results by passing input through a model. White-box attacks sometimes also know the labels of the training data. With this knowledge the attacks usually generate adversarial examples by using the gradient to manipulate the input and violate the decision boundaries of the respective model.
In this thesis we do not attempt to achieve a fooling rate of 100% for each model, since we are interested in the behavior of a model under attack. We are aware that most models can be fooled in every input case when using white box attacks and although this also happens when applying our attack setups, but the general intention is to observe and evaluate the behavior under attack where some models might be more resistant than others to specific attacks. This is also the reason why we used a lot of different ϵ values (refer to section 6.2.5 for more details) to observe and analyze the model behavior.
We note here that we use models, which require a black background for their images. To not mix up adversarial attacks with out-of-distribution data, we adapted the adversarial attacks for these cases in a way that they only perturb the bird within the image, but that they leave the black background unaltered.

6.2.1 FGSM

Fast gradient sign method (Goodfellow, Shlens, and Szegedy 2015) is the first adversarial attack we used. This attack determines the sign of the gradient after a forward pass of the image and depending on the determined sign at each pixel of the image, it adds some noise level ϵ to the image. This is the simplest adversarial attack we use.

6.2.2 BIM

The second attack is basic iterative method, which is also called "Iterative FGSM". This attack applies FGSM multiple times, depending on the number of iterations. In each iteration, the noise level added to the image is a small value α with $\alpha < \epsilon$, which we set to $\frac{1}{255}$. After each iteration, the pixel values are clipped to ensure that the sum of the changes of each pixel do not exceed the ϵ border of the original image. The number of iterations used is determined by a heuristic, which follows the formula: $floor(min(\epsilon * 255 + 4, 1.25 * \epsilon * 255))$. They will get very large for larger epsilons, for example, with an epsilon of 0.01, BIM will do three iterations, but with an ϵ of 0.05 it will already do 15 iterations. Because this number of iterations will get even larger for larger ϵ, we introduced an upper bound for the number of iterations, limiting this heuristic to a maximum number of seven iterations. The final formula also includes a lower border for doing at least one iteration for small ϵ. Therefore, the final formula for the heuristic is: $min(max(floor(min(\epsilon * 255 + 4, 1.25 * \epsilon * 255)), 1), 7)$.

6.2.3 PGD

The authors of the publication of projected gradient descent (Madry et al. 2019) optimized a saddle point problem and developed PGD to solve it. They state that the saddle point problem regarding adversarial attacks and their counterpart of robust optimization and adversarial training. Robust optimization as well as adversarial training both have similar goals. They both aim for training networks which are more resistant and ideally not susceptible to adversarial attacks. They defined this adversarial training as a saddle point problem as

$$\min_{\theta} \rho(\theta),\ where\ \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{S}} L(\theta, x+\delta, y)\right]$$

. This equation consists of an inner maximization and an outer minimization problem. It maximizes the loss of the network, which the exact problem of attacking a given neural network and it is achieved by using an adversarial attack. To optimize the outer minimization problem wants to minimize the adversarial loss given by the inner attack problem, which is the task of training a robust classifier by using adversarial training. PGD also bases on FGSM, since it is a multi-step variant of it, and it also has similarities to BIM. PGD first applies randomly distributed noise in the range of $[-\epsilon, \epsilon]$ to each pixel of the image. Afterwards it applies FGSM iteratively for given number of steps and α and ensures after each iteration, that the maximum change per pixel of ϵ is not violated by clipping the maximum change to ϵ. For this attack, we limited the number of iterations to two and set the alpha to $\frac{1}{255}$. The alpha represents the maximum perturbation per iteration per pixel.

6.2.4 SparseFool

The fourth and final adversarial attack we applied was SparseFool (Modas, Moosavi-Dezfooli, and Frossard 2018). This attack is different from the others in several ways. First, all the already described attacks manipulate every pixel in the image by a little, without taking the impact of the manipulation for a certain pixel into consideration. For the reason of manipulating every pixel in a given image, the first three attacks, FGSM (Goodfellow, Shlens, and Szegedy 2015), BIM (Kurakin, Goodfellow, and S. Bengio 2016) and PGD (Madry et al. 2019) can also classified to fulfilling the distance measure L_∞. SparseFool on the other hand only manipulates a few pixels in each image, but the changes to the pixel are quite large and more easily perceived by humans. The distance measure for SparseFool is L_0. Secondly, FGSM, BIM and PGD all use information from the simple first order derivative to determine the manipulation applied to an image, while SparseFool will calculate the jacobi matrix between the predictions of a network and the input image. SparseFool will also allow an overshoot, meaning it allows to violate the constraint of a maximum manipulation of a pixel and it is a geometric interpretation of finding an adversarial example. This overshoot is allowed since it increases the misclassification rate.
SparseFool is an iterative approach to adversarial attacks, and it consists of two parts. In each iteration, SparseFool first applies DeepFool (Moosavi-Dezfooli, Fawzi, and Frossard 2015) to find an estimation of a boundary point for the decision hyperplane and the associated normal vector. Now SparseFool applies a linear update by projecting previously estimated boundary point onto the decision hyperplane and it will stop once a specific direction does not provide a solution. At this point, the perturbed image at this coordinate has reached its extrema value and can be ignored in future iterations.
SparseFool requires the parameter λ as an input. This λ indicates a trade-off between fooling rate, sparsity and complexity. λ values close to 1 lead to sparser perturbations, but a lower fooling rate and increased complexity, while higher λ values lead to faster convergence, but less sparse solutions. We chose to set our

λ values to $\lambda = \epsilon * 10$. Due to the same naming of other hyperparameters as λ too, we will refer to the λ of SparseFool as ϵ, while the formula

$$\lambda = \epsilon * 10$$

still holds.
We set the number of iterations for both SparseFool and DeepFool to two. We had multiple reasons for this choice. Firstly, the library, which provided the attacks, was not implemented to use a batch size different from one and secondly it requires roughly one to two seconds per image to perturb it using two iterations. We also limited the number of images to 100 at this point, which were chosen randomly from the test set of the CUB dataset. While one to two seconds of calculation time per image by using 100 images might not sound much, we have to emphasis here that these number only hold for one lambda and one model. Since we used models of three different seeds and eleven lambda values, we are roughly at one hour just to attack one single image by using two iterations for SparseFool and two Iterations for DeepFool. We used an overshoot of 0.02 for SparseFool and the inner DeepFool iterations.

6.2.5 Epsilon Value

The first three attacks (FGSM, BIM, PGD) require ϵ values as the maximum perturbation to each pixel in the image. SparseFool as the fourth attack requires a λ to indicate the sparsity and complexity of its perturbations. We used the same ϵ value for each attack and to determine the λ-values for SparseFool, we multiplied the ϵ values by 10. We estimated the range for the λ-values by the analysis of the effect of λ on SparseFool which was provided by Modas, Moosavi-Dezfooli, and Frossard (2018). The ϵ values were:

$$[0.0, 0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025, 0.05, 0.1, 0.15, 0.2]$$

for all our setups, with $\epsilon = 0.0$ calculating the accuracy on the unaltered dataset.
We will stage here againt, that to avoid any confusion regarding the λ hyperparameter of *Scouter* and the λ hyperparameter of *SparseFool*, we will refer to the λ value of *SparseFool* as ϵ and we will depict it as such in our visualizations. For all our calculations and visualizations, the previously described formula for the *SparseFool* λ is still: $\lambda = \epsilon * 10$ holding.

6.2.6 Attack Setups

Each of the concept-based models consist of two steps. Firstly, predicting the concepts from the image and secondly to predict the bird class from the already predicted concepts. We had to consider this structure of the models when attacking them. We therefore introduced three different setups satisfy every individual requirement of the models.
We will now introduce some definitions to indicate which part of the model gets attacked, shortly describe why we attacked this part of the model, and we will also describe difficulties we faced in the individual requirements of the models.
First, we attacked three different kinds of models. The concept model, which gets an image as its input and predicts the concepts (*XtoC*), the second part (sequential or independent model, depending on the training routine), gets the predicted concepts and predicts the bird classes (*CtoY*) and the end-to-end version takes an image as inputs and predicts the bird classes as output (*XtoY*), with the intermediate step of predicting the concepts. When attacking the end-to-end version of the concept-based models, we

do not consider or manipulate the intermediate prediction of the concepts by ourselves, but we just pass the predicted concepts to the next step. For models which are not concept-based, such as Scouter (Li et al. 2020), Vision Transformer (Dosovitskiy et al. 2020) and Inception-v3 (Szegedy et al. 2015), we only consider the end-to-end version of the attacks (*XtoY*), since there is no intermediate step of predicting concepts.
We will emphasis here that the pipeline of the *Mask Bottleneck model* even consists of three parts, which we divided in two stages (compare to 5.7), the object detection as a first step, where there is a larger processing step afterwards and then there is the second stage of the pipeline, consisting of concept prediction and class prediction. For this pipeline we additionally designed an individual step due to the two possibilities here: we can attack the first stage of the Mask Bottleneck model, namely the Mask R-CNN (He, Gkioxari, et al. 2017) and feed the adversarial predictions to the next stage or we could attack the second stage of the pipeline by attacking the unaltered outputs of the Mask R-CNN. Although the second scenario is very unlikely in a real-world application, we investigated this scenario because it provides comparability to the other concept bottleneck models.
While attacking the Mask R-CNN we also had to deal with the possibilities that the model is not able to detect any bird in the input image anymore. In this case it is not possible to crop out a part of the image, which contains the bird prediction, to predict the attributes and species in the next step. We will provide three kinds of evaluation here. One will only consider the images, which do contain birds, one will consider only the images which do not contain birds. In this case the adversarial images will be put directly into the second stage of the pipeline. The third kind of evaluation we will consider all images, either, when they are cropped and have a segmentation mask applied (the details of this process were described in section 5.7) or when the Mask R-CNN does not find a bird in the adversarial image it will pass the adversarial image unaltered to the next stage of the pipeline. Since the gradient flow is interrupted when trying to build one forward pass from the input of the Mask R-CNN to the output of the second stage model due to the cropping at the processing step, there will be no attack considering all stages of the pipeline at once.

6.2.7 XtoY

The first attack setup attacks the end-to-end version of all models. Here we have the *input* $\boldsymbol{X}$, which will be perturbed by an adversarial attack to change the *prediction* $\boldsymbol{Y}$. The idea of this attack is to evaluate the performance of the bottleneck models, when they have to carry out the full predictions starting with the images and predicting the class labels. Additionally, this is, as already mentioned, the only attack setup for end-to-end models such as Inception-v3, Scouter, and Vision Transformer.

6.2.8 XtoC

The second setup will attack the concept part of the models. Here we will perturb the *input* $\boldsymbol{X}$ to change the *concept predictions* $\boldsymbol{C}$. At this point, we have to notify that we will not use SparseFool in this setup, since the calculation of the decision boundaries depend on predicting one single class out of several, while the concept predictions $\boldsymbol{C}$ require the binary prediction of multiple concepts, where some of them can be either present or absent, but in all cases, it is expected that multiple concepts are present. The intention of this attack is to see how susceptible the concept models are to adversarial attacks. To determine the accuracy of the concept predictions, we binarize the concept outputs with a threshold of 0.5. We are aware that the binarization might enhance the performance of the model, but since the training and validation accuracy of the concept model is determined by binarized labels and the independent training is performed on binary labels as well, we evaluated the change in accuracy due to adversarial attacks this way. We will still remark at this point that the loss, which we can calculate using binary cross entropy might increase a lot more than the accuracy decreases in this case.

6.2.9 CtoY

The third setup will change the *concept outputs* $\boldsymbol{C}$ of a concept model to alter the *class prediction* $\boldsymbol{Y}$ of the second stage model (independent or sequential). We chose this setup to evaluate how many predicted concepts are actually relevant for the prediction of the class labels and how changing a few or all concepts affects the prediction of the final class labels.

6.2.10 Mask Bottleneck Model

Due to the more complex pipeline of the Mask Bottleneck model, we will add additional setups here. Since the pipeline has two stages, we will attack both stages individually. This way, we will try to get a more detailed insight into the capabilities of the Mask Bottleneck Pipeline.
As a first part, we will attack the Mask R-CNN as it is by first passing the images of the TravelingBirds dataset through and calculate the performance of Mask R-CNN regarding the number of birds detected across all images (one bird is possible per image) and we will also track the certainty scores of its prediction. The second attack will include the three adversarial attacks FGSM, BIM and PGD, with which we will attack the Mask R-CNN (He, Gkioxari, et al. 2017). We will not use SparseFool here.
All adversarial attacks we used are generally designed to attack classification networks. FGSM, BIM and PGD do this by using the loss to predict the sign of the gradient of each pixel and perturb the images on this basis. SparseFool on the other hand bases its perturbation on the predicted classes and the labels of the model and it does not base its perturbation on the loss. Partial adaptations of SparseFool to the new circumstanced would have been possible, but there were also some obstacles, which would have exceeded the scope of this thesis. These obstacles include that for Mask R-CNN we needed to base the perturbation on the loss, since there is not only a classification part, but also the prediction of segmentation mask and bounding boxes, which is a regression problem.
After applying the adversarial attacks as well as TravelingBirds, we processed the outputs accordingly to be ready to be send to the second stage of the pipeline. When attacking Mask R-CNN (He, Gkioxari, et al. 2017) we did not perturb the images any further except for resizing the image to fit into the second stage of the pipeline and tracked the results. This included that we pass on all images put into the Mask R-CNN. If the Mask R-CNN was not able to detect a bird in the TravelingBirds set or in an adversarial image , we did not apply cropping or segmentation, but we passed on the image after resizing it and calculated a second score for these images.
The second setup is the already familiar process of attacking a slightly different Concept Bottleneck (Koh et al. 2020) model where we attack the second stage pipeline by applying adversarial attacks on the input image of the second stage using the three setups ***XtoY***, ***XtoC***, ***CtoY*** and calculated the results from here.

7 Experimental Results

In this chapter we will describe the results for each model and the different experiments. We will also include the respective baseline for the respective, which is the Inception-v3 model (Szegedy et al. 2015) trained on 200 classes for most cases. For Scouter it is the Inception-v3 model trained on 25 classes, and we will additionally describe the Inception-v3-black model (Szegedy et al. 2015), which we trained on CUB-black (4.3).
The accuracies on the home datasets can be seen in table 7.1. The best results for each task for the respective dataset (CUB or CUB (25 classes)) are in bold letters.
This section will start with the Concept Bottleneck (Koh et al. 2020) models and followed with Scouter (Li et al. 2020), Vision Transformer (Dosovitskiy et al. 2020), the combination of Slot Attention (Locatello et al. 2020) and Set Transformer (Lee et al. 2018) and our self-developed Mask Bottleneck model. After the description of the results, we will also discuss them shortly in the environment of the baseline model, variations of the respective model and the different attack setups. Following this section, we will provide a general cross-model discussion of the results.

7.1 Concept Bottleneck Models

The first group of models, we attacks were the Concept Bottleneck models from (Koh et al. 2020). Here all three of the adversarial attack setups were applicable as well as the TravelingBirds-out-of-distribution attack.

7.1.1 TravelingBirds

Koh et al. (2020) measured the prediction errors on different tasks. Therefore, we reran the experiments on the TravelingBirds dataset to get the classification accuracy of the predictions $\boldsymbol{Y}$. In figure 7.1, we show the absolute and relative classification accuracies on the different datasets. The first dataset is also the dataset on which the models were trained on. We show the absolute accuracies in fig. 7.1 (top) and we will also show the relative accuracies in fig. 7.1 (bottom) to not only see, which model performed better overall, but to evaluate, which model performed better or worse on a relative basis.

In fig. 7.1 (top) one can see, that the Inception-v3 baseline clearly performs the best with a classification accuracy of 83% and it also performed best on the out-of-distribution datasets with an accuracy of 69% on CUB-black 67% on CUB-fixed and 67% on CUB-random. The second-best model is the independent version of the Concept Bottleneck models with an accuracy of 76% on CUB, followed with a larger gap by the sequential version with an accuracy of 62%. When we now take a look on fig. 7.1 (bottom), we can

Model name	**Type**	**Home Dataset**	**XtoY acc**	**XtoC acc**	**CtoY acc**
Inception-v3-200	end-to-end	CUB	**0.827**	-	-
Inception-v3-25	end-to-end	CUB (25 classes)	**0.862**	-	-
Inception-v3-black	end-to-end	CUB-black	0.794	-	-
Concept Bottleneck	Independent	CUB	0.760	**0.966**	0.594
	Sequential		0.617		**0.752**
Scouter25+	end-to-end	CUB (25 classes)	0.798	-	-
Scouter25-	end-to-end	CUB (25 classes)	0.779	-	-
Scouter25+_s1_l10	end-to-end	CUB (25 classes)	0.817	-	-
Scouter25-_s1_l1	end-to-end	CUB (25 classes)	0.814	-	-
Scouter25-_s1_l10	end-to-end	CUB (25 classes)	0.837	-	-
ViT small p16	end-to-end	CUB	0.702	-	-
ViT small p32	end-to-end	CUB	0.515	-	-
ViT base p16	end-to-end	CUB	0.508	-	-
Mask Bottleneck cropbb	Sequential	CUB-Cropped	0.612	0.937	0.618
	Independent		0.605		**0.608**
Mask Bottleneck segbb	Sequential	CUB-Cropped	0.548	0.925	0.556
	Independent		0.542		0.547
MaskBottleneck cropbb use-seg	Sequential	CUB-Cropped	0.597	0.935	0.602
	Independent		0.588		0.593
MaskBottleneck segbb use-seg	Sequential	CUB-Cropped	0.519	0.922	0.532
	Independent		0.509		0.525

Table 7.1: Training results for all Models and model types, we used, the cases **XtoC** and **CtoY** are only for Concept Bottleneck models.

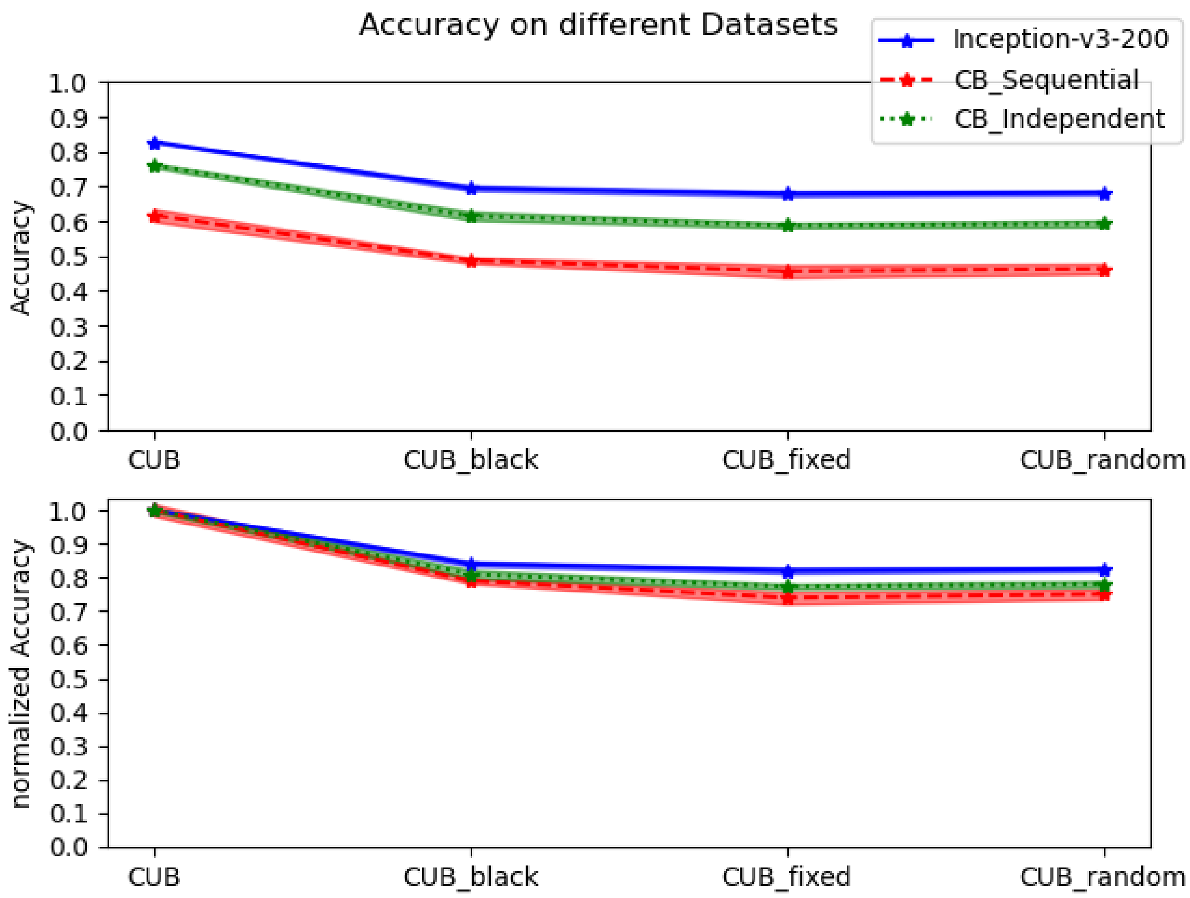

Figure 7.1: Top: the absolute accuracies for Concept Bottleneck models on the different datasets. Bottom: the normalized (relative) accuracies for Concept Bottleneck models on the different datasets.

see that the relative accuracy and the relative loss of accuracy on the out-of-distribution datasets. There is a loss of 15% to 25% in relative accuracy when changing from CUB to the TravlingBirds datasets, with the smallest loss coming from the Inception-v3 baseline and the highest loss in accuracy coming from the Sequential Bottleneck model.

Summary

With these results, we can see that the baseline model clearly performs the best on an out-of-distribution scenario. Especially since the difference in accuracy is larger when the background gets more different from the in-distribution dataset and when it gets more complex than in the original distribution. We can still assume that the Concept Bottleneck models as well as the baseline model are all able to extract features of the bird during the training process but that they also take background information into consideration when making their decisions.
One fact that is also important is the general difference between the accuracy of the independent and the sequential version of the Concept Bottleneck models. As we already stated in the model description, both models have the same Concept Models as a first step of the pipeline, and they therefore get the same

concept predictions. Due to the fact that the difference in accuracy is large, even on CUB, we can safely assume that the independent training is more robust to outliers and changes than the sequential version. This also reflects in the slightly better results which can be seen in fig. 7.1 (bottom), where the relative accuracy of the independent models is slightly higher. We assume that the most likely reason for that behavior lies in the input of the training, which is always zero or one for the independent model, while the sequential model is getting input values between zero and one, which can be seen as the certainty of the concept model for a concept to be present or absent.

7.1.2 Adversarial Attacks

The second kind of attacks were adversarial attacks. As described previously, we will start with the ***X to Y*** setup and follow up with the setups of ***X to C*** and ***C to Y***.

XtoY

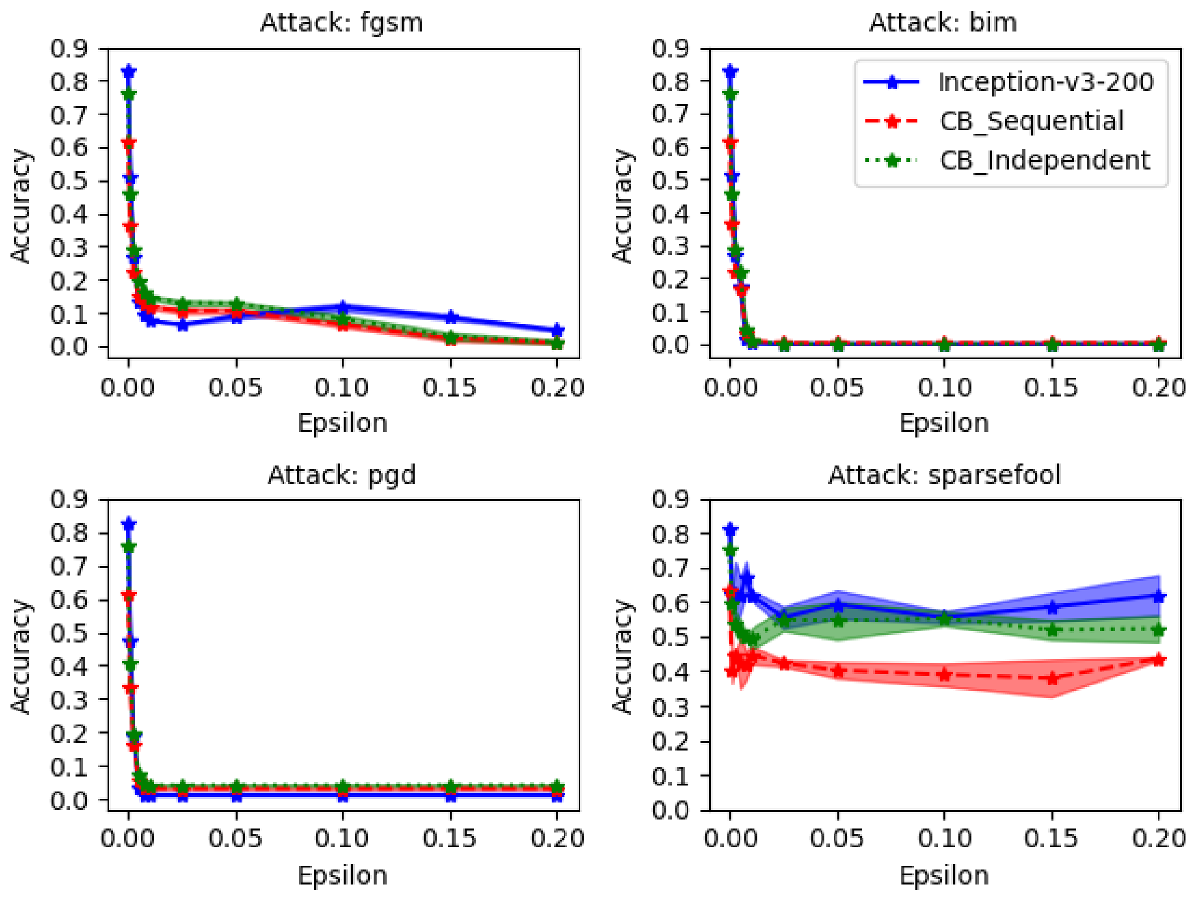

Figure 7.2: Results of the adversarial attacks on the Concept Bottleneck models of setup *XtoY* for all ϵ values. Task: Attack image **X** to change class predictions **Y**.

Fig. 7.2 shows the absolute accuracies of the two Concept Based models (Sequential in red, Independent in green) and the Inception-v3 baseline from Koh et al. (2020). The first important fact is again the difference in accuracy on the unchanged CUB dataset. Here the Inception-v3 baseline clearly performs the best, followed by the Independent and then the Sequential model.
When we take a look at fig. 7.2 top left, which shows the results of the *FGSM* attack (section 6.2.1), one can see that all accuracies decrease quite fast with increasing ϵ values. the models with the largest decrease in accuracy on the first ϵ values is the Inception-v3 baseline, which started with the highest accuracy and fell to the lowest accuracy at an $\epsilon = 0.025$. What is really interesting is the behavior of the Inception-v3 baseline with $\epsilon > 0.025$. While the Concept Bottleneck models both further decrease in accuracy with increasing ϵ and therefore stronger perturbations to the input images, all three seeds of the Inception-v3 model increased its accuracy and reached a top at $\epsilon = 0.1$. Afterwards the accuracy decreased again.
Fig. 7.2 top right, shows the adversarial attack BIM (section 6.2.2). Here the behavior of all three models is very similar. The accuracy falls very fast with increasing ϵ values up to an accuracy of 0% at $\epsilon = 0.025$ for all models. The decrease in accuracy is only slightly slowed down at the ϵ values of 0.0025, 0.005 and 0.0075, which can be seen in fig. 7.3, top right.

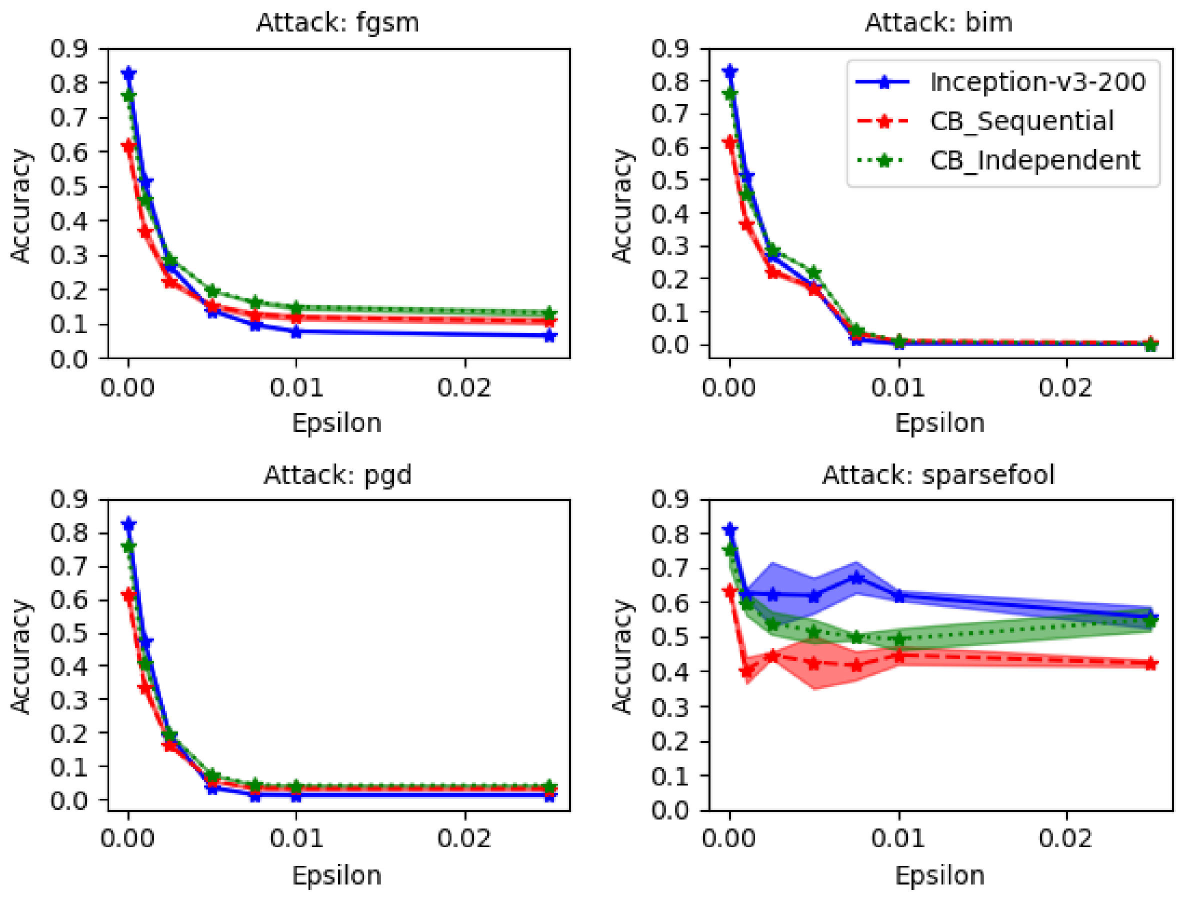

Figure 7.3: Results of the adversarial attacks on the Concept Bottleneck models of setup *XtoY* for $\epsilon <= 0.05$. This graph displays the change for small ϵ in a better way.

The results of the PGD attack can be seen in fig. 7.2, bottom left. Here the accuracy of all models drops below 10% up to $\epsilon = 0.005$ and it remains close to 0% for all the larger ϵ values. SparseFool shows a completely different behavior for all models. While there is still the initial drop in accuracy for all models, larger ϵ and therefore the according $\lambda = \epsilon * 10$ values do not have any further impact on the classification accuracy. This can be seen in in 7.2, bottom right, where the accuracy is moving around a mean value for each of the models. In absolute values, the Inception-v3 model does perform the best in this attack, followed by the Independent and then the Sequential Bottleneck model.

Summary

While the Inception-v3 baseline seems to perform the best in absolute accuracy, one can see in fig. 7.2 that the largest benefit of the Inception-v3 model is the higher initial accuracy and that it is in fact more susceptible to the two of the first three adversarial attacks (FGSM, PGD). For BIM, we cannot make a final analysis, since all the models perform bad under this attack, because they are not able to classify a relevant number of images correctly. Even for PGD, our analysis is critical since the Inception-v3 model performs worse in absolute and relative accuracy, but all models have an absolute accuracy below 5%.
The behavior of the models under the attack of SparseFool is interesting, since the attack does not seem to be able to harm the decision making of the models further than up to a certain point, but the accuracy remains fairly constant then.

XtoC

Fig. 7.4 shows the results of the ***X to C*** attack. As we already stated, we will display all results of the concept model from Koh et al. (2020) in one figure, since we do not have another model to compare it to, yet.
As can be seen in fig. 7.4 the concept accuracy is decreasing from a very high level of 97%. For PGD, the decrease in accuracy is very low, since the accuracy stays above 90%. BIM and FGSM decrease the accuracy a lot more to an accuracy around 80%, which is still very high considering the previous attack setup *textbfX to Y*. While the initial decreases in accuracy for FGSM and BIM are quite large, the accuracy for BIM does not change after $\epsilon = 0.025$ anymore. The decrease in accuracy then stagnates for a few ϵ, but it continues to fall again with $\epsilon >= 0.05$, leading to the lowest accuracy of all attacks on this model.

Summary

We will note here again that the predicted concepts are binarized before calculating the accuracy. This process might leave the impression that the attacks do not change the concept output by a lot. There are still a few important points to note here, which can be derived from analyzing the results of this attack setup. Firstly, every attack is able to change the image in a way, that the predicted concepts change their binary label. Secondly, larger changes, regarding FGSM and BIM are only able to change roughly 20% of the predicted labels by the perturbation of the input images. The ability of the attacks to manipulate only 20% of the predicted labels lead to two interpretations. The concept model seems to have a high certainty in its predictions, even with larger changes to the input image, since the barrier of 0.5 is not violated in 80%.
Since the algorithm of BIM and PGD inside the loop is quite similar due to their derivation from FGSM the

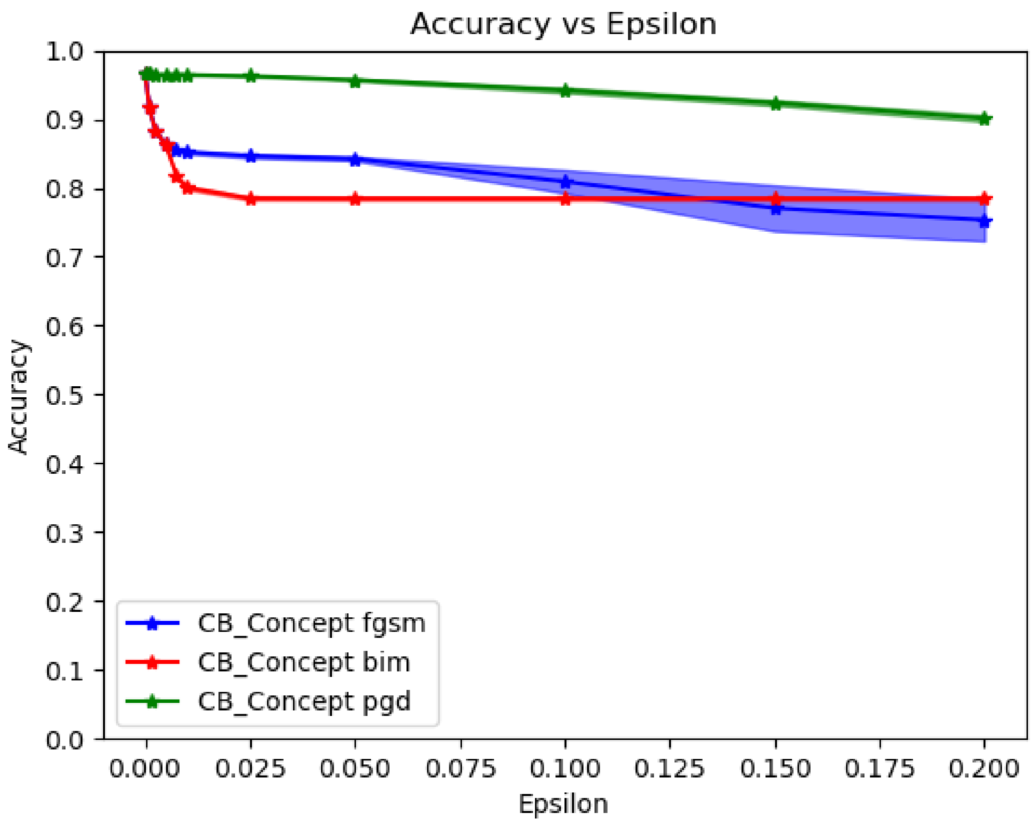

Figure 7.4: Results of the adversarial attacks on the Concept Bottleneck models of setup *XtoC*. Task: Attack image **X** to change concept predictions **C**.

only larger difference of both attacks is the number of iterations performed per attack. It is very difficult to find reasons for the very slowly decreasing accuracy of the model attacked by PGD. One reasonable explanation might be that the certainty of the model regarding its predictions is quite high and that the certainty does not decrease far enough to be affected by the small perturbations of coming from the attack. Since the accuracy decreases quite a lot when using BIM, the initial decrease of accuracy is similar to FGSM. This can be observed until the heuristic determining the number of iterations for BIM changes the number of iterations to two at $\epsilon = 0.0075$, we can assume that first, BIM is much more closely related to FGSM, than PGD is and secondly that the clamping behavior to ensure values in range 0 to 1 for each pixel is more efficient for BIM.
We will also state here again that while the proportion of changed concepts for the model under attack seems to be quite low, the effect of the changed concepts might still be a lot higher, when attempting to predict the bird classes from these changed images or concepts, since the number of attributes per image is only 112 and the number of unique attributes is even lower.

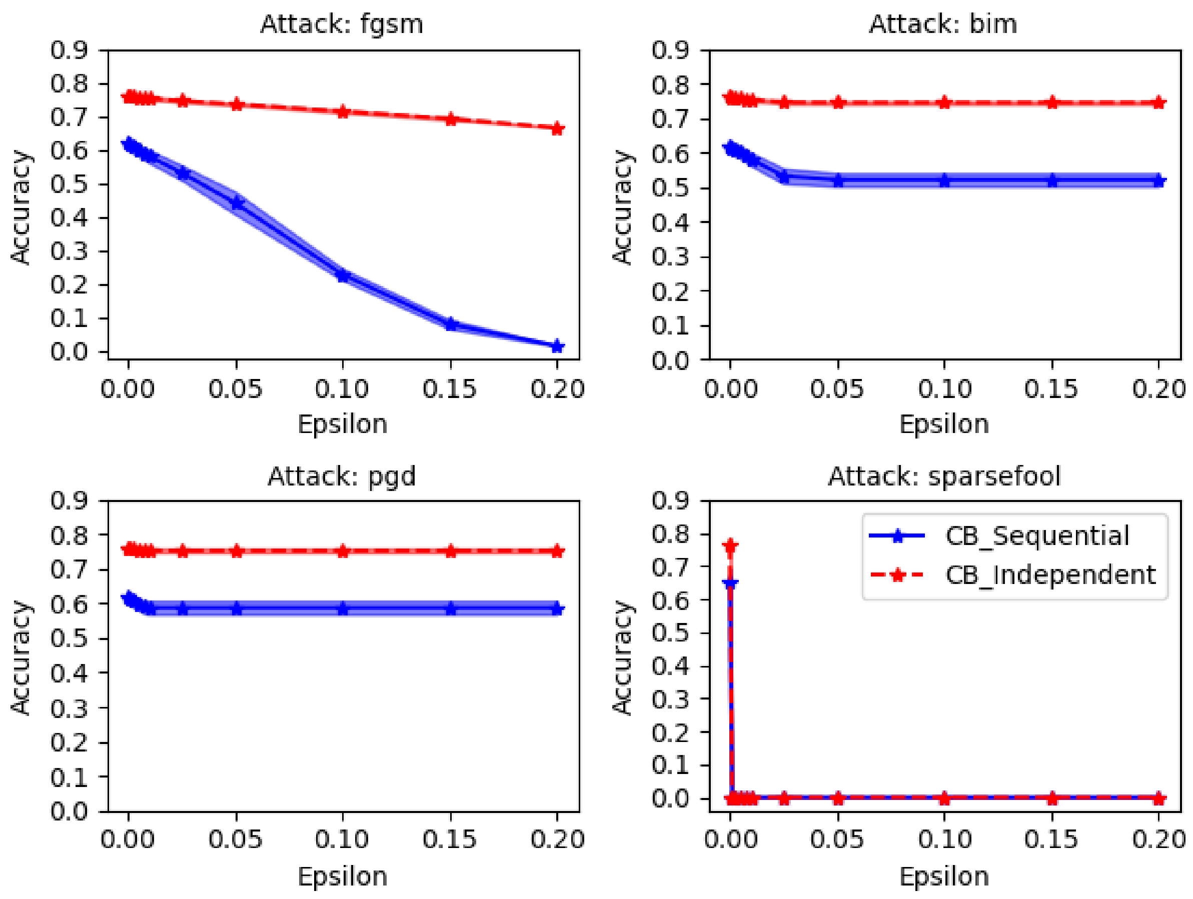

Figure 7.5: Results of the adversarial attacks on the Concept Bottleneck models of setup *CtoY*. Task: Attack concepts **C** to change class predictions **Y**.

CtoY

The results of the attacks for the ***C to Y*** can be seen in fig. 7.5. There is a very interesting behavior to be seen here across both the sequential and the independent version of the models as well as in between these models.
First, regarding FGSM (top left of figure 7.5) we see a very slight decrease in class predictions with increasing changes to the input concepts for the independent model. The absolute decrease in accuracy for the independent model is below 10%. The accuracy of the sequential model on the other hand decreases a lot faster up to 0%.
For BIM and PGD, the behavior of the models under attack is quite similar (fig. 7.5 top right and bottom left). There is an initial decrease in accuracy while the accuracy does not change anymore with $\epsilon >= 0.0025$. The sequential model still has a lower initial accuracy compared to the independent model and the initial decrease in accuracy is slightly larger.
Regarding SparseFool (fig. 7.5, bottom right) there is not much to describe. An attack with even the smallest ϵ decreases the accuracy of both models to 0%.

Summary

When analyzing the attacking results for this setup, it seems that small changes to all concepts do not have a large impact on the class prediction. This hold especially for the independently trained version of the model, but it is also true for the sequential version, although the impact of changes of the input concepts is larger than for the independent version. As we already stated for the previous attacking setup, the most likely reason for the higher susceptibility for adversarial attacks regarding the sequential model lies in the much higher uncertainties in concept predictions during the training time.
We can also safely state that manipulating a few of the input concepts using SparseFool, leads to very bad prediction results for both the independent and sequential version of the model. The main reason for this behavior might the sparsity of attributes, from which the model is forced to make its decision. While there are 112 concepts to predict, but each one is binarized and there are only 27 unique attributes, we oftentimes have multiple binary concepts referring to the same attribute, for example the wing color or the bill shape. Due to this sparse representation, we can see that changing very few of the predicted concepts by a lot, the prediction changes a lot.

7.1.3 Summary

Apart from the larger difference in initial accuracy between the Inception-v3 baseline and the two end-to-end Concept Bottleneck models, consisting of a *concept* part and either the *Independent* or the *Sequential* fully connected layer, the Concept Bottleneck models provide strong relative results regarding the out-of-distribution attacks.
Although none of the models tested in this section are very robust against adversarial attacks, which manipulate every pixel in an image, the end-to-end Concept Bottleneck models both performed quite well, and in most cases even better than the Inception-v3 baseline. The sole attack where the baseline model always performed better was *SparseFool*.
Regarding the attacks on the two parts of the concept bottleneck models, namely the concept model (*XtoC*) and the fully connected layer predicting the task *CtoY* showed good and robust results. The largest difficulty for these models seems to be the very sparse number of attributes, which have to be used to tell different bird classes apart. This holds especially since there are oftentimes larger families of birds of which the different species look very similar, even for human layman. Since the difference in accuracy between the concept predictions and the final class predictions is between 15% to 30% and therefore very large, we can assume that the most difficult task for the models is to master this task of predicting bird species from their attributes. We also saw that difficulty, when evaluating the results of the *XtoC* and *CtoY* tasks, which both showed that only a few predicted concepts change, when the models are attacked, but that very few changes of the concepts largely change the final class prediction.

7.2 Scouter Models

After describing and discussing the results for the Concept Bottleneck models (Koh et al. 2020), we will now move on to Scouter (Li et al. 2020). Scouter is using a variant of slot attention (Locatello et al. 2020) called “xSlot” (Li et al. 2020), which we described previously in section 5.4.

When we were retraining the Scouter models (Li et al. 2020), we knew about Li et al. (2020)'s statement about training instability, when having more than 100 categories. The authors assumed, this instability arises because effective supports do not appear consistently in all images of the same class, and they are not shared across other classes too. We tried to train Scouter on 100 classes instead of 25 classes, but the training instability appeared as the authors stated and after multiple attempts resulting in an accuracy below 10%, we stopped the training and went on to evaluate Scouter solely on the basis of 25 classes.
We will keep the order regarding the experimental setups and describe the results for TravelingBirds and then for the adversarial attacks. For each setup, we will first present the generated results for two Scouter models, one with positive supports and one with negative supports, corresponding to the training parameters provided by Li et al. (2020) and afterwards we will describe and discuss the results from the variations of the hyperparameters. All hyperparameter settings can be found in table 5.1. As a final part of this section, we will discuss the results in the environment of all Scouter models we tested. We will always provide the Inception-v3-25 model trained on 25 classes as a baseline for all Scouter models for fair comparison, since Scouter is trained on the first 25 classes of CUB too.

7.2.1 TravelingBirds

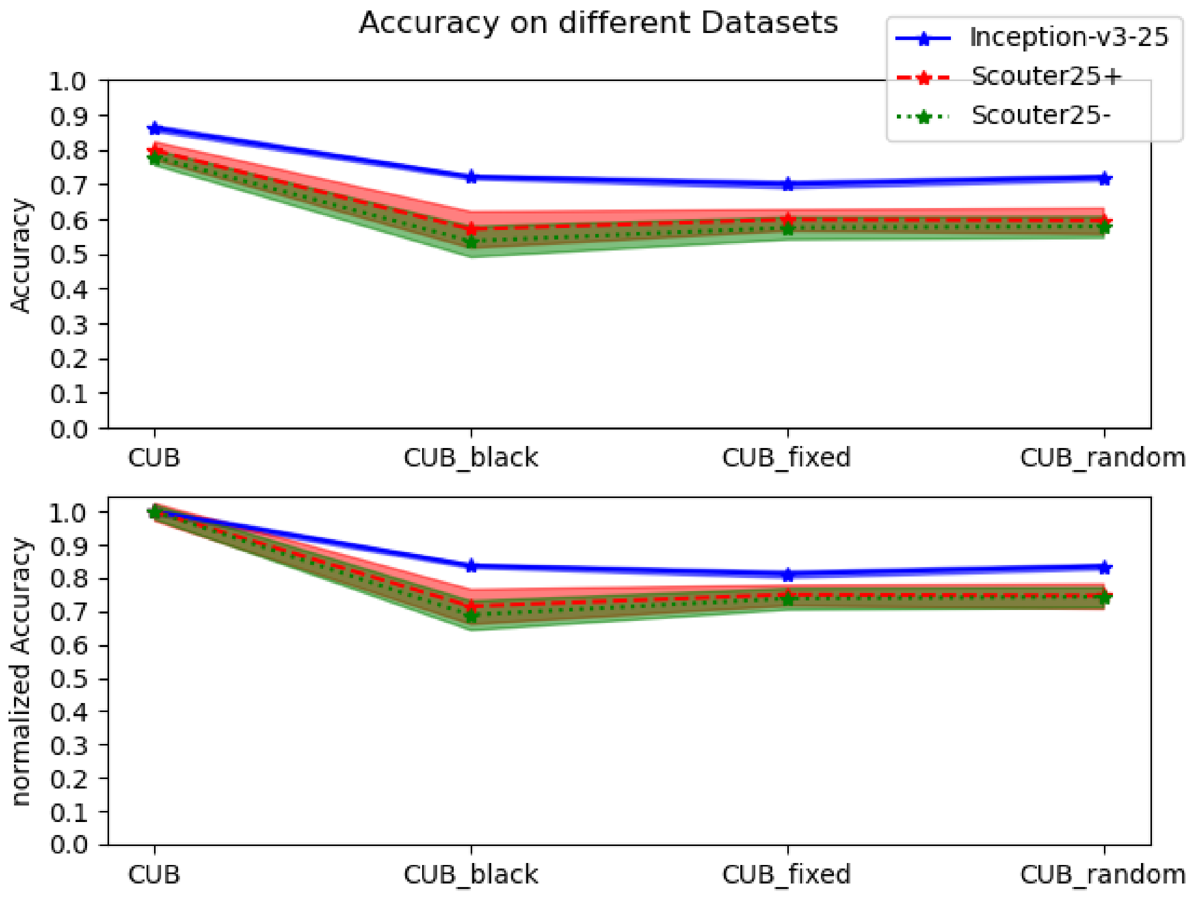

Figure 7.6: Top: the absolute accuracies for the first two Scouter models on the different datasets. Bottom: the normalized (relative) accuracies for the first two Scouter models on the different datasets.

Starting with the TravelingBirds setup, we can see the results of the positive and negative Scouter variants in fig. 7.6 (top), in red for the positive supports and in green for the negative supports. As can be seen, both scouter models behave quite similar. The accuracy is the highest on CUB and the lowest accuracy on the out-of-distribution datasets is on CUB-black. On CUB-fixed and CUB-random the accuracy is higher than on CUB-black, but a lot lower than the accuracy of the Scouter models on their already known dataset CUB.
Fig. 7.6 (bottom) shows the normalized accuracy for these models. Here, it is very interesting that both Scouter models almost loose the same amount of relative accuracy on the different out-of-distribution datasets. The difference in relative accuracy is the largest on CUB-black with 4% and the smallest difference in relative accuracy on the TravelingBirds datasets is 2% between the two Scouter models.
To now compare the two Scouter models to the Inception-v3 model both graphs show a clear tendency. First, the Inception-v3 model achieves a higher absolute accuracy (fig. 7.6) (top) on all four datasets, the in-distribution CUB dataset as well as the three out-of-distribution datasets. Additionally, the Inception-v3 model performs worst on CUB-fixed and slightly (1%) better on CUB-black and CUB-random. This behavior can also be seen in fig. 7.6 (bottom). Here there is a drop in relative accuracy, when classifying the out-of-distribution images, but the accuracy remains pretty similar on all the three datasets.
One other interesting fact is that the Inception-v3 model performs better than the two Scouter variants on all datasets and that is also has a very much lower standard deviation between the three seeds than the Scouter models.

We will now move on to the *Scouter variations* we tested. Fig. 7.7 (top) shows the accuracies of the three variations as well as the accuracy of the Inception-v3 baseline and fig. 7.7 (bottom) shows the normalized results of these models. To remind again of the different hyperparameters and their implications, we refer to section 5.4. The accuracies of the different variations all show a similar behavior to the initial two Scouter models regarding the different datasets and we will therefore not repeat it, but we will refer to the previous paragraph. The first observation is that Scouter25+ using one slot per class and $\lambda = 10$ behaves equally to Scouter25-, which is trained with negative supports, one slot per class and $\lambda = 1$. These results are unexpected, especially since the version (Scouter25-, one slot per class, $\lambda = 10$) with negative supports but otherwise similar hyperparameters yields slightly better results on an otherwise similar behavior. Therefore, we expected a more different behavior.

Summary

Regarding the TravelingBirds test, all Scouter models perform worse on out-of-distribution datasets than the Inception-v3-25 baseline does. Therefore, the choice would not be one of the Scouter models if one should choose the most robust model against out-of-distribution attacks. But since Scouter still delivers decently strong results, Scouter might still be interesting when it comes to choosing models for such a task, which enable partial comprehension of their decision.

7.2.2 Adversarial Attacks - XtoY

We will now describe the second kind of attacks: adversarial attacks. Since the Scouter models do not predict any intermediate concepts and do not consist of multiple steps, we will only describe the ***X to Y***

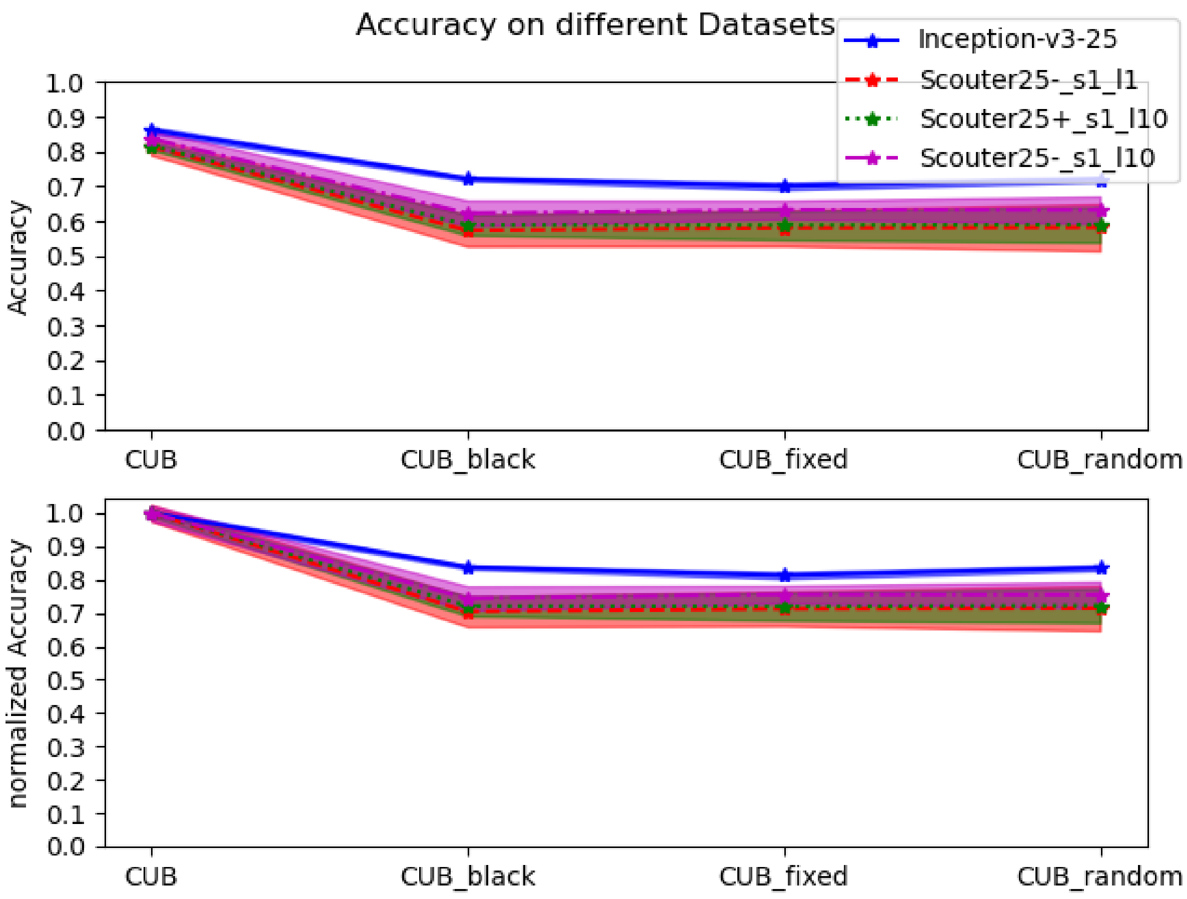

Figure 7.7: Top: the absolute accuracies for the first two Scouter models on the different datasets. Bottom: the normalized (relative) accuracies for the first two Scouter models on the different datasets.

case here, but we will describe the results for the initial two models and for the three variations again. We will not show the normalized graph of the initial two scouter models and of the variations here since there is no notable difference to be seen in this case and all relevant observations can be done by using the absolute accuracies.

The results of the adversarial attacks on the two initial scouter models can be seen in fig. 7.8. We can see some interesting features here. First, let's start with FGSM. For very small ϵ there is a strong decrease in accuracy for both, the Scouter models and the baseline. Interesting here is that both Scouter models build a plateau for a few ϵ values here up to $\epsilon = 0.05$ before their accuracy decreases again and anneal to the adversarial accuracy of the baseline. Notable is also that the baseline model performs worse for all ϵ values of FGSM.

The behavior of the models attacked by BIM is quite similar for all Scouter models. While there is a very small slowdown in the decrease of accuracy, all models reach an accuracy of close to 0% within the first six ϵ values up to $\epsilon = 0.025$.

For PGD, the behavior is slightly different compared to BIM. Here the accuracy initially drops very fast too, but then it stagnated for the Scouter models on 10% for Scouter25- and on 16% for Scouter25+ without decreasing any further. The accuracy of the baseline model does not fall as fast as in beginning, starting with an $\epsilon = 0.001$ but the accuracy still slowly decreases to 0%.

For SparseFool there is an initial decrease in accuracy to roughly 70% for the baseline, 55% for Scouter25+

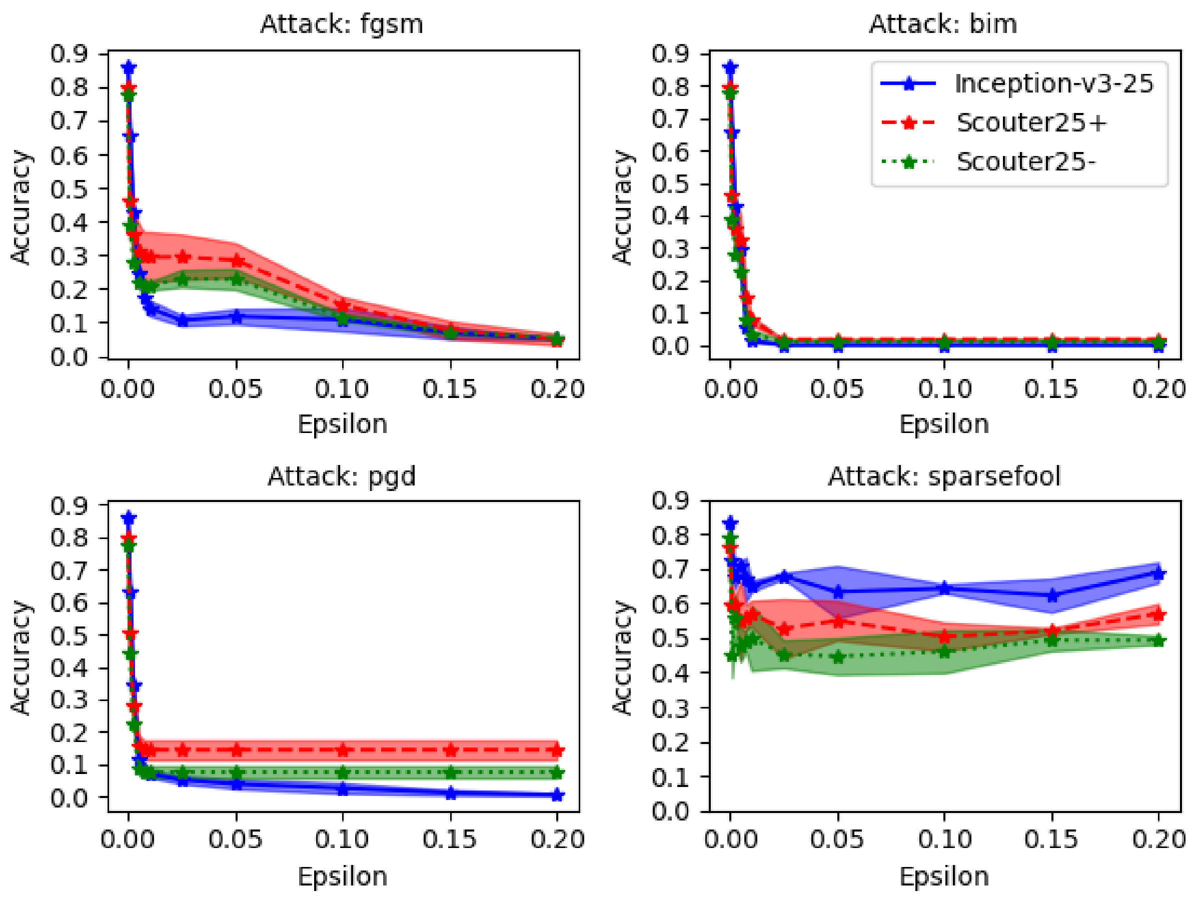

Figure 7.8: Results of the adversarial attacks on the first two Scouter models of setup *XtoY* for all ϵ values. Task: Attack image **X** to change class predictions **Y**.

and 47% for Scouter25-. After the initial decrease, the accuracies of all models oscillate around these accuracies.
When looking at 7.9, showing the results of the Scouter variations, one can again see a very similar behavior compared to the first two Scouter models. We will therefore again refrain from describing these results and pointing out the one major difference with takin a look at 7.10, which displays the results of all Scouter models we trained and attacked with an adversarial attack. In this figure, one can see that the Scouter25- model using one slot per class and $\lambda = 10$ performs best among all Scouter models for FGSM, BIM and except for one ϵ also for SparseFool. For the models under attack of PGD the behavior of all Scouter models using only one slot per class is different from the first two Scouter models. Here the accuracy decreases further for the models using one slot per class, leading to the Scouter25- version with one slot per class and $\lambda = 10$ performing best up to an $\epsilon = 0.05$, then the Scouter25+ using 5 slots per class and a $\lambda = 10$ performs the best for the remaining ϵ values.

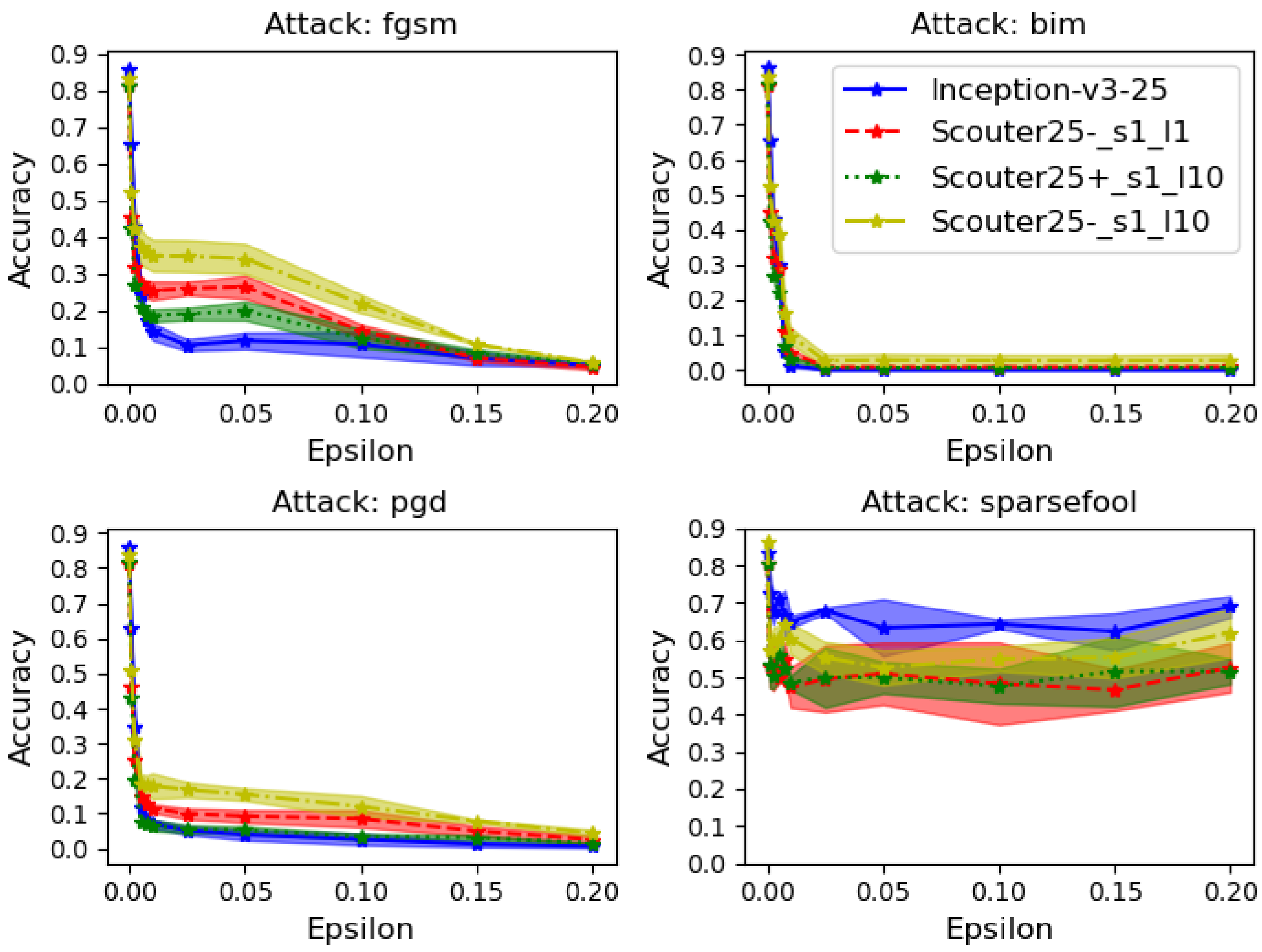

Figure 7.9: Results of the adversarial attacks on the Scouter hyperparameter variations of setup *XtoY* for all ϵ values. Task: Attack image **X** to change class predictions **Y**.

Summary

We use fig. 7.10 to analyze the results of all scouter models. In this figure, we show all the results calculated for Scouter and the Inception-v3-25 baseline. Using Scouter over the Inception-v3-25 baseline seems to be a good idea, since there are different variants of Scouter, which all are more robust to adversarial attacks, than the baseline model. The most robust model to adversarial attacks is *Scouter25- using one slot per class and a* $\lambda = 10$. For Scouter models it seems that there is no clear rule on whether positive or negative supports are more robust to adversarial attacks, but that the best performing model choice is solely dependent on the weight initialization, since the models using positive and negative supports are never grouped together in the graphs of fig. 7.10.
It is not possible to say is that one type λ values seem to be more robust to adversarial attacks, because there is no clear grouping on models with a certain λ value, which would support this statement. The same uncertainty hold for the number of slots per class, since in some cases higher numbers of *slots per class* show better results and in other cases lower numbers of *slots per class* show better results. The same goes for positive and negative supports.

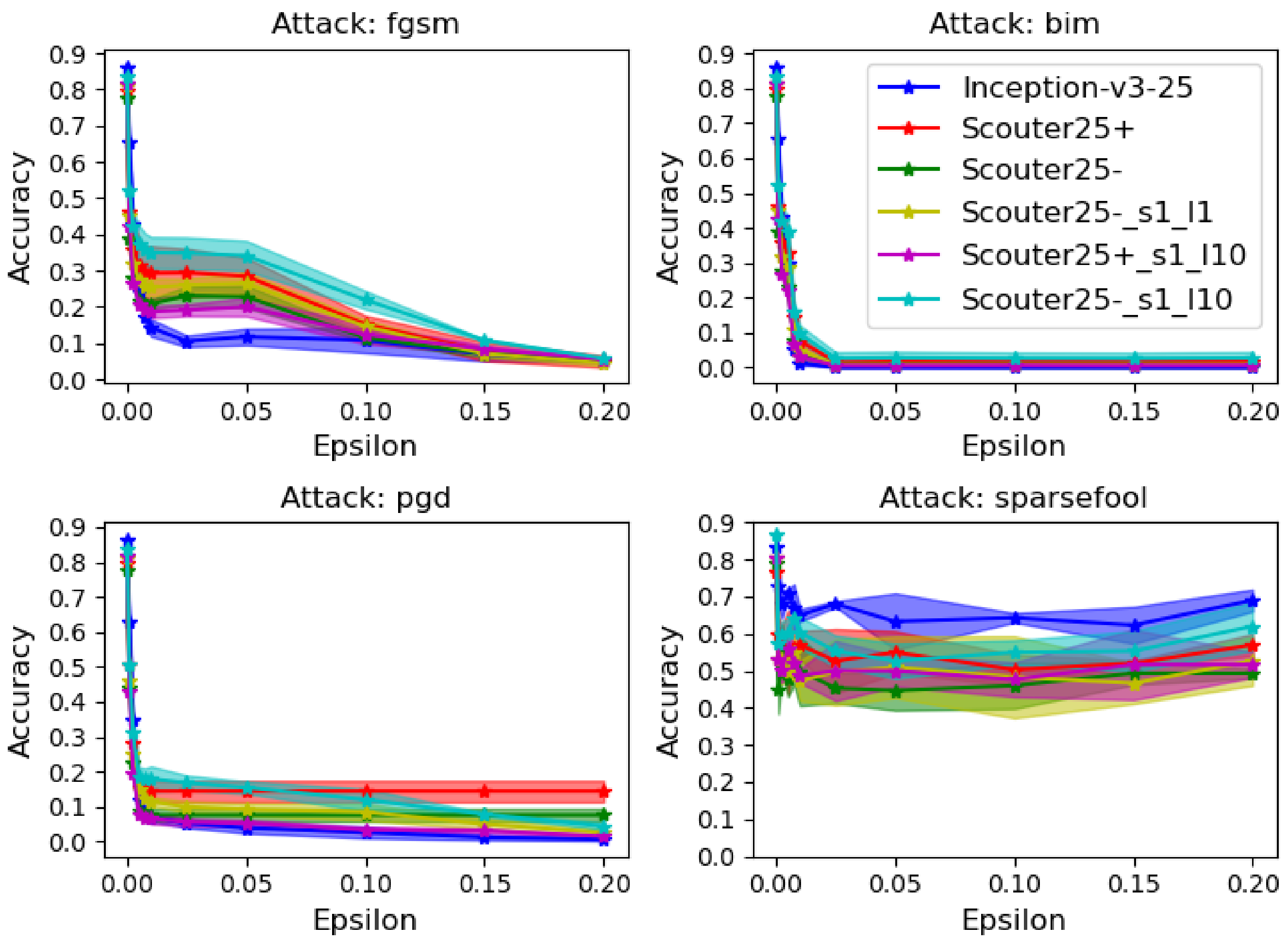

Figure 7.10: Results of the adversarial attacks on all Scouter models of setup *XtoY* for all ϵ values. Task: Attack image **X** to change class predictions **Y**.

7.3 Inception-v3-black Model

With the results of the Scouter models, we described the second large group of models we investigated. By presenting and describing the results of the adversarial and out-of-distribution attacks on the Inception-v3 CUB-black model, we will in this case describe the results of only one model. But as already stated, we will use this model as a baseline in the later part of this thesis.

7.3.1 TravelingBirds

The graph in fig 7.11 shows the accuracy of the model on the different datasets. We emphasize again, that the in-distribution dataset for this model is CUB-black while CUB is one of the adversarial datasets due to the training on the model on segmented images, which contain the birds on a black background. The initial accuracy on the in-distribution model is 80%, while the accuracy on the out-of-distribution datasets is 29% on CUB and 17% on CUB-fixed and CUB-random.

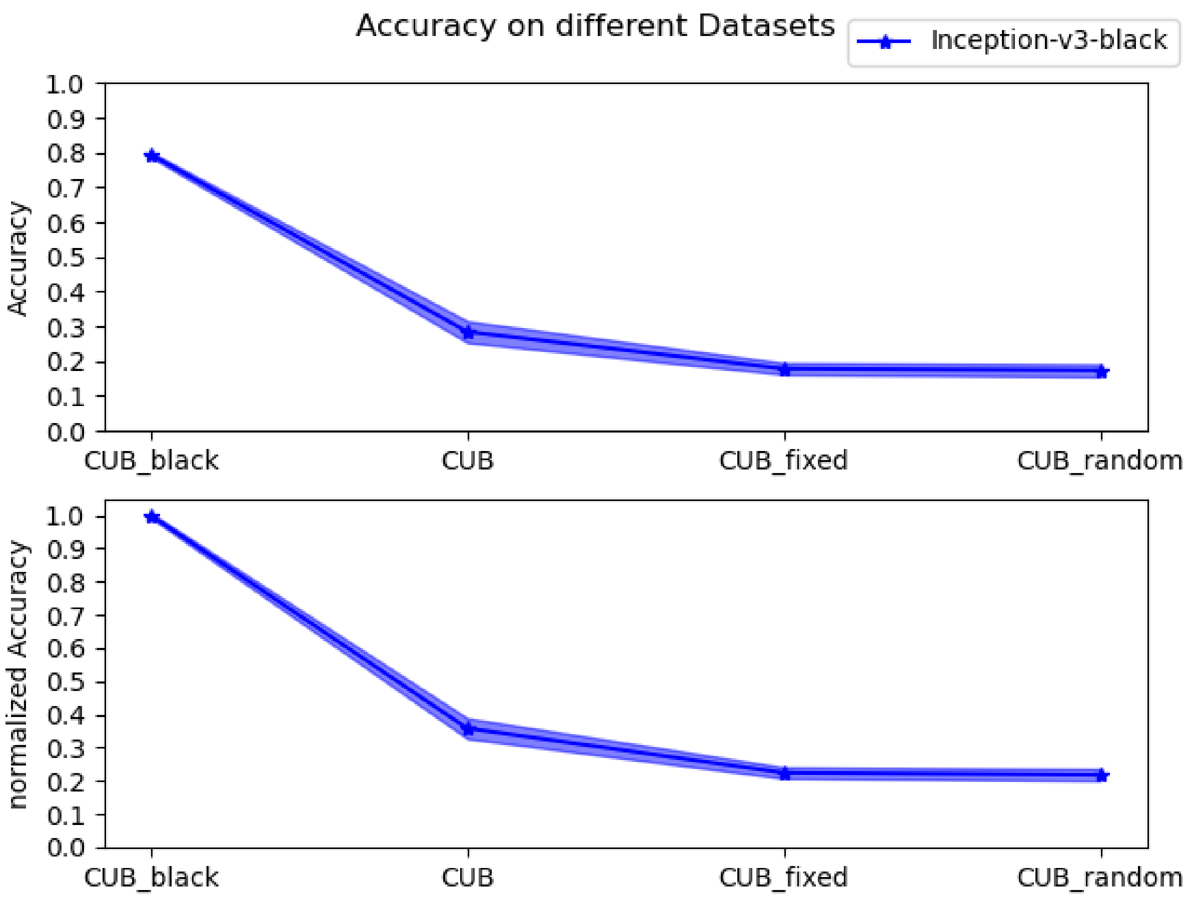

Figure 7.11: Top: the absolute accuracies for Inception-v3-black on the different datasets. Bottom: the normalized (relative) accuracies for Inception-v3-black on the different datasets.

Summary

This decrease in accuracy is very large, especially compared to the previous results of the Inception-v3 baseline, we described in section 7.1. The difference in accuracy between the models for the in-distribution datasets is quite small with it only being 3% smaller for Inception-v3 CUB-black.
We can assume from this difference, that the background itself plays a subordinate role for the model when being trained on the dataset and that the model is able to find good and reliable features for both cases. The low accuracy for the Inception-v3 CUB-black model on the other hand shows in the out-of-distribution test, that the background information is still taken into consideration to predict the bird classes. Since the shift of the background is very large in this case, changing from a black background to different environments, where none of them is black, is a difficult task for this model. We can explain the slightly higher accuracy of the model on the CUB dataset with the circumstance that many images in CUB have a blue background, which is somewhat constant and related to a fully black background, while the backgrounds in CUB-fixed

and CUB-random vary a lot more drastically.

7.3.2 Adversarial Attacks

Inception-v3 CUB-black is an end-to-end model without an intermediate step of predicting concepts. We therefore only have the ***X to Y*** setup for the adversarial attacks. The results of this setup are shown in fig. 7.12. All three attacks, which are changing all pixel values in an image show an equal pattern. The accuracy of the model decreases very fast with increasing ϵ values to close to 0%. The only one of the three attacks, where the decrease is slightly lower is FGSM, where the accuracy decreases more slowly, starting with $\epsilon = 0.005$.
SparseFool in the bottom right of fig. 7.12 shows a similar pattern compared to the other ***X to Y*** setups. Initially the accuracy falls fast with the first perturbations of the image, then the accuracy is oscillating around a certain mean accuracy, which is 50% in this case.

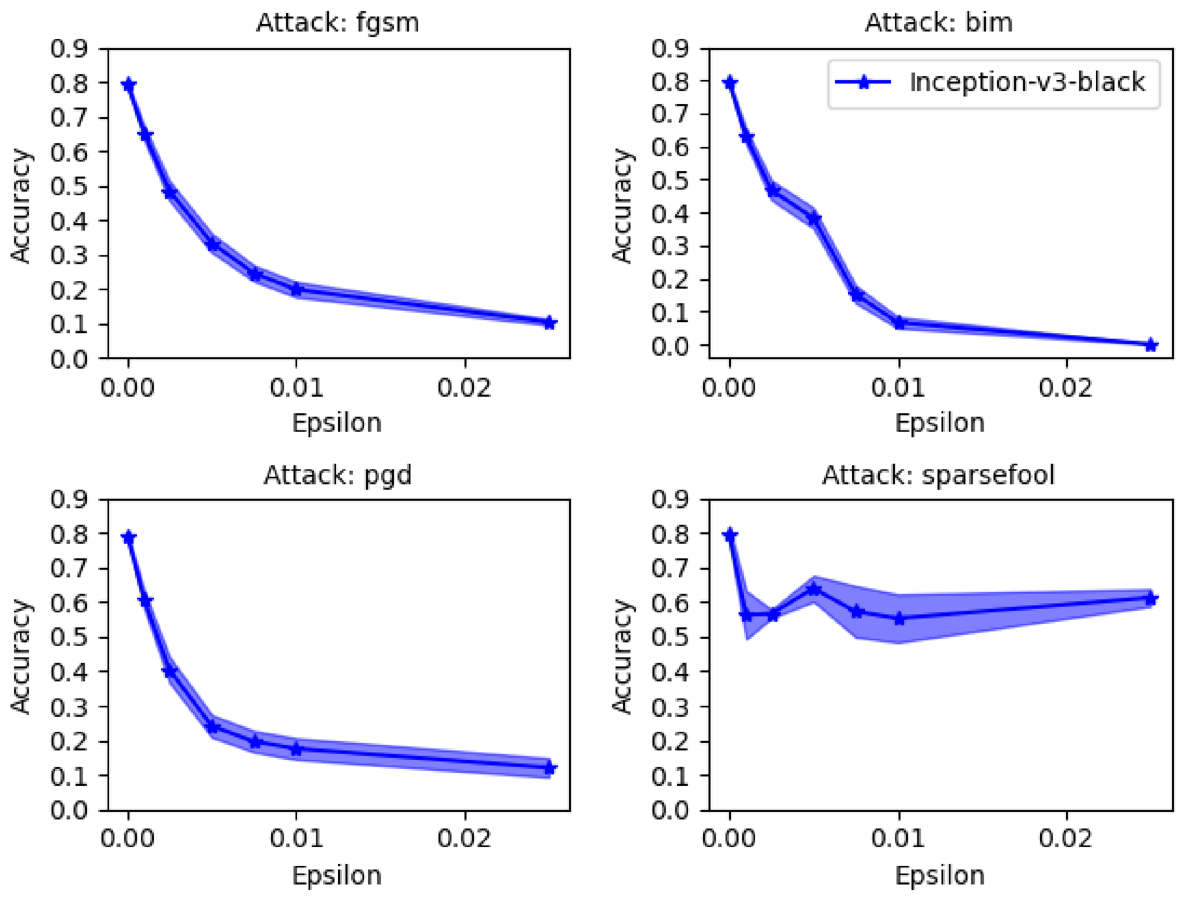

Figure 7.12: Results of the adversarial attacks on Inception-v3-black. Task: attack image **X** to change predictions **Y**

Summary

The results of this model show strong similarities to the Inception-v3 baseline for all four attacks. We will therefore not analyze these results any further.

7.4 Vision Transformer

As we described earlier, we fine-tuned and tested three different versions of Vision Transformers (see section 5.5), first a basic version (*ViT base*) using a patch size of 16, secondly a small version of a Vision Transformer using patch size of 16 (*ViT small p16*) and finally another small version of a Vision Transformer using a patch size of 32 (*ViT small p32*). All three of these models use an image size of 224x224 pixels. For all results we also added the results of the baseline model.
We will again first describe the TravelingBirds attacks and follow it up by the adversarial attacks.

7.4.1 TravelingBirds

Regarding the results in fig. 7.13 (top) one can see that the results of *ViT base* and *ViT small p32* are very similar on CUB with an accuracy of 51% and on CUB-black with an accuracy of 32%. The accuracy of *ViT base* then decreases more than the accuracy of *ViT small p32* to an accuracy of 14% for both CUB-fixed and CUB-random, while the accuracy for *ViT small p32* only drops to 18% for CUB-fixed and CUB-random.
ViT small p16 achieves the best results among the Vision Transformers we tested by achieving an accuracy on CUB of 70%, which drops lower to 56% on CUB-black and 30% and 29% for CUB-fixed and CUB-random.
For all ViT models, there is a smaller decrease in Accuracy between the in-distribution dataset CUB and first out-of-distribution dataset CUB-black, while the decrease of accuracy to the out-of-distribution datasets CUB-fixed and CUB-random are much larger.
All Vision Transformers we tested, achieve a worse accuracy than the baseline Inception-v3 model, both for the in-distribution dataset as well as for the out-of-distribution datasets in absolute and relative accuracies (compare with fig. 7.13).

Summary

All Vision Transformers we fine-tuned, performed worse on the different datasets than the Inception-v3 baseline. Since we were at that time – to our knowledge - the first ones to fine-tune a Vision Transformer on the CUB dataset, we could only refer to the very newly released publication (Steiner et al. 2021) to get some information about best practices for retraining and fine-tuning Vision Transformers. While Steiner et al. (2021) achieved better results on finetuning Vision Transformers on different datasets, we also had to deal with the difficulty, that CUB is a very small dataset compared to most of the chosen ones tested by Steiner et al. (2021), especially for large models such as the Vision Transformers are. Interestingly enough, the best performance of the Vision Transformers across all datasets was provided by the smallest of the models, although there is still a large decrease in accuracy on the out-of-distribution datasets. At this point we assume that all Vision Transformers memorized the training images in some way without really

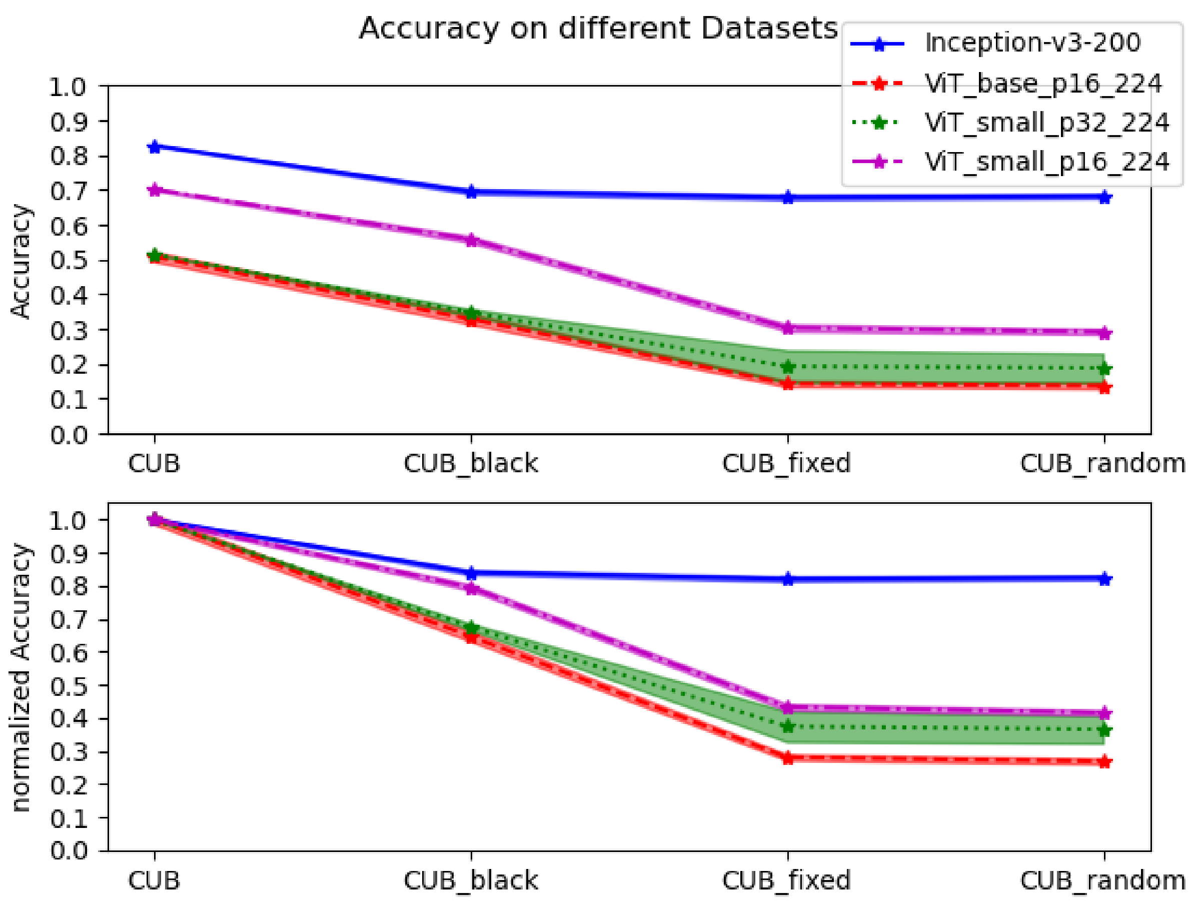

Figure 7.13: Top: the absolute accuracies for the Vision Transformers on the different datasets. Bottom: the normalized (relative) accuracies for the first two Scouter models on the different datasets.

learning specific features of the different bird species. This assumption is supported by the observation that the smallest one of the models, which we can assume as the worst at memorizing the images due to the smallest number of parameters, achieves a lot stronger results on the already known distribution of the CUB dataset.
Additionally, all models performed quite badly on out-of-distribution datasets. This might therefore also be the correct assumption here, especially since the drop in accuracy is lower on CUB-black than for the CUB-fixed and CUB-random datasets. For the two latter datasets, the shift of the distribution is larger, because the backgrounds of the images are a lot more different from CUB than the background of CUB-black is. We remind here that CUB oftentimes has a blue background coming from sea or sky, while CUB-fixed and CUB-random have backgrounds from various locations, with many of them representing places or indoor locations and are therefore strongly different from CUB.

7.4.2 Adversarial Attacks - XtoY

The results of the Vision Transformers on the different adversarial attacks will be presented in this section. Since Vision Transformers are end-to-end models, we will again have only the *XtoY* setup, as we had it for

the Inception-v3 versions and for the Scouter models.
The experimental results for the three Vision Transformer models and the Indeption-v3 baseline can be seen in fig. 7.14. For all three of the adversarial attacks *FGSM*, *BIM* and *PGD*, which perturb every pixel of an image, the results are similar. All versions of Vision Transformer are decreasing their classification accuracy very fast to 0% when being attacked by one of these adversarial attacks.
A different behavior can be seen for the results of *SparseFool* (fig. 7.14 bottom right). Here all three Vision

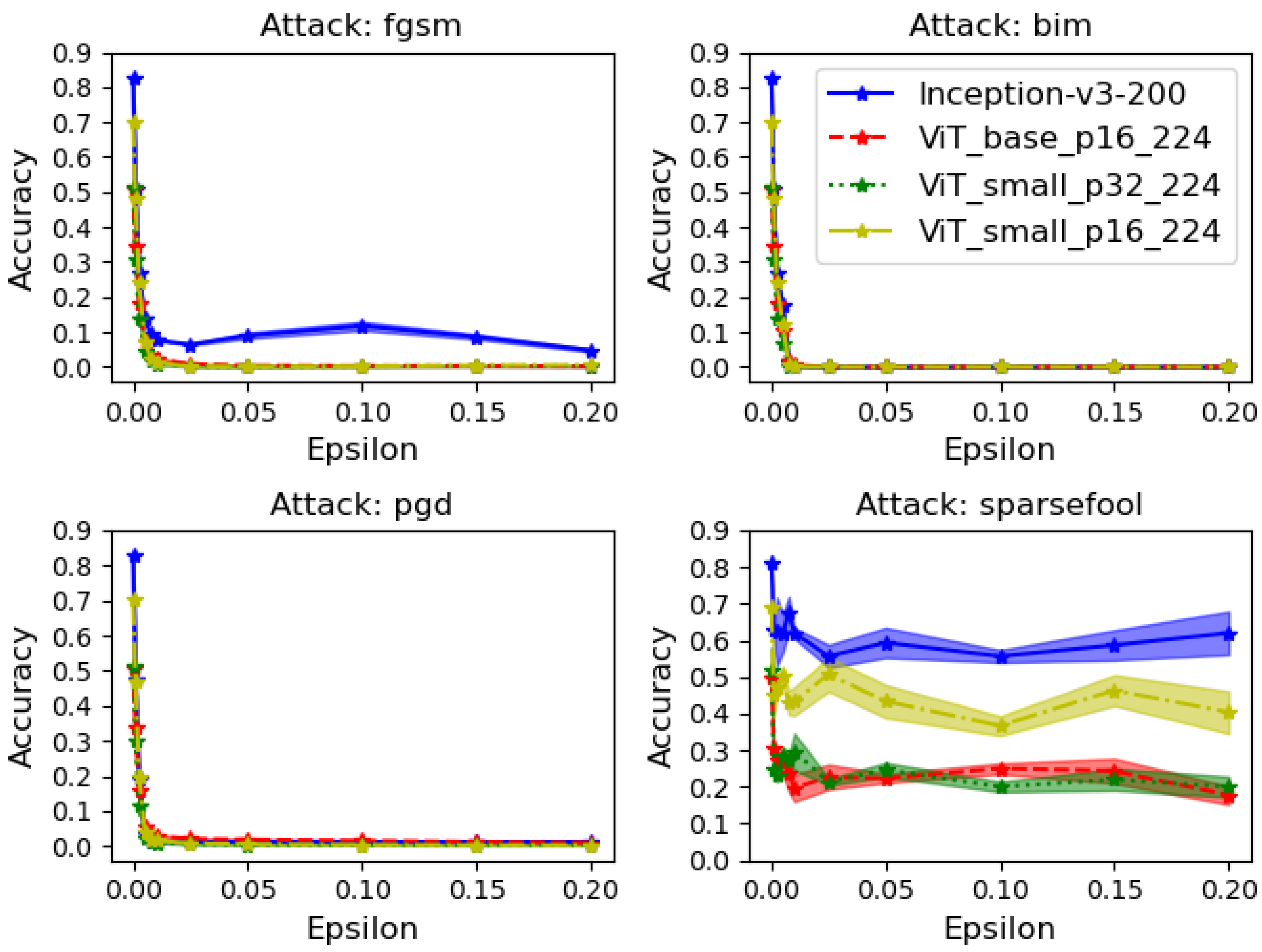

Figure 7.14: Results of the adversarial attacks on the Vision Transformers of setup *XtoY* for all ϵ values. Task: Attack image **X** to change class predictions **Y**.

Transformer models drop 20% in accuracy for the first $\epsilon > 0$, similar to the Inception-v3 baseline, and then oscillate around a mean accuracy of 45% for *ViT small p16* and 24% for *ViT base* and *ViT small p32*. When being attacked by *SparseFool*, the best performing Vision Transformer is again *ViT small p16*, while *ViT small p32* and *ViT base* both achieve very similar results compared to each other, as one can see in fig. 7.15. Here only *SparseFool* shows meaningful results again, due do the very fast drop of the accuracy to 0% for the other three adversarial attacks. The bottom right graph of fig. 7.15 shows that, relatively seen, the oscillation of *ViT small p16* is larger than the one of the two other Vision Transformer models, but it is also at a higher level, which is sometimes close to, or even better than the Inception-v3 baseline, although the absolute accuracy of *ViT small p16* on CUB without perturbations is 14% lower than the one of the baseline.

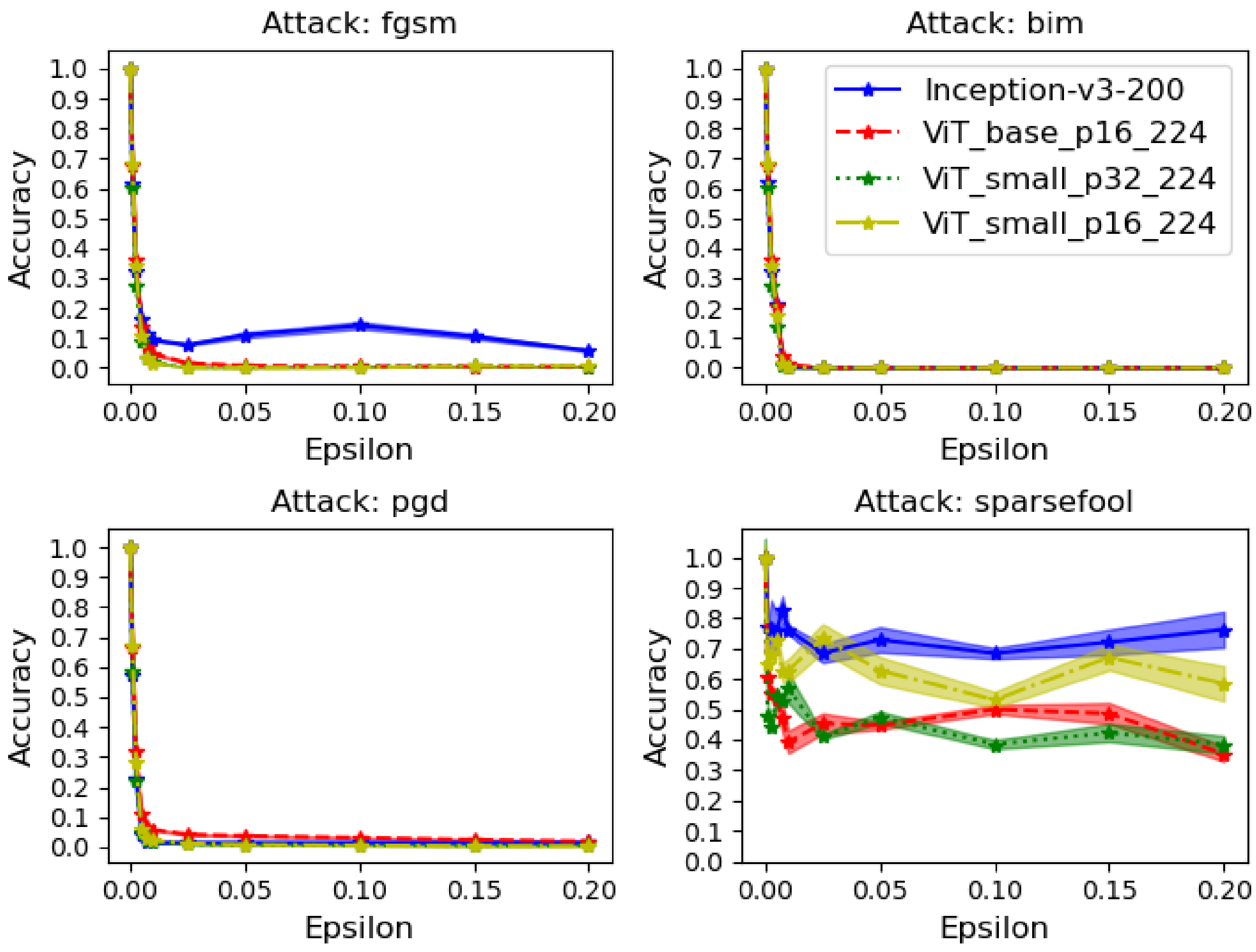

Figure 7.15: Normalized results of the adversarial attacks on the Vision Transformers of setup *XtoY* for all ϵ values. Task: Attack image **X** to change class predictions **Y**.

Summary

As it was for the TravelingBirds out-of-distribution attack, the Vision Transformer models perform quite bad on the different adversarial attack setups. While *ViT small p16* still achieves reasonable results in some cases by just performing a little bit worse in the relative results compared to the Inception-v3 baseline (e.g., on CUB-black for the TravelingBirds setup and on the SparseFool attack), the other two models, which either have a larger patch size in case of *ViT small p32* or have much more capacity like *Vit base*, yield very bad results.
As it seems, the previous statement of larger Vision Transformers mostly memorizing the training data rather than learning features of the given images seems to hold again for the Vision Transformer models, which might make them a bad choice of a model to use on smaller datasets.

7.5 Slot Attention & Set Transformer

As our first own attempt at developing a model, which achieves an inherent robustness to out-of-distribution attacks as well as to adversarial attacks, we modified the Slot Attention model to be able to predict attributes based on the current slot. Unfortunately, we were not able to generate reasonable predictions during the training and evaluation phase. While we were able to generate reasonable results on the independent training of the Set Transformer model, which is trained by putting in the annotated attributes of the CUB dataset expanded by adding a background slot indicated by all zeros, we were not able to predict such concepts using the concept model.
We might assume multiple reasons on why we were not able to get reasonable results despite a validation accuracy of 85% for the concept model. At first, we saw, that the annotated and reduced 112 attributes of CUB were highly imbalanced with roughly 20.5% of the attributes being present, while 79.5% of the attributes not being present, and only 10.2% of attributes being present in the version with using two slots, where one slot is intended for foreground slot containing the bird in the foreground and the second slot is intended for representing the background.
We used several different attempts of a loss function to enable the model to handle this imbalance. After first using the Hungarian loss based on the smooth-L1-loss, our second attempt was to use the binary cross entropy loss used in Koh et al. (2020). In both cases we were not able to get reasonable results. Here the true negative rate was very good with being at 0.999, but the false negative rate was at 0.973, which indicated that the model found the best possible optimization for the loss in simply predicting all concepts as not present. Using this solution for the optimization, we were not able to predict any bird class in a reasonable way due to the lack of information. Now, we adapted the loss by including parts of the confusion matrix, namely by first adding the false positive and false negative rate to the loss, to give the model an indication to also minimize these two. While this change was better, it was not enough to increase the accuracy and the true positive rate by a lot.
Our best attempt at training such a model was with using the binary cross entropy loss as described before and then applying the function $loss_{total} = loss_{BCE} + fp + 2 * fn$ to it. Here the combined loss is the sum of the binary cross entropy loss, the false positive rate and two times the false negative rate.
By using this loss, we were able to increase the true positive rate of the concept part of the model to around almost 25% and we tried to increase it by varying different hyperparameters such as number of hidden layers for the Slot Attention module, layer size of the layer right before the bottleneck layer and by varying the size of the CNN backbone. While all these changes resulted in varying true positive rates on the validation set, we were not able to increase the true positive rate above 0.25 and therefore we were not able to properly train the second part of the mode or to get reasonably well predictions from it.
Since further testing and experimenting to enable this architecture to work on real datasets would exceed the scope of this thesis, we stopped at this point and left further research for future work.

7.6 Mask Bottleneck Model

Unfortunately, we were not able to train a combination of Slot Attention and Set Transforme, which provides at least acceptable results. We therefore fell back to a related idea of Yi et al. (2018). Their idea is to use object detection as a first step of a so called *scene parsing* module to extract the objects present in the scene. After extracting the objects, they put each object into a second stage to predict several features of each

object to take further steps at their task.
Since the first test run of the Mask R-CNN trained on the Microsoft COCO dataset (Lin et al. 2014) on CUB yielded bird predictions and segmentations for more than 99% of the images of CUB when using a certainty threshold of 0.3 (more than 98.6% for a threshold of 0.5) as a prediction score, we did not finetune Mask R-CNN, but we used the already trained version. After passing the images through this first step of the pipeline, we processed the outputs to get different images through different ways of cropping (bounding box and segmentation borders) and additional segmentation of these cropping possibilities. More details on the process of cropping and segmentation after the first stage of the pipeline can be found in 5.7. Now, we added a bottleneck network for the second stage of the pipeline. This bottleneck network consists of an adapted ResNet-34 (He, X. Zhang, et al. 2015) as concept model and a fully connected layer as the final part of the model.

7.6.1 Training Results

In this section we will present the training results of our Mask Bottleneck model. As we already stated, the first stage of the pipeline, namely the Mask R-CNN, was able to predict a *bird* label in 99% of the images of CUB by using a threshold of 0.3 as a minimum certainty score. We therefore refrained from finetuning it, because the task of the Mask R-CNN is to detect the bird and narrow the focus of the image to it by cropping (and potentially segmenting) the image around the bird to make it easier for the second stage of the pipeline to predict the bird species.
Regarding the second stage of the pipeline, we trained a model for every case, we got from the processing step after stage one. These four cases are cropping around the predicted bounding boxes (*cropbb*), cropping around the borders of the segmentation mask (*segbb*), and apply the predicted segmentation mask to the image before using one of the cropping types (*cropbb_useseg, segbb_useseg*).
For each of these four cases, we trained an adapted ResNet-34 (He, X. Zhang, et al. 2015) as concept model and we trained the fully connected layer, which predicts the bird classes from the concepts for five cases. The first four cases were the *sequential* training, training one fully connected layer for each of the concept models. The fifth case is the *independently* trained fully connected layer, which was trained on the annotated attributes of CUB. This *independent* version of the fully connected layer can be placed at the end of each of the concept models without retraining it.
For the following results, we will always append the name of the crop type and possibly the postfix *_useseg* to the model's name (e.g., Independent_cropbb) to indicate the version of the concept model. But for all cases, if *Independent* is in the name of the model, it is always the same fully connected layer.
The training results can be seen in table 7.2, where the best results are marked in bold letters. While the accuracy of the concept models (*XtoC*) is very high at 93.7% for the best performing model of the four cases and the lowest at 92.2%. For the *CtoY* models, which consist of a fully connected layer, the accuracies for *cropbb* models are between 60.2% and 61.8% and for the *segbb* models, the accuracies are between 52.5% and 55.6%. We will note here that we showed the results for all *CtoY* models on the predicted concepts, although the *Independent* model is the *same model for every case*, which we trained only once on the annotated attributes. The *Independent* model achieved an accuracy of 100% on the validation set during the training, but it was expected that the real accuracy is lower due to the non-ideal setup coming due to the inaccurate concept predictions.
After training these models, we applied the different attacks to the Mask Bottleneck pipeline, as we already described in section 6.2.10. We will describe these attack results in the next sections, starting with the attacks on the first stage (Mask R-CNN) and the forward pass through stage two and ending with the

Model name	Type	XtoY acc	XtoC acc	CtoY acc
Mask Bottleneck cropbb	Sequential	**0.612**	**0.937**	**0.618**
	Independent	0.605		0.608
Mask Bottleneck segbb	Sequential	0.548	0.925	0.556
	Independent	0.542		0.547
MaskBottleneck cropbb use-seg	Sequential	**0.597**	**0.935**	**0.602**
	Independent	0.588		0.593
MaskBottleneck segbb use-seg	Sequential	0.519	0.922	0.532
	Independent	0.509		0.525

Table 7.2: The training results of the Mask Bottleneck model. The best results for each individual task are marked in bold.

adversarial attacks on the stage two models.

7.6.2 Stage 1 Attacks & pass through stage 2

Four kinds of values were calculated for the stage one attacks. We will use these values on both the out-of-distribution attack, and we will use these values again later for the adversarial attacks on Mask R-CNN. These values state the certainty of the model, that a bird is in the image, and the certainty for the respective segmentation mask. We tracked the *top 1 certainty*, this is the mean of the highest certainty corresponding to a “bird” label, the *Average birds found* certainty, which tracks the mean certainty across all possible bird predictions in an image above a threshold of 0.3 across the respective test set. Finally, we tracked the *Average all images* certainty, which calculates the certainty of all *bird* labels found in an image across all images, but this value also considers images without a *bird* label having a certainty above the threshold of 0.3 with a value of zero.

TravelingBirds

Stage 1 In this paragraph we will describe the results of passing CUB and the three TravelingBirds datasets through stage one of the Mask Bottleneck Model, and afterwards we will also describe the following forward pass of the processed outputs from the Mask R-CNN through the second stage of the pipeline.
Fig. 7.16 and 7.17 show the results after passing the images through Mask R-CNN as the first stage of the pipeline. Note here again, that even CUB is an out-of-distribution dataset for Mask R-CNN, since we did not fine-tune the original version, which was trained on Microsoft COCO (Lin et al. 2014) due to its good results on CUB. The first figure shows three kinds of certainties regarding the bird predictions in images.
We will start with fig. 7.16 displaying the three different certainty scores. The *top 1* certainty is very high with a score above 92% across all datasets. The *top 1* score is the highest for CUB (98.4%) and the lowest for CUB-random (92.6%). The *average found* is a little lower at 90% on CUB and 85% on Cub-random, while the *all images* score is starting at 90% on CUB and ending with a certainty of 79% on CUB-random. Mask R-CNN was not able to find a bird in 32 images on CUB, in 206 images on CUB-black, in 354 images on CUB-fixed and 371 images on CUB-random (compare to fig. 7.17). Considering that the datasets all consist of 5794 test images, this is a failure rate of 0.6% on CUB as the lowest failure rate. The highest

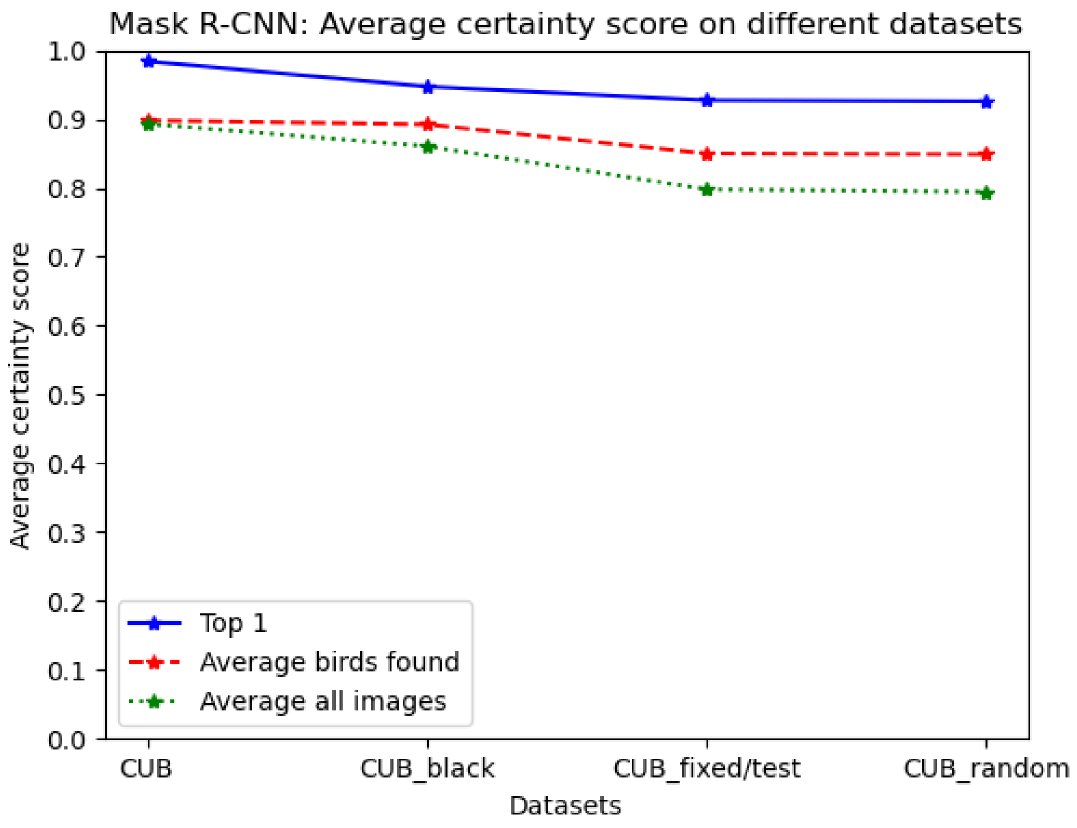

Figure 7.16: The average certainty score of Mask R-CNN on different datasets. A higher certainty indicates better results. *Top 1*: the certainty of the bird label with the highest score above a threshold of 0.3, *Average birds found*: the average score of all bird labels above a threshold of 0.3, *Average all images*: the average score of all bird labels above a threshold of 0.3. If no bird is found, the score for this image is weighted as 0.

failure rate on CUB-random is at 6.1%.

Stage 2 forward pass To present the results of the forward pass through stage two, we will calculate the accuracies according to the different types of certainty, we already presented for the TravelingBird out-of-distribution attack on the stage one model. These accuracies are the accuracies for: all images including *bird found* and *no bird found* images, which we will declare again as *all images accuracy*. Additionally, we will provide the results for both, images where a bird was still detected at stage one (*bird found accuracy*) and only the images, where no bird was detected after the TravelingBird out-of-distribution attacks at stage one (*no bird found accuracy*).
We will start with the *all image accuracy* for the models that are not using segmented images, which can be seen in fig. 7.18. The first interesting fact that is coming to an eye is that the *all image accuracies* of the *segbb* models (54% *all image accuracy* are constantly 7% lower than the *all image accuracies* (61% *all image accuracy* of the *cropbb* models for all datasets. Other than that, the results of the models behave equally.

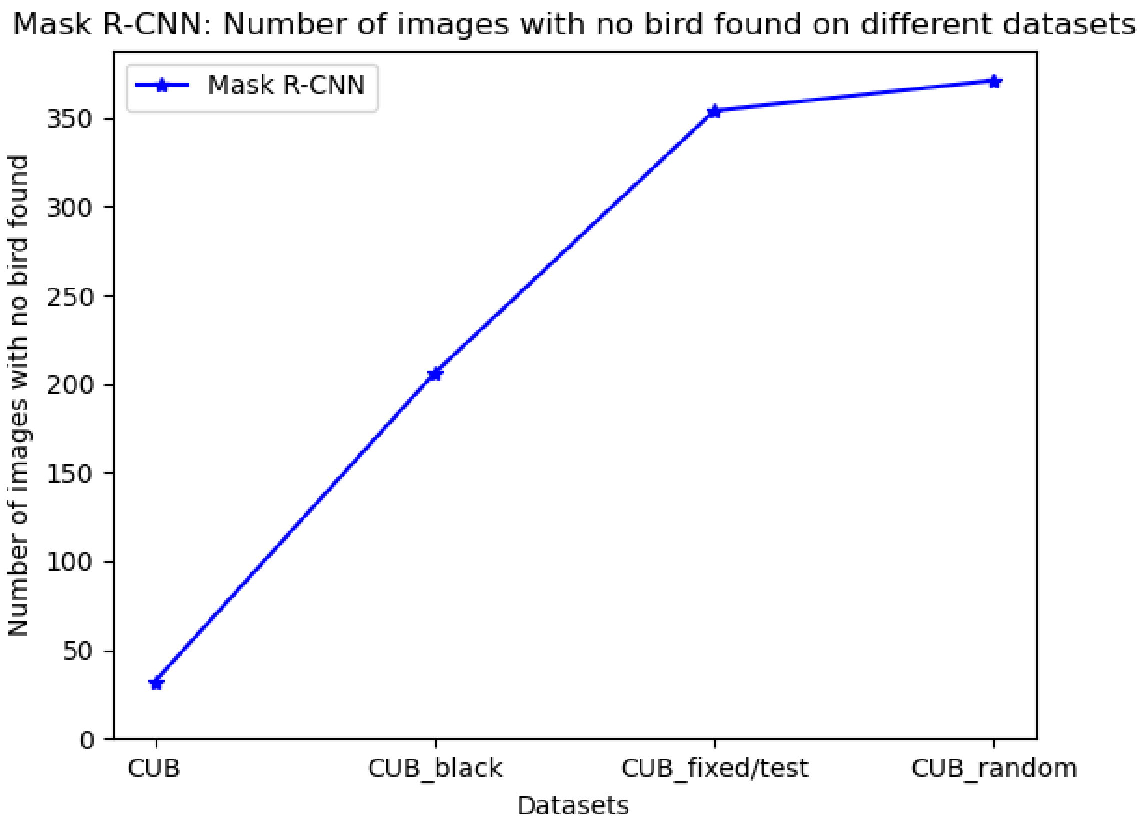

Figure 7.17: The number of images, without a bird label having a score above 0.3. A higher number indicates a worse result.

The highest accuracy for all models is on the in-distribution dataset CUB-cropped. The second highest accuracy, which is 8% lower for all models is the one for CUB-black and CUB-fixed and CUB-random have both the lowest accuracy, which is 10% lower than the accuracy of the in-distribution dataset CUB-cropped.
The graph for the *bird found accuracy* (fig. 7.19) is completely similar to the one showing the *all images accuracy* in fig. 7.18 except for deviations of 1%, which do not change the overall observations of this setup. Therefore, we will refrain from repeating the observations and move on to the *no bird found accuracy* directly.
Fig. 7.20 shows the *no bird found accuracy*. Throughout this section regarding the *Mask Bottleneck models*, we will not show these results on CUB-cropped since we removed these images from the dataset to provide clean training datasets for the stage two models, which do not contain these images.
The *no bird found* accuracy on the out-of-distribution datasets is quite low, being relatively constant on all three TravelingBirds datasets with an accuracy of 21% for *cropbb* models and 15% for *segbb* models. For *cropbb* models, the lowest *no bird found* accuracy is determined for the CUB-fixed dataset, being 2% lower than for the other two datasets. For *segbb* models, the worst performance is on CUB-random with an accuracy, which is 3% lower than on CUB-black, which is having the highest *no bird found* accuracy. The models, which are using an applied segmentation mask to the input images of stage two, show a closely related behavior compared to the models which are not applying the segmentation masks to the input images. We will therefore refrain again from repeating these results and point out the differences for the

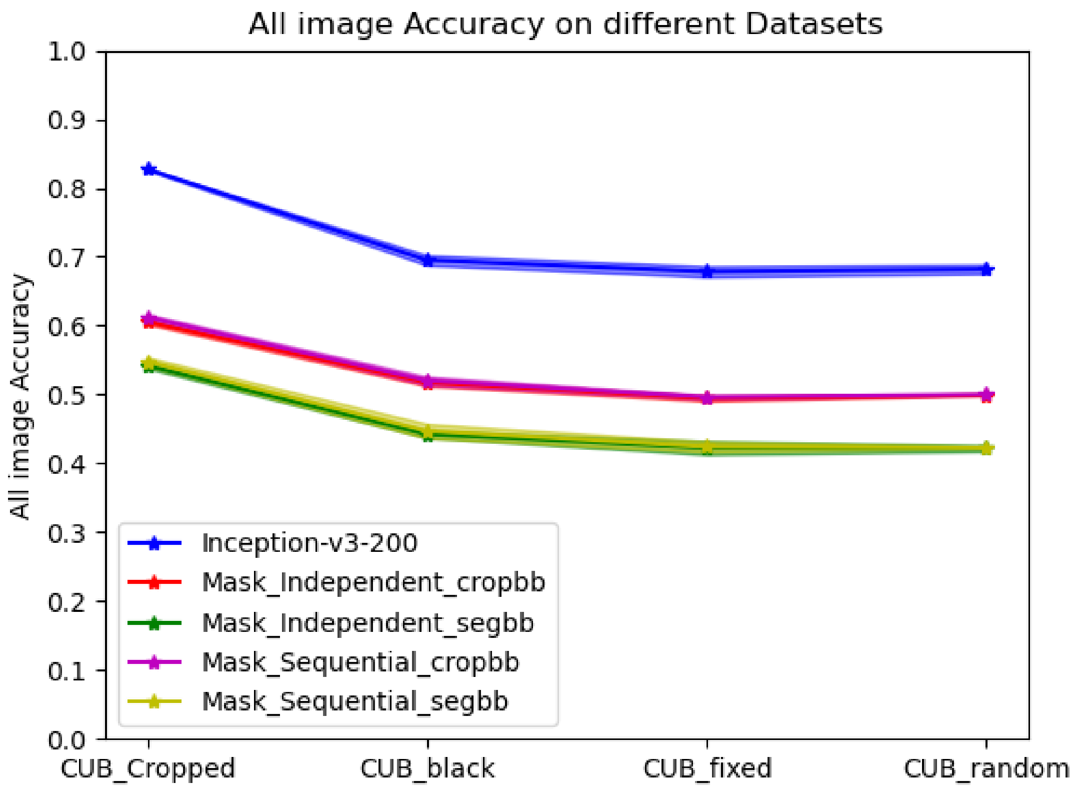

Figure 7.18: The *all image accuracies* of the forward pass of the out-of-distribution data in stage 2. Higher accuracies indicate better results. For the baseline, it is a normal forward pass for each image in each dataset.

all image accuracy (fig. 7.21) and the *no birds found accuracy* (fig. 7.22).
The very important difference is that the maximum decrease in *all image accuracy* is at 5% for each model across the different datasets. An additional very important point is that while the Inception-v3-black baseline provides better results (20% and 30% better accuracy) than the *Mask Bottleneck models* on the original dataset, the accuracy on the out-of-distribution datasets is very much lower, being between 35% to 40% lower for *cropbb* models (and 25% to 30% for *segbb* model). Fig. 7.22 shows the *no bird found accuracy* for models applying the segmentation mask to an image before cropping. The accuracy on CUB-black is the highest with 35% for *cropbb* models and 25% for *segbb* models. Otherwise the behavior is similar to the models not applying segmentation masks. The accuracy drops for CUB-fixed and CUB-random to a slightly lower accuracy of 15% and 9% for *cropbb* and *segbb* models.

Summary

We described the results of attacking the *Mask Bottleneck model* with the out-of-distribution TravelingBirds datasets. All the results showed that the full Mask Bottleneck model has a low susceptibility to wrong

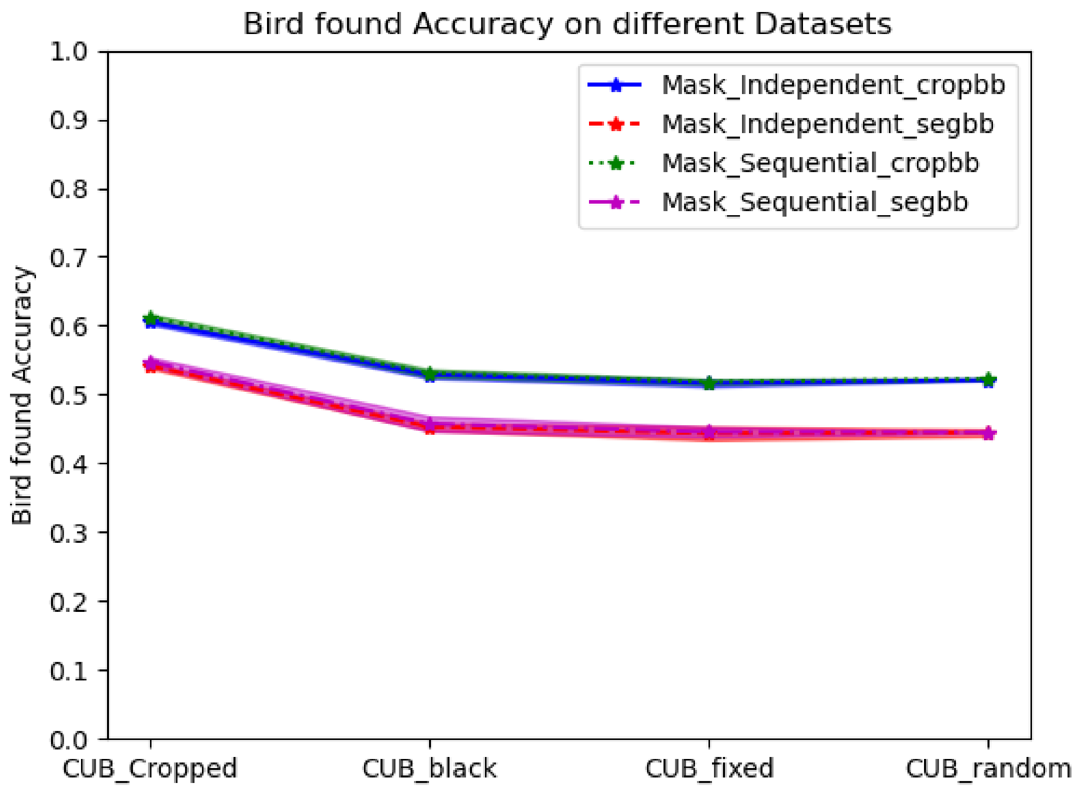

Figure 7.19: The *bird found accuracy* of the forward pass of the out-of-distribution data in stage 2. Higher values indicate better results.

predictions, when seeing data different from the already know training distribution.
The Mask R-CNN performs quite well on all out-of-distribution datasets, including CUB in this case, since we did not fine-tune the version of Mask R-CNN, which was trained on Microsoft COCO. The highest proportion of images, where no *bird* could be detected, was at 6.1%.
The second stage models provided similar results with low decreases in their classification accuracy on all images. We will summarize the small differences between the different processing variants as follows. The models applying segmentation masks before cropping the images achieve a classification accuracy on all images, which is with 2% slightly lower compared to the models not applying segmentation masks, but on the other hand the accuracy drop for models applying the segmentation masks is also lower, which provides very similar results on the out-of-distribution datasets.
All in all, our *Mask Bottleneck model* shows a very low susceptibility to out-of-distribution data. We will later discuss the implications on the research questions of our thesis in more detail.

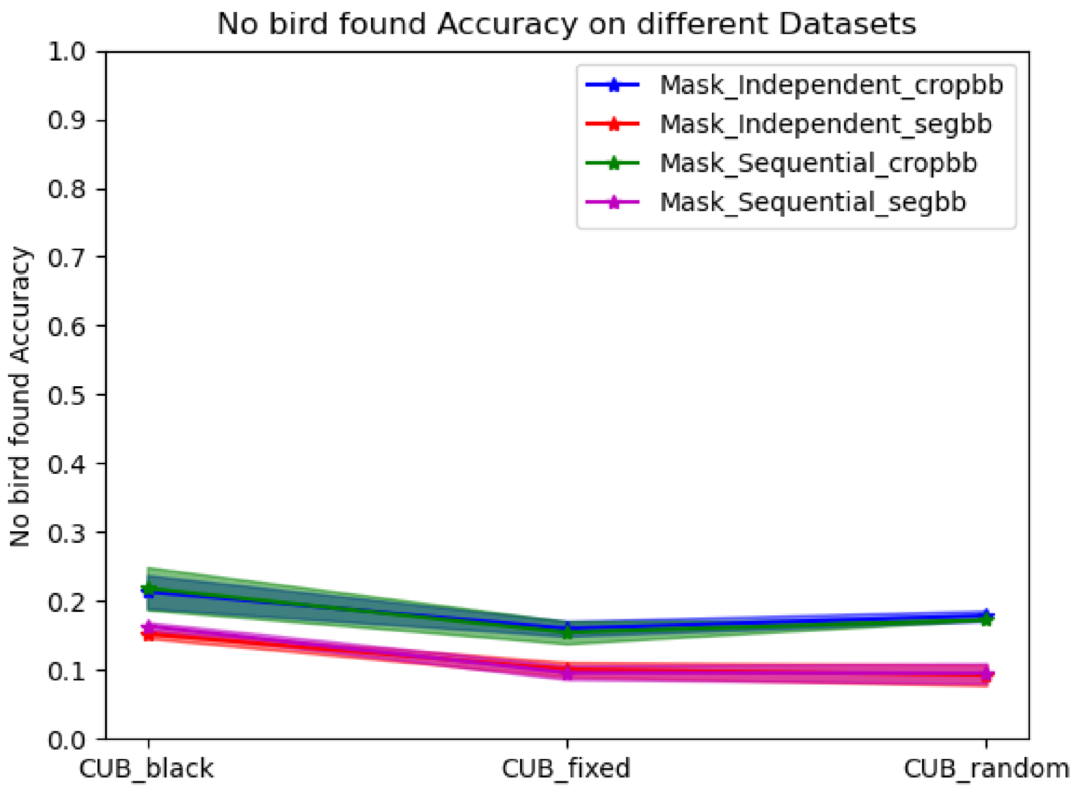

Figure 7.20: The *no bird found accuracy* of the forward pass in stage 2. These images were the unaltered input image, since the Mask R-CNN was not able to detect a bird label, therefore no cropping/segmentation was applied. For the baseline (blue) it is a normal forward pass of each image of each dataset. Higher values indicate better results.

7.6.3 Adversarial Attacks

In this section we will describe and later discuss the results of the adversarial attacks on Mask R-CNN. As we already stated earlier, we did use the three adversarial attacks *FGSM, BIM and PGD*, which we described in section 6.2. We also stated that we will not use *SparseFool* as an adversarial attack on Mask R-CNN due to the complexity of fitting the attack to the task of object detection instead of classification for Mask R-CNN.

Stage 1 We will use the same metrics, we also used for the TravelingBirds setup on Mask R-CNN. Fig 7.23 and 7.24 show the results of the adversarial attacks. Starting with fig. 7.23, one can see that the average certainty scores for all cases falls for FGSM and PGD.
The *top 1* score falls from 98% for non-perturbed images up to a score of 72% for $\epsilon = 0.2$. As it was in other attacking cases too, the score initially decreases faster and later goes to an almost linear decrease for larger ϵ values. The curve displaying the average *all images* score behaves similar to the *top 1* score

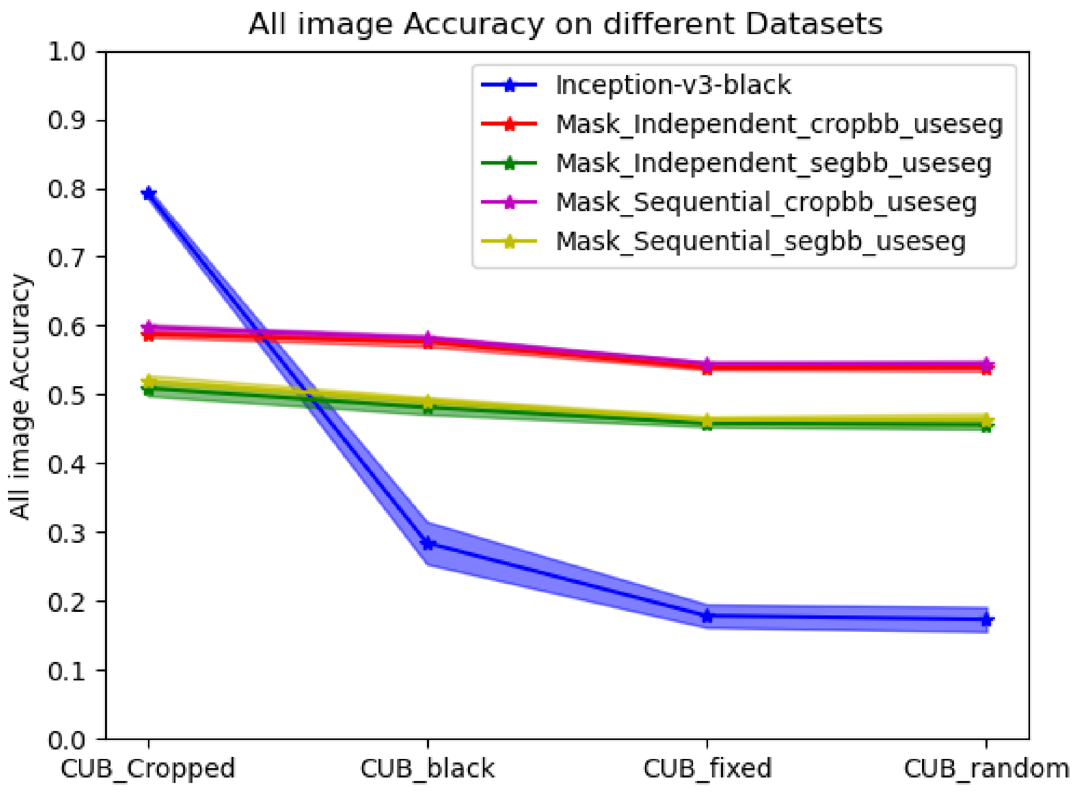

Figure 7.21: The *all image accuracies* of the forward pass in stage 2. In the processing step, a segmentation mask is applied before cropping. Higher accuracies indicate better results.

with the difference of starting at a lower percentage of 90% certainty and falling much further to a score of 33%. The *average found* score initially decreases from 90% certainty to 76% at $\epsilon = 0.01$, then almost stagnates up to $\epsilon = 0.1$ and then decreases slightly stronger again to a certainty of 65% at $\epsilon = 0.2$.
For BIM the *top 1* score as well as the *all images* score decrease initially up to $\epsilon = 0.025$, then they remain constant at their respective level of 93% for *top 1* and 59% for *all images*. The *average found* score behaves differently. Initially, the accuracy drops to 77% for $\epsilon = 0.0075$, then the *average found* score increases again up to $\epsilon = 0.025$ and remains constant afterwards for increasingly large ϵ.
The behavior of Mask R-CNN when being under attack by PGD is very similar for *top 1* and *average found* compared to the behavior of theses scores when being attacked with FGSM. The behavior of *all images* is different. Initially the certainty decreases very fast from 90% to 64% and then slows down the decrease to an almost linear decrease in certainty up to 24%.
The fourth metric we tracked was the number of images where no *bird* was found anymore. The results for this metric can be seen in fig. 7.24. Here the graphs for FGSM and PGD mirror the behavior of the average *all images* certainty score: the lower the certainty, the higher is the number of images with no bird found. The graph for images with no birds found when Mask R-CNN is attacked by BIM is different though. Here there is a sharp increase in images without a bird found up to an $\epsilon = 0.01$, which then slows down to $\epsilon = 0.025$ and remains constants for the remaining $\epsilon > 0.025$. Without any perturbations, Mask R-CNN is not able to find the “bird” in 30 images. For FGSM the number increases up to 2875 images, for BIM, the

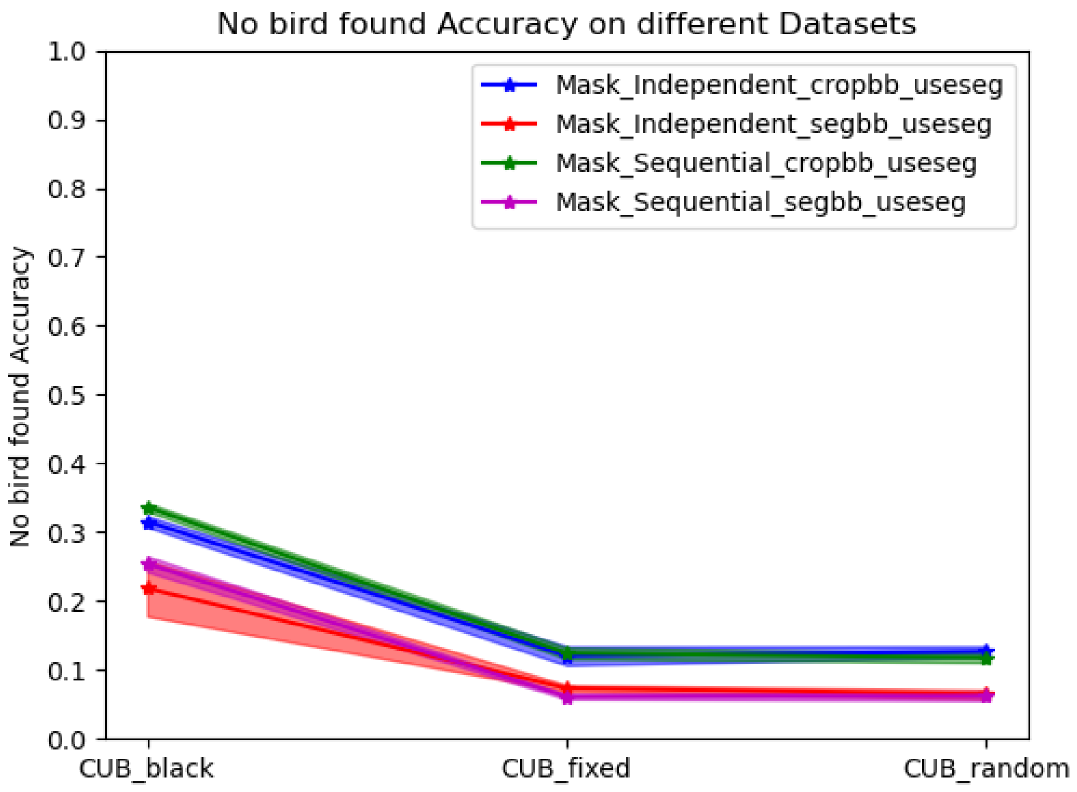

Figure 7.22: The *no bird found accuracy* of the forward pass in stage 2. These images were the unaltered input image, since the Mask R-CNN was not able to detect a bird label, therefore no cropping/segmentation was applied. In the processing step, a segmentation mask is applied before cropping. Higher values indicate better results.

number is 1664 images and 3709 images without a *bird* found for PGD. The are 49.6% (FGSM), 28.7% (BIM) and 64.0% of the images where no bird is found at the largest ϵ.

Stage 2 forward pass To present the results of the forward pass through stage two, we will calculate the accuracies according to the different types of certainty, we already presented for the adversarial attacks on the stage one model. These accuracies are the accuracies for: all images including *bird found* and *no bird found* images, which we will declare again as *all images accuracy*. Additionally, we will provide the results for both, images where a bird was still detected at stage one (*bird found accuracy*) and only the images, where no bird was detected after the adversarial attacks at stage one (*no bird found accuracy*).
We will start with the *all images accuracy* (fig. 7.25). For a later discussion this accuracy will be the most relevant one, since all the other models always get tested on all images for each adversarial attack. Due to the circumstance that we did not fit *SparseFool* (6.2.4) to be able to attack Mask R-CNN, we will also not provide results for stage two for *SpaseFool* here.
The initial *all images accuracy* is at 61% for models using bounding box crops, while models using the

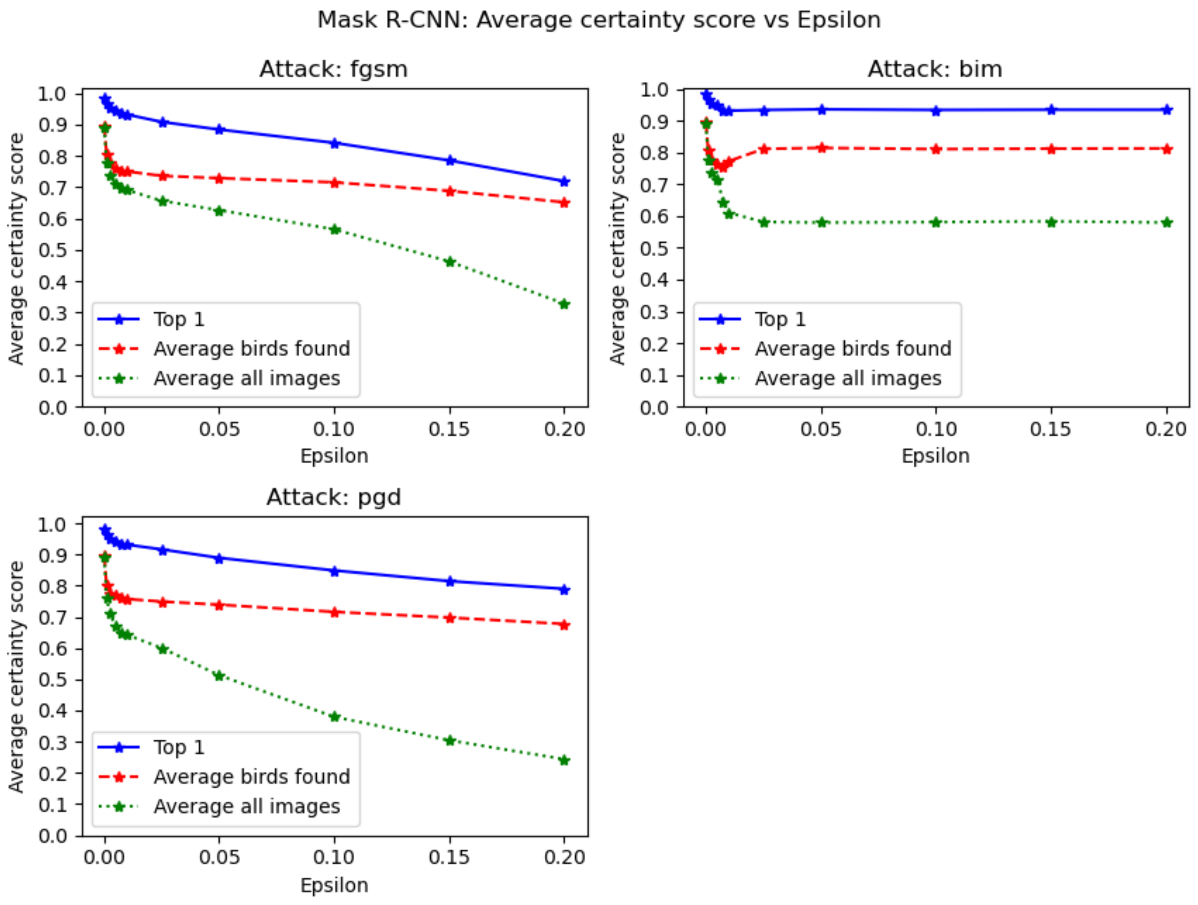

Figure 7.23: The average certainty score of Mask R-CNN when being attacked by adversarial attacks. A higher certainty indicates better results. *Top 1*: the certainty of the bird label with the highest score above a threshold of 0.3, *Average birds found*: the average score of all bird labels above a threshold of 0.3, *Average all images*: the average score of all bird labels above a threshold of 0.3. If no bird is found, the score for this image is weighted as 0. Higher values indicate better results.

borders of the segmentation as crops have an initial accuracy of 55%.
Regarding FGSM, the accuracy drops for both model types to 0% for the largest $\epsilon = 0.2$. *Cropbb* models drop at an overall slower rate for $\epsilon <= 0.1$ and for $\epsilon > 0.1$ they drop at a faster rate compared to the *segbb* models. This can be seen in fig. 7.26, top left. For this attack the accuracy drops to 0% for both all models not using segmented images.
The second attack is BIM, which can be seen in the figures 7.25 and 7.26, top right. Here the accuracy drops to 40% for $\epsilon <= 0.025$ and remains constant afterwards for *cropbb* models. The same behavior can be seen for *segbb* models, whose accuracy drops from the initially lower accuracy of 55% to an accuracy of 31%. The normalized graph for *BIM* shows that the relative drop in accuracy is lower for *cropbb* models and higher for *seggbb* models.
For PGD as the last attack in figures 7.25 and 7.26, bottom left, one can see that the accuracy for all models initially drops fast up to an $\epsilon = 0.01$ and then slows down its decrease rate. The *cropbb* models end up at an accuracy of 23% for *PGD* and the *segbb* models achieve an *all image accuracy* of 10%. The normalized graphs (fig. 7.26) show again that the rate in which the *all image accuracy* decreases is higher for the

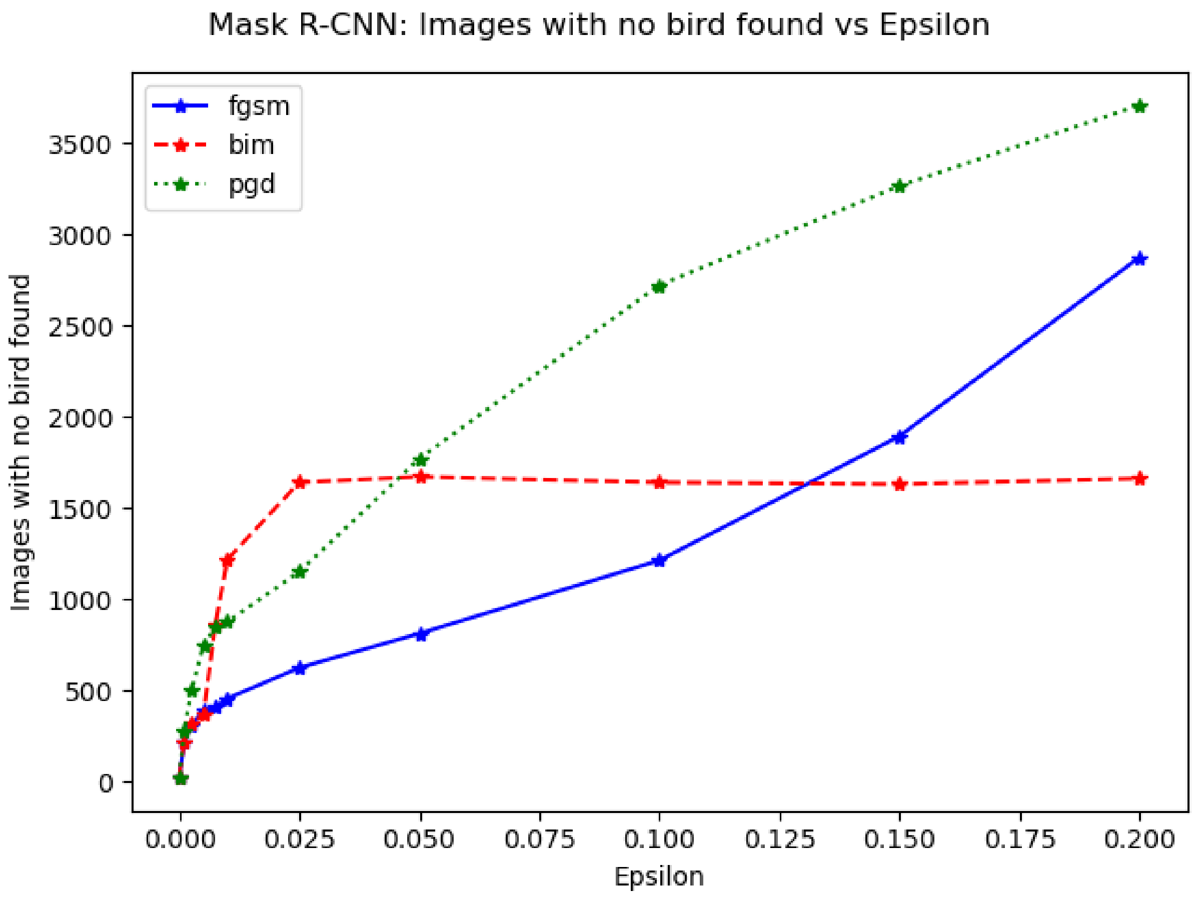

Figure 7.24: The number of images, where no bird with a score above a threshold of 0.3 was found. Higher values indicate worse results.

models using the borders of the predicted segmentation masks for cropping.
Regarding the *bird found accuracy* for these models, although there are very small differences in the accuracies itself compared to the *all images accuracy*, the plots are very similar, and we will therefore refrain from repeating the observations and showing the plot again.

The last kind of accuracy for the models not using segmentation masks for their images is the *no bird found accuracy*, which can be seen in fig. 7.27. For all attacks, the graphs for the *no bird accuracy* is in some parts very different from the *all images accuracy*. Initially the *no bird accuracy* is very low at 24%, but it increases very fast for small ϵ up to an area of 45% to 50% for *cropbb* models and the respective attacks and a small area around 35% for *segbb* models. For FGSM and PGD, all *no bird accuracies* drop again for $\epsilon > 0.025$ to end up lower than the initial *no birds accuracies* were, while the *no bird accuracy* remains almost constant for $\epsilon > 0.025$. The lowest *no birds accuracy* for all models for the adversarial attack *FGSM* for all models is close to 0% and for all models attacked by *BIM* the lowest *no birds accuracy* is also the initial accuracy at 25%. The lowest *no bird accuracy* for *PGD* for *cropbb* models are the initial accuracies at 25% and for *segbb* models, the lowest accuracy is at the largest ϵ value with an *no bird accuracy* of 15%.

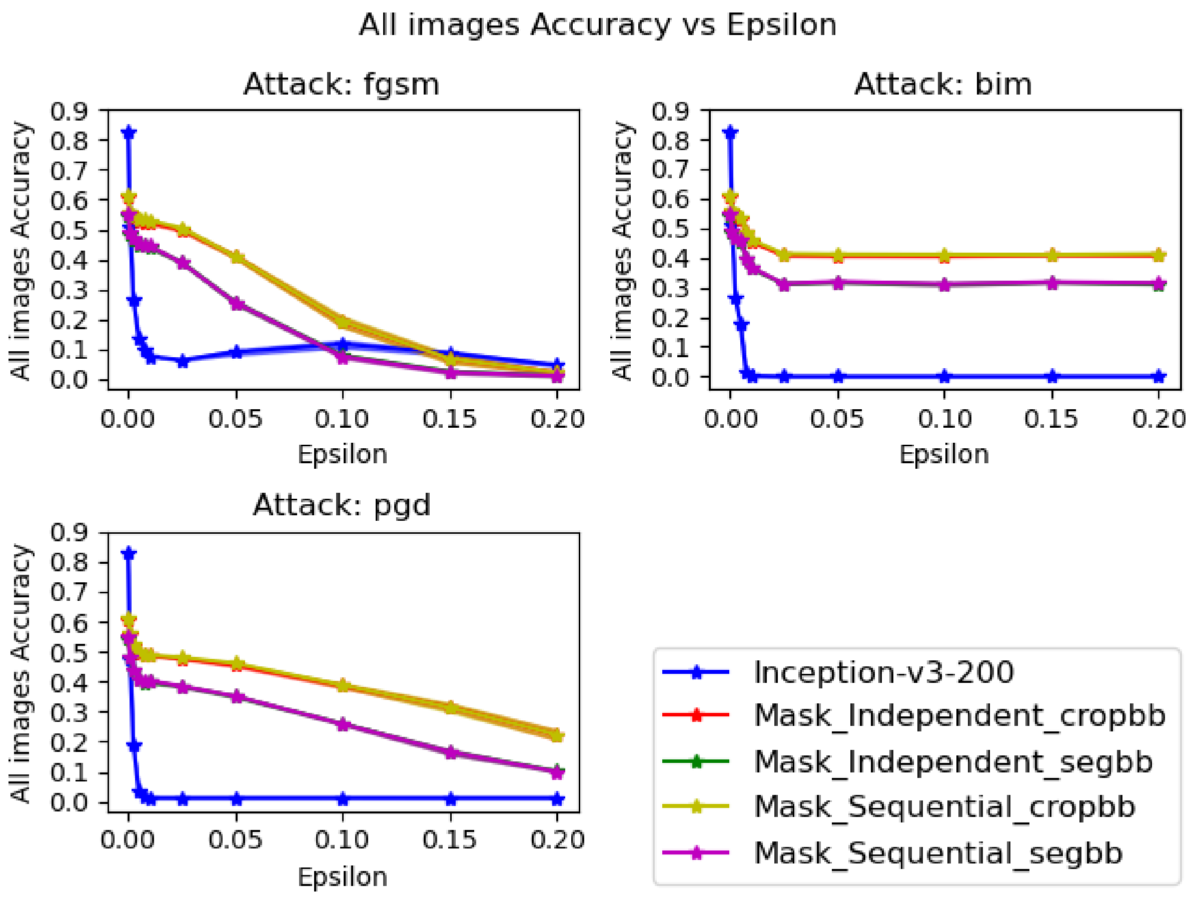

Figure 7.25: The *all image accuracies* of the forward pass in stage 2 and the results of the adversarial attacks on the baseline. Higher accuracies indicate better results.

Models with an applied predicted segmentation mask before cropping (bird on black background) We will now start with the models which are applying the predicted segmentation mask to the image before cropping. We will start again with the *all images accuracy*, followed by the *no bird found accuracy*. The graph for the *bird found accuracy* is again very similar to the graph for the *all images accuracy* and we will again not show it.

The results of the models using segmentations on the initial, non-perturbed images are similar to the results of the models not having segmentations applied to the input images. But for the models using segmentations (we will refer to them as *use_seg models*), have a higher decrease in accuracy for small ϵ. Except for this difference, the graphs of the *use_seg* models are quite similar to the graphs of the models not using segmented images (compare fig. 7.28 and 7.25).

For *FGSM*, there is an initial sharp decrease in accuracy, which slows down for $\epsilon >= 0.01$. The final adversarial *all image accuracy* is close to 0% for all models. For *BIM* the *all images accuracy* drops very fast to, but for $\epsilon >= 0.025$ is remains constant again at a level of 10% for *segbb* models and at a level of 15% for *cropbb* models. The *all image accuracy* for *PGD* sharply drops for small $\epsilon <= 0.01$ and drops much slower and in a linear way for $\epsilon >= 0.025$ up to an *all images accuracy* of 10% for *segbb* models and 15% for cropbb models. These results can be seen in fig. 7.28.

The relative decreases in *all images accuracy*, which can be seen in fig. 7.29, is quite similar for all models.

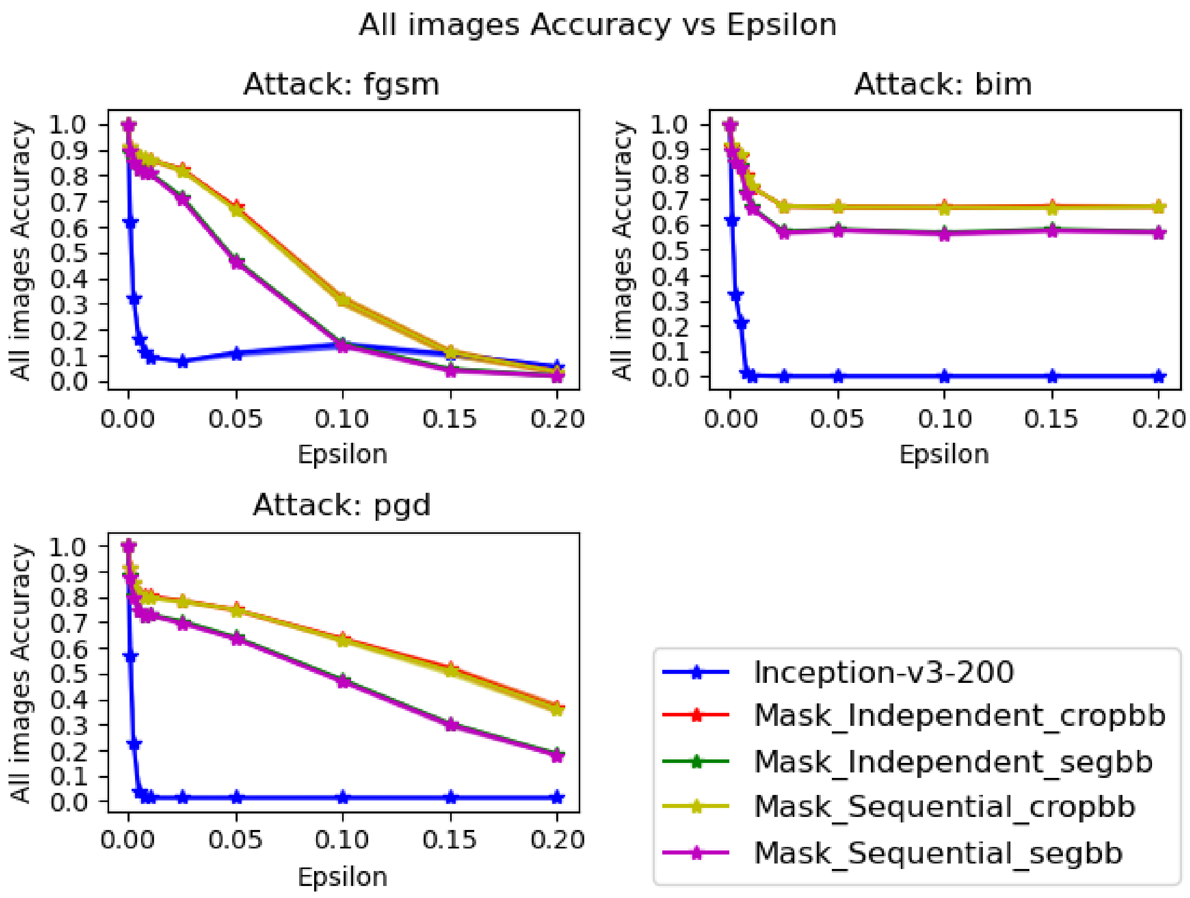

Figure 7.26: The normalized *all image accuracies* of the forward pass in stage 2 and the results of the adversarial attacks on the baseline. Higher accuracies indicate better results. This graph depicts the relative loss of accuracy for the models under attack.

The sole difference is that the *cropbb* models have slightly smaller relative decreases of the accuracy in the cases of *BIM* and *PGD*. These smaller decreases are visible for $\epsilon >= 0.01$.
Regarding the *no bird found* accuracies of the *use_seg* models, one can see again, that the graphs are quite similar to the ones from the models which are not getting segmented images, but the *no bird found* accuracies of *use_seg* models remain on a much lower level of 20% for *segbb* models and 30% for *cropbb* models, which is roughly 20% less *no birds found* accuracy compared to the models not getting segmented images. There is again a sharp increase in adversarial accuracy for all attacks up to $\epsilon = 0.01$ and then a decrease of the *no bird found* accuracy for *FGSM* close to a final 0% while the decrease for *PGD* end at 5% for *segbb* models and 10% for *cropbb* models. The only outlier is the adversarial attack *BIM* again, where the accuracy for the models remains almost constant at 16% for *sebb* models and 25% for *cropbb* models respectively.

Summary

In the previous sections we described the results of the three adversarial attacks *FGSM*, *BIM* and *PGD* on Mask R-CNN as the first stage of the *Mask Bottleneck model*, and the results of the following forward

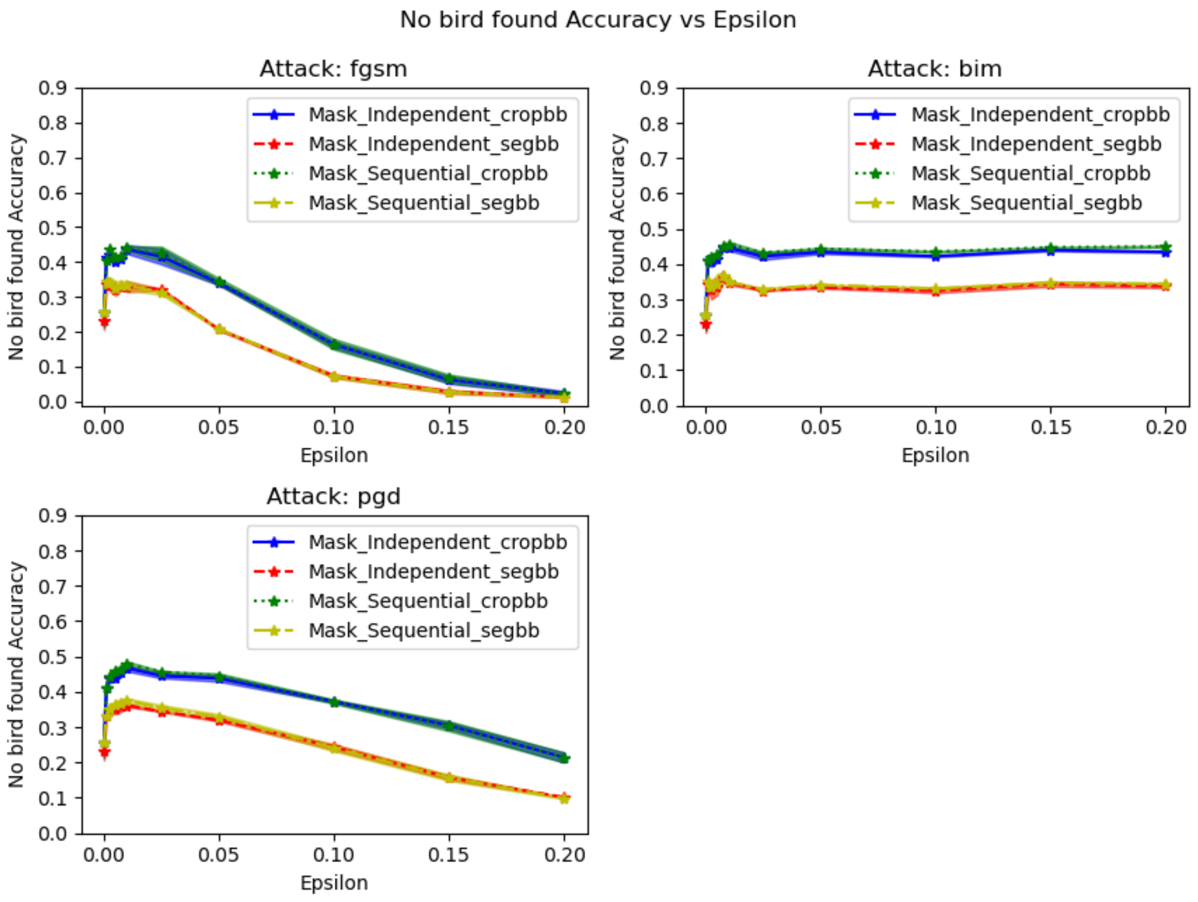

Figure 7.27: The *no bird found accuracy* of the forward pass in stage 2. These images were the unaltered input image, since the Mask R-CNN was not able to detect a bird label, therefore no cropping/segmentation was applied. Higher values indicate better results.

pass through the different second stage networks. The general result is that Mask R-CNN seems to be quite robust against adversarial attacks, since it can still detect the bird in 35% of the images while being attacked with the *most harmful* attack for the Mask R-CNN and the largest perturbation of the image possible. Regarding the second stage models, each of the model performs decent for smaller perturbations regarding the different prediction accuracies, both for *all images* as well as only the images, where a bird was found. For stronger perturbations, the accuracy of all models is close to 0% in the worst cases for these attack setups but not for all cases, which is a good start for finding robust models. Additionally important is the fact that *cropbb* models, which are models that are getting images as inputs, cropped by the predicted bounding boxes of the Mask R-CNN generally achieve better results than *segbb* models and models not using segmented images are more robust than models using segmented images.

7.6.4 Stage 2 adversarial attacks after forward pass through stage 1

In this section we will show the results of adversarial attacks solely on stage two after a forward pass through the Mask R-CNN and for the ***XtoY*** setup, we will also add the Inception-v3 baseline to the graph.

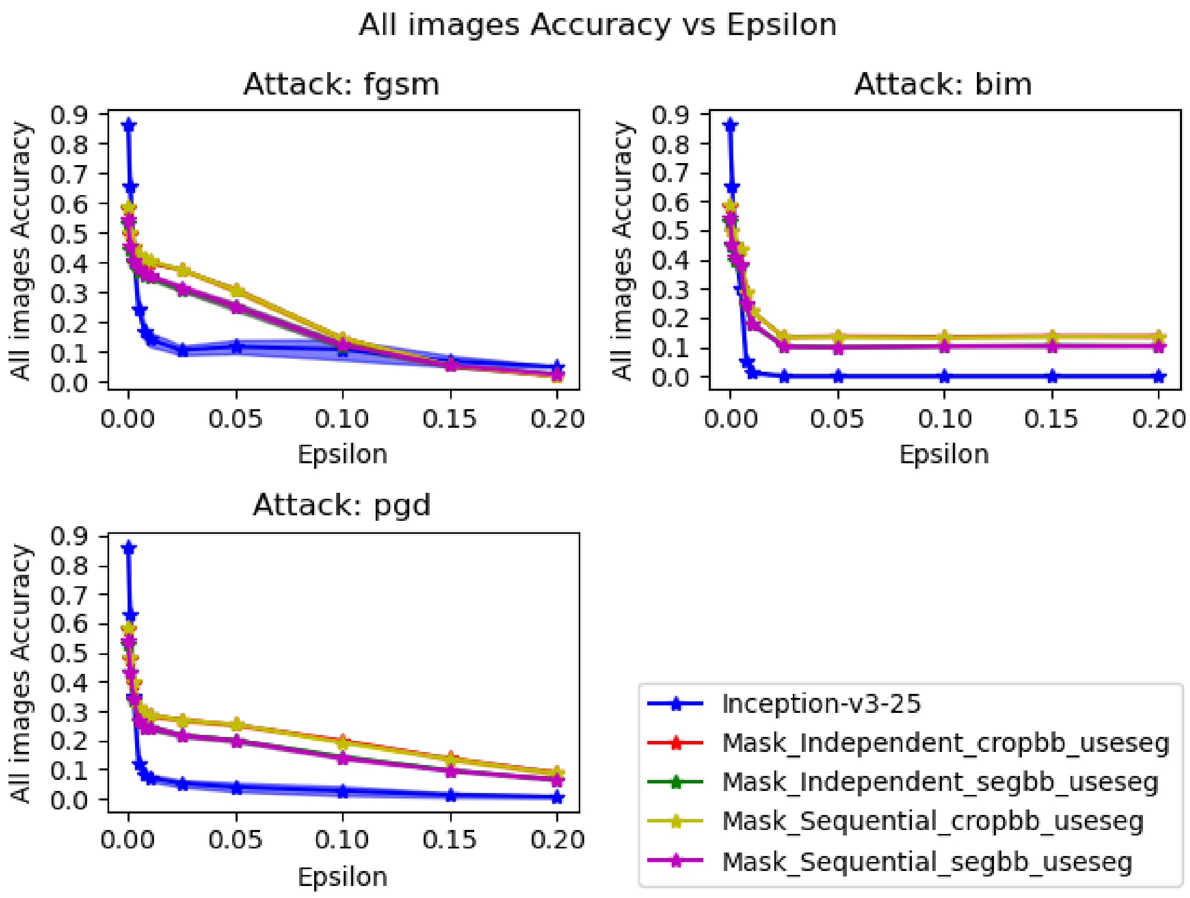

Figure 7.28: The *all image accuracies* of the forward pass in stage 2 and the results of the adversarial attacks on the baseline. In the processing step, a segmentation mask is applied before cropping. Higher accuracies indicate better results.

We remind here that the input images for Inception-v3 are unaltered except for normalization and resizing, while the input images for stage 2 of the *Mask Bottleneck* pipeline are first passed through the Mask R-CNN, then for some setups, the predicted segmentation mask was applied and then the images were cropped. The test input as well as the training datasets can therefore vary quite drastically in some cases although the underlying dataset is CUB for all models.

XtoY

We will now describe the first of the adversarial attacks on stage 2. Figure 7.30 shows the results of the attacks on the models, which are not applying segmentation masks before cropping the images. The behavior of all stage two models, which are not using segmented images is similar for FGSM and BIM. For each of the two attacks, the accuracy almost immediately drops to 0% and remains there for increasingly large epsilon. There is only a small slowdown in the decrease in accuracy for ϵ in range $0.001 <= \epsilon <= 0.005$. Regarding PGD, the accuracy drops close to zero immediately, without even slowing down for a few ϵ.
For SparseFool the accuracy initially drops to 30% for each of the models, then the accuracy remains

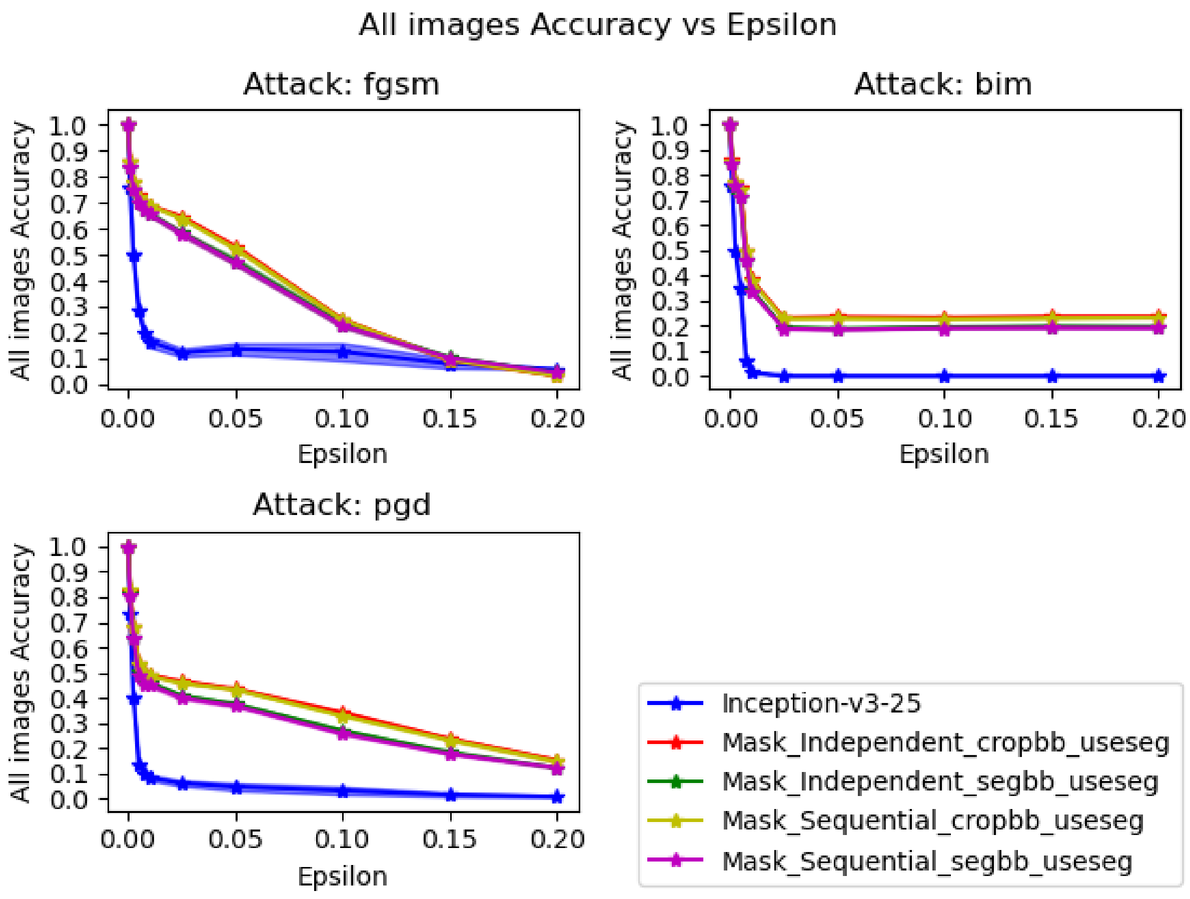

Figure 7.29: The normalized *all image accuracies* of the forward pass in stage 2 and the results of the adversarial attacks on the baseline. In the processing step, a segmentation mask is applied before cropping. Higher accuracies indicate better results. This graph depicts the relative loss of accuracy for the models under attack.

constant for increasingly large ϵ. For models, which are applying segmentation masks before cropping the images, the results can be seen in figure 7.31. For BIM, PGD and SparseFool, the results are similar with the sole difference of a slightly lower accuracy for unperturbed images and the constant accuracy for the models under attack of SparseFool is slightly lower.
There is a difference for FGSM though. The models, which are applying segmentation masks before cropping and cropping around the borders of the segmentation masks achieve an almost constant accuracy at 17% across all ϵ values. This accuracy for these two models is higher than the accuracy of all other models, including the Inception-v3 baseline.

Summary The results of the adversarial attacks of the setup *XtoY* on the second stage of the *Mask Bottleneck pipeline* show that the model is in most cases not robust to such attacks but very susceptible to the perturbations to the input images by the adversarial attacks of the L_∞-norm. The results of *SparseFool* on the other hand show a similar behavior to the original concept bottleneck models and to the other end-to-end models.

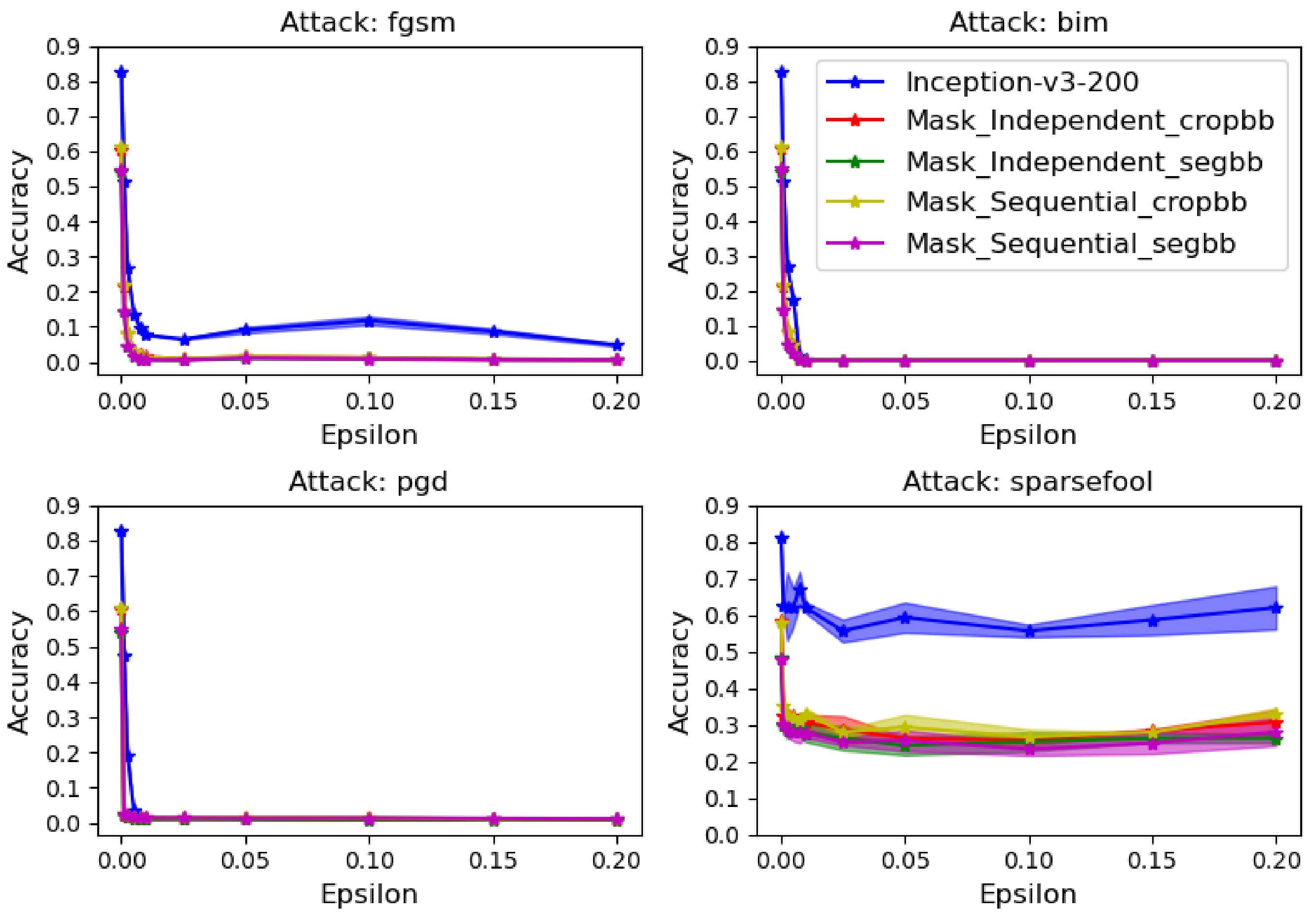

Figure 7.30: MB Stage 2 adv: Results of the adversarial attacks on stage 2 of the Mask Bottleneck model of setup **XtoY** for all ϵ values and the results of the adversarial attacks on the baseline. Task: Attack image **X** to change class predictions **Y**.

The results of the *segbb* models applying segmentation masks before cropping, show an interesting behavior. Its seems that there are some models, which are not as susceptible to adversarial attacks, then others, but that they have a good inherent robustness to such attacks.

XtoC

In this section we will describe the results of the adversarial attacks for the setup ***XtoC***. These results can be seen in fig. 7.32, firstly for the two models not using segmentation and in fig. 7.35 secondly for the two models, which are trained on segmented images.
As can be seen in fig. 7.33, the Concepts Models do first have a high accuracy of 97% while predicting concepts. Remarkable is here that the behavior of the two variants of cropping models is completely similar, even regarding the initial accuracy and the accuracy when being under attack with a difference in accuracy of less than 1%. This behavior can be seen in fig. 7.33. Due to this very similar behavior, we will describe both models at once using the model using bounding box crops as an example. We will refer to fig. 7.32

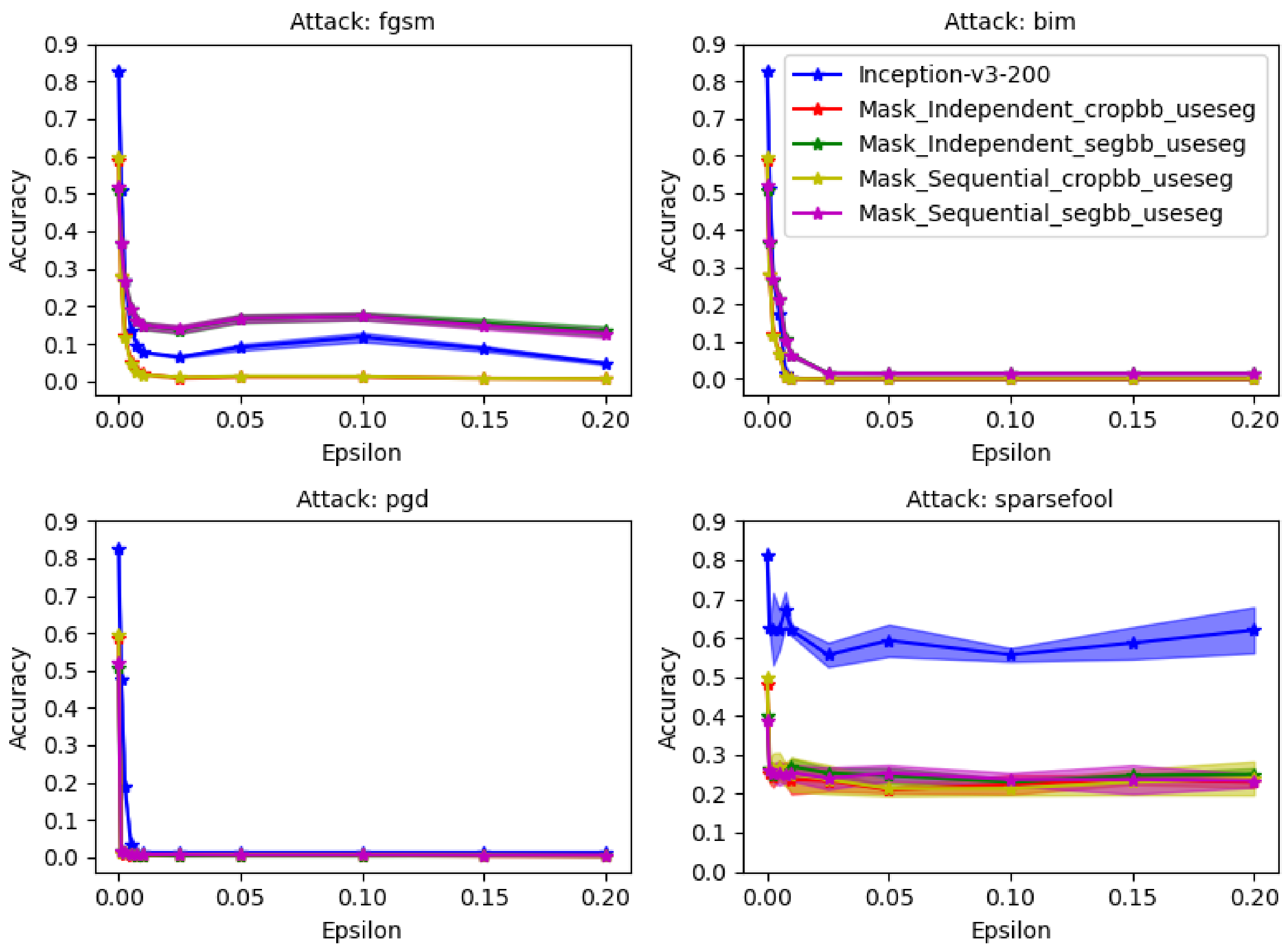

Figure 7.31: MB Stage 2 adv: Results of the adversarial attacks on stage 2 of the Mask Bottleneck model of setup *XtoY* for all ϵ value and the results of the adversarial attacks on the baselines. Segmentation masks are applied before cropping the image. Task: Attack image **X** to change class predictions **Y**.

for the description.

This accuracy decreases with increasing epsilon to 80% up to $\epsilon = 0.01$ and remains constant afterwards for FGSM. FGSM is also the only attack, which achieved a visibly higher standard deviation at the largest $\epsilon = 0.2$, although the standard deviation is too small to be considered important, it is a notable difference from the other attacks.

The Concept accuracy for the models under attack of PGD drop to 73% initially and remain constant afterwards, with a very strong drop in accuracy for the first ϵ and an otherwise similar behavior of the attack not being more successful with $\epsilon > 0.01$.

For BIM, the behavior is slightly different compared to FGSM and PGD. The accuracy is dropping lower to 72% and one can see the change in the number of iterations from the heuristic. With $\epsilon = 0.0075$, the accuracy drops slightly differently than for FGSM and PGD, having a shortly stronger decrease and the accuracy drops longer up to $\epsilon = 0.025$, which is one more ϵ value than for FGSM and PGD. Afterwards the concept accuracy remains constant again. Remarkable is here that the change of the heuristic to a second iteration is clearly visible for BIM, leading to a second sharper drop in accuracy.

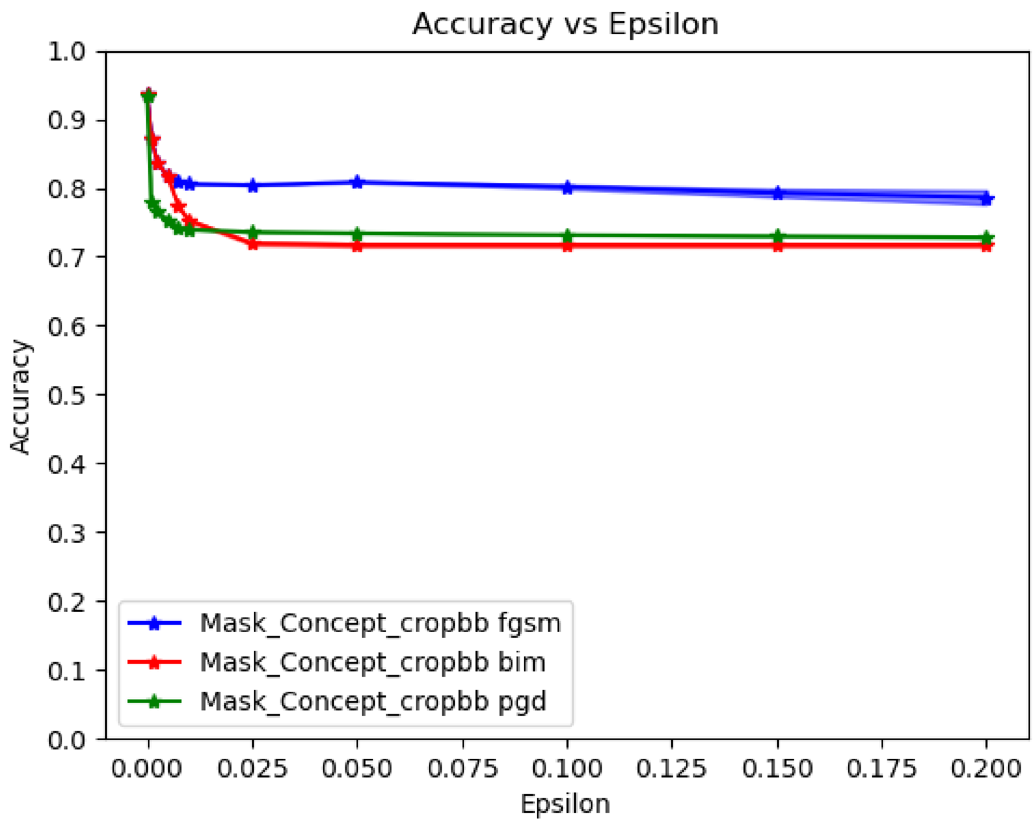

Figure 7.32: MB Stage 2 adv: Results of the adversarial attacks on stage 2 of the Mask Bottleneck model of setup *XtoC* for all ϵ values. Task: Attack image **X** to change concept predictions **C**.

Models with an applied predicted segmentation mask before cropping (ideally: bird on black background)
We will now show the results of the two models trained on cropped images after applying the segmentation masks predicted by the Mask R-CNN. As fig. 7.34 shows, the behavior of both models is quite similar again, but the adversarial accuracy of the model cropping with bounding boxes drops 5% lower to 80% compared to the model using the segmentation borders to crop, which drops to an adversarial accuracy of 85% for FGSM. This behavior is consistent for all attacks. The *segbb* model always achieve an accuracy which is about 3% to 5% higher than the *cropbb* model.
To look more detailed on the behavior of the models under attack, which will again use only one of the models as an example, because the behavior of the two models is similar again except for the higher drop in accuracy for the *cropbb* models.
In fig. 7.35, we again have a graph containing all attacks for one model at once. The sole difference of this graph to fig. 7.32 is, that FGSM has no notable increase of the standard deviation. This is why we will refrain from repeating the observations of the previous paragraph here. We will only point out here, that the behavior of the two cropbb models, one using segmented images, the other one using images with natural backgrounds is completely similar except for small differences in accuracy. The only model

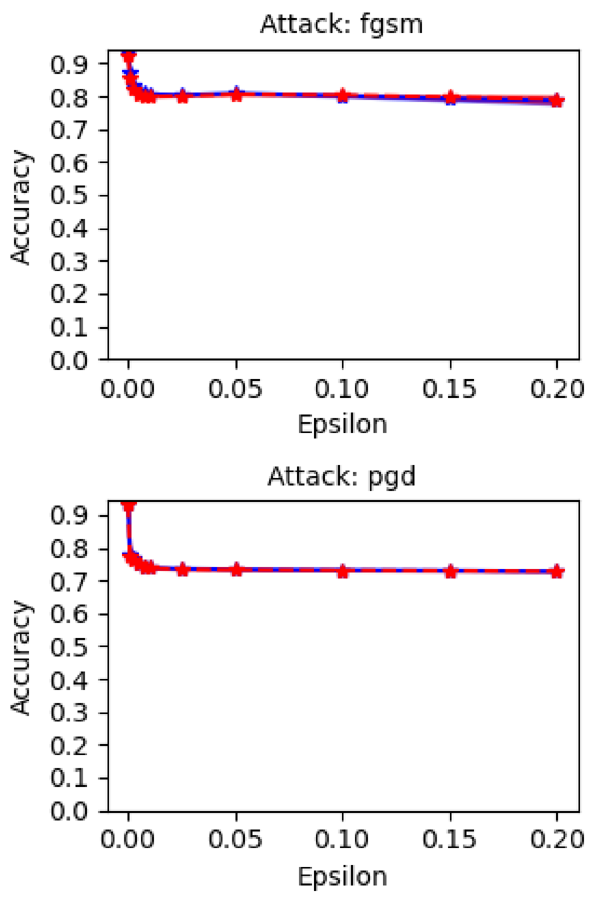

Figure 7.33: MB Stage 2 adv: Results of the adversarial attacks on stage 2 of the Mask Bottleneck model of setup *XtoC* for all ϵ values. All results in one graph and no segmentation masks are applied before cropping. Task: Attack image **X** to change concept predictions **C**.

standing out is *Mask_Concept_seggbb_useseg*, a model using segmentation borders to crop the images and an applied segmentation mask, with a slightly higher accuracy of 3% to 5% when being under attack.

Summary The concept models seem to be certain enough on their prediction that the proportion of changed concepts is quite low when it comes to the three adversarial attacks: FGSM, BIM and PGD. It does seem to make a difference though whether the adversarial attack is iterative (BIM, PGD) or not (FGSM), since the accuracy for the iterative approaches is slightly lower than for the non-iterative approach. Since the difference in accuracies for all models is quite low, but only the Concept model using segmentation masks and cropping around the borders of the segmentation mask (*segbb_useseg* is an outlier with a slightly higher accuracy compared to the other concept models, we can suggest using this model as the most robust one versus adversarial attacks.

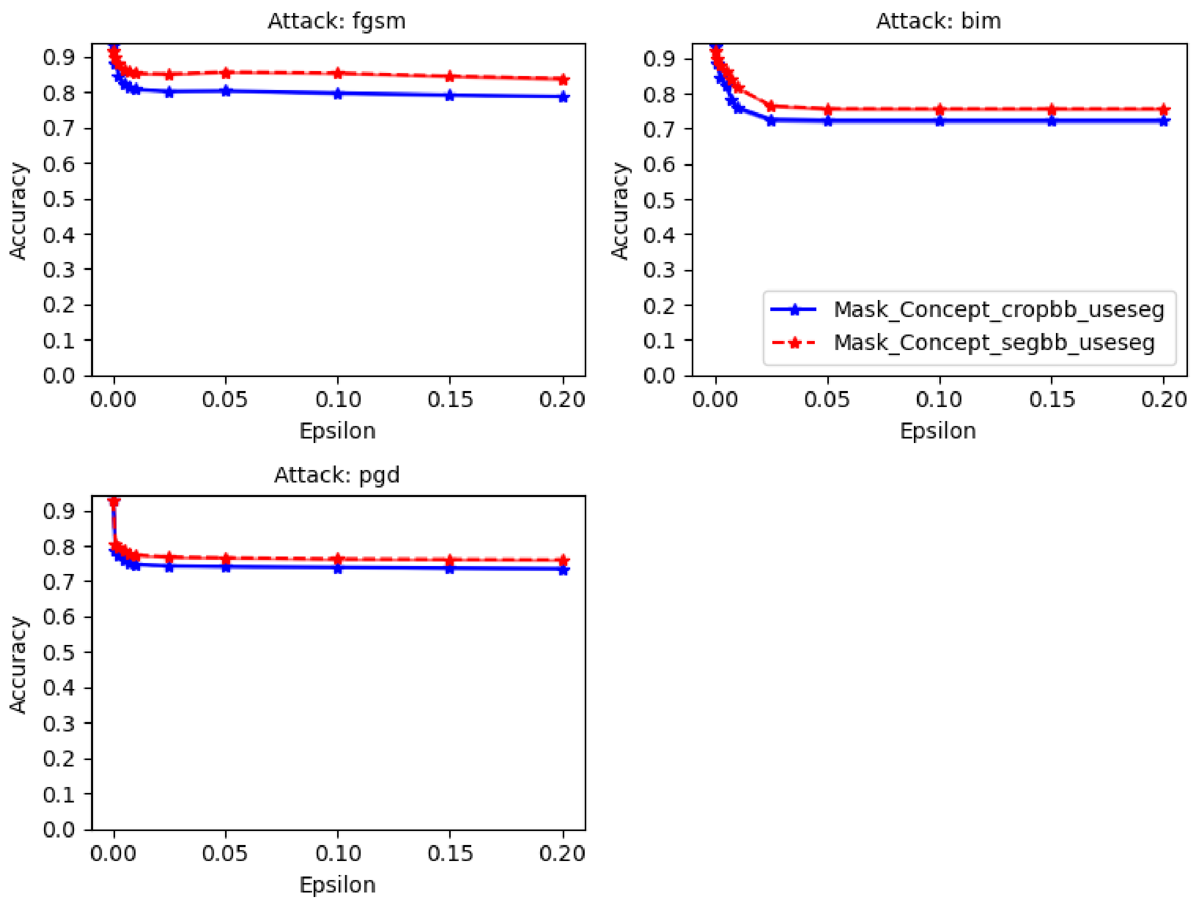

Figure 7.34: MB Stage 2 adv: Results of the adversarial attacks on stage 2 of the Mask Bottleneck model of setup *XtoC* for all ϵ values. All results in one graph and the segmentation masks are applied before cropping. Task: Attack image **X** to change concept predictions **C**.

CtoY

The next and final experimental setup for this part of the model is ***CtoY***. The results are depicted in fig. 7.36. First, this figure shows a clear trend for all models and attacks: the models with images cropped around the bounding boxes perform better than the models cropping around the edges of the segmentation masks and all models behave similar for each individual attack. This statement cannot only be made for the models not using segmented images but also for the models applying the segmentation mask before cropping. After this general observation, we will now describe the results for each individual attack.
Starting with FGSM, all models start with an initial accuracy at roughly 60% for *cropbb* models and 53% for *segbb* models and the accuracy for the models under attack fall linearly to an adversarial accuracy in range of 20% to 30% for *segbb* models and 35% to 45% for *cropbb* models with a slightly increasing standard deviation for each model with increasing ϵ. Fig. 7.37, shows that the *cropbb* models additionally achieve a smaller drop in accuracy for increasingly large ϵ values compared to the *segbb* models.
For BIM all models start with their initial accuracy and fall a total of 3% to 5% up to $\epsilon = 0.025$. Afterwards there is no further decrease in accuracy for all models and the standard deviation also remains constant.
A similar behavior can be seen for PGD regarding the constant standard deviation. The accuracy for all models only drops by 1% to 2% up to $\epsilon = 0.01$ and remains constant with larger ϵ. For SparseFool, all

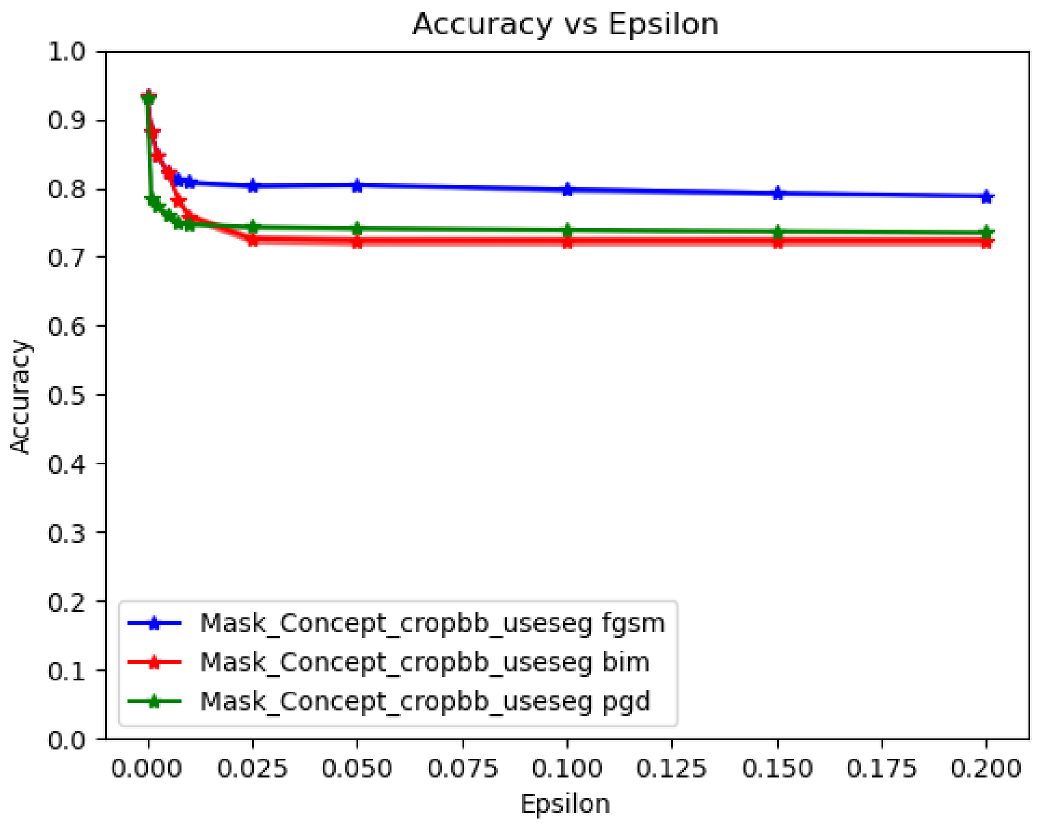

Figure 7.35: MB Stage 2 adv: Results of the adversarial attacks on stage 2 of the Mask Bottleneck model of setup *CtoY* for all ϵ values in one graph and for one model. Segmentation masks are applied before cropping. Task: Attack concepts **C** to change class predictions **Y**.

models behave similar again. We have to note here that since we use only 100 images as examples and not the full dataset, the initial accuracies vary depending on the randomly chosen images. Other than that, all models achieve an accuracy of 0% even for the smallest $\epsilon > 0$ and no model achieves an accuracy higher than 0% while being attacked by SparseFool.

Summary For this attack setup all models seem to be quite robust when it comes to FGSM; BIM and PGD, while they are not robust to large the large changes of SparseFool.
The choice for the most robust model might be one of the *cropbb* models, but whether it also uses segmentation or not is fully open, since there is no significant difference in their accuracy when being attacked by adversarial attacks in this setup and as it seems, the accuracy as well as the adversarial accuracy of the ending part of the model is mostly dependent on the certainty coming from the predicted concepts and it also comes from whether the adversarial attacks manipulate the concepts all a little or if they manipulate a few concepts in a very strong way so that some of them change their binary value.

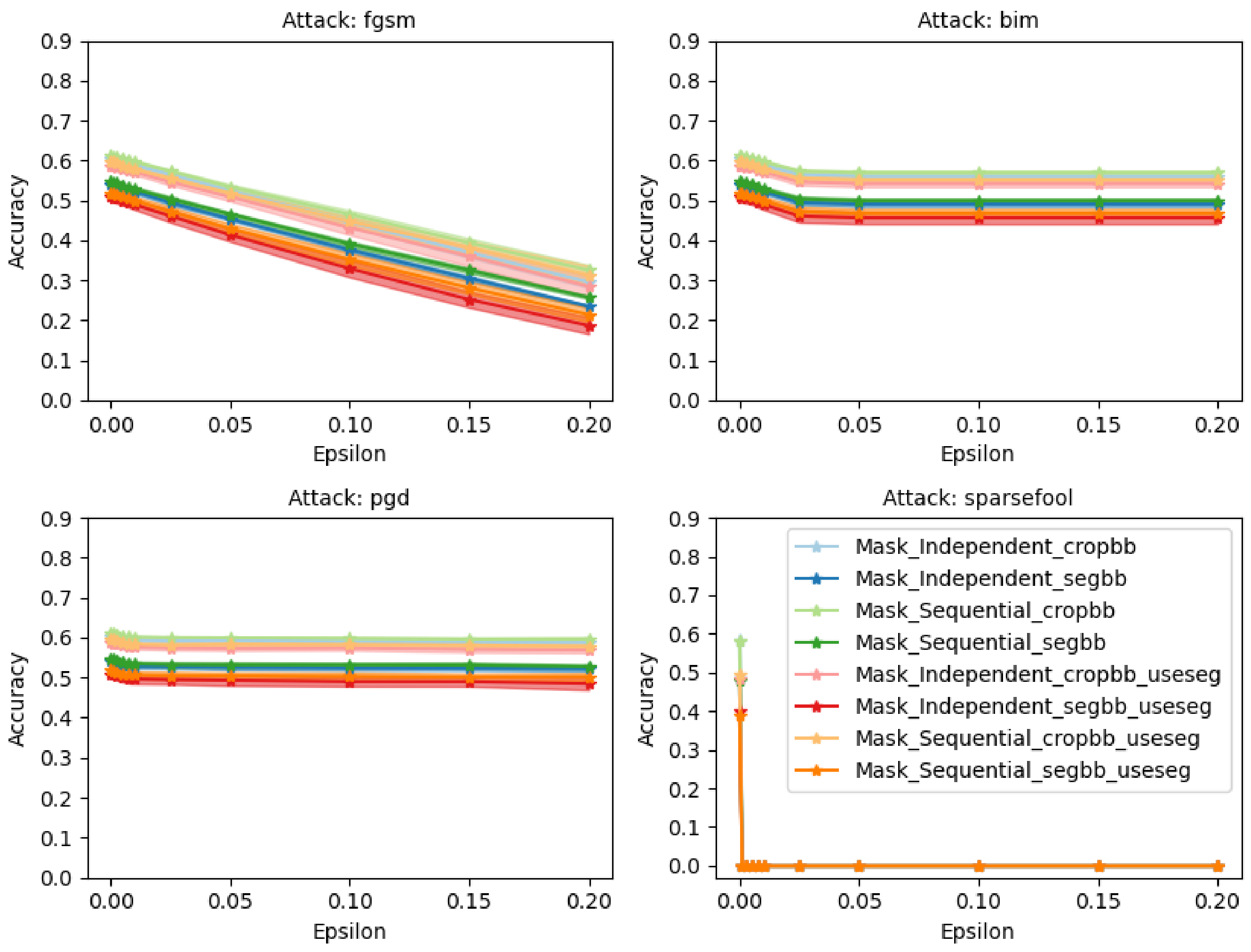

Figure 7.36: Results of the adversarial attacks on stage 2 of the Mask Bottleneck model of setup *CtoY* for all ϵ values. Task: Attack concepts **C** to change class predictions **Y**.

7.6.5 Summary

We provided the experimental results and setup dependent discussions for each of our setups regarding the *Mask Bottleneck* models. These results include the attacks on the first stage model only with a forward pass through stage two and adversarial attacks on the second stage models and input after a forward pass through stage one.
When describing the experimental setups previously, we already mentioned that it is a lot more likely to attack the first stage of the pipeline than to attack the second stage. We also mentioned that due to the interrupted gradient flow due to the processing step between the first and second stage of the pipeline, it is not possible to attack the full *Mask Bottleneck pipeline* at once.
Due to the unlikeliness of direct adversarial attacks on the second stage of the pipeline, we will keep the analysis short here and just summarize our findings. The second stage of the pipeline is very susceptible to adversarial attacks, when being attacked directly without showing much inherent robustness to these attacks in most cases. Some cases on the other hand show a strong inherent robustness to adversarial attacks, even when being compared to the baseline, such as the *segbb* model, applying segmentation masks

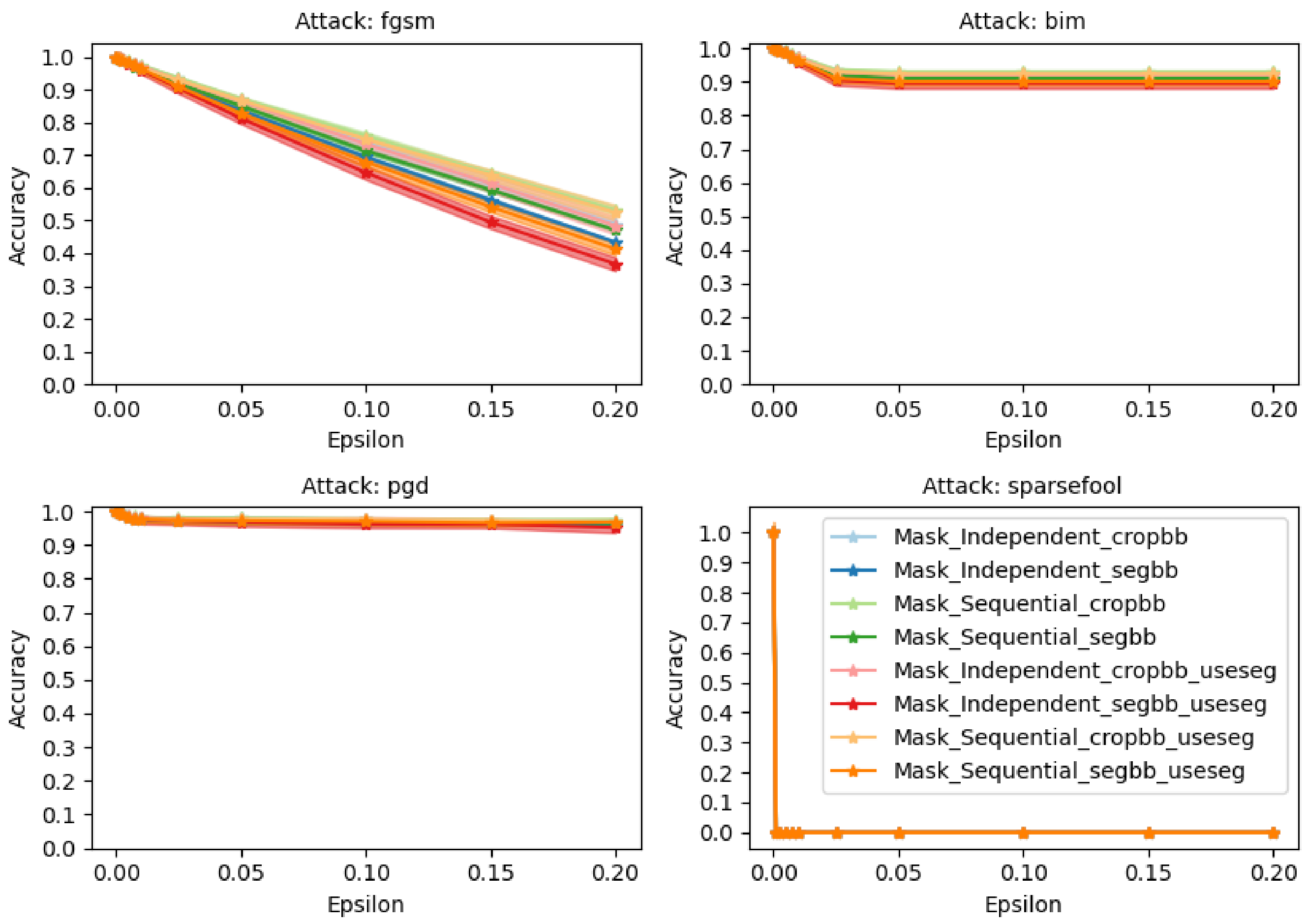

Figure 7.37: Normalized results of the adversarial attacks on stage 2 of the Mask Bottleneck model of setup *CtoY* for all ϵ values. Task: Attack concepts **C** to change class predictions **Y**.

before cropping in the *XtoY* setup.
Regarding the different setups for the first stage of the pipeline, we will start with the *TravelingBirds* out-of-distribution attack. Here it is obvious that the Mask R-CNN is very robust to such outliers by not being able to find a bird in 6.1% of the cases in the worst case. This means, that the ratio of *birds found in all images* is at 93.9%. This indicates a high abstraction level of the model as well as a good inherent representation of a *bird* without the necessity of fine-tuning the Mask R-CNN.
The pass through stage two of the pipeline then also shows, that after the forward pass and the pre-processing, that the second stage of the model is also quite robust to the remaining image parts of the out-of-distribution images. We call it remaining here since the cropping applied in the processing after stage one, is indented to remove most of the background.
The second kind of attack we applied to stage one followed by passing the processed results through stage two were the adversarial attacks. Here the results also show an inherent robustness to the adversarial attacks. The Mask R-CNN is able to find birds in at least 39% of the perturbed images and the following forward pass yields a classification accuracy of roughly 20% on all images, for $\epsilon < 0.1$.
With these results for the *Mask Bottleneck* model, we found a setup, which has not the highest initial classification accuracy on the unaltered dataset (which is roughly 60%), but the pipeline shows inherent robustness to both out-of-distribution data and adversarial attacks. This is achieved by two factors. First by

using a high-level object detection network, which just finds the abstract object, which is in our case a *bird*. Then a second stage model classifies the bird in the cropped into its species. The second factor is the processing step, where first, the background is reduced by a lot and secondly, the image is resized to fit as input for the second stage. In most cases the image is upscaled, when being resized.

8 Discussion

In the previous sections, we presented the datasets we used as well as the different models and experimental setups. Then we presented the experimental results on all models regarding the *TravelingBirds out-of-distribution* attack and the different *adversarial attacks* and analyzed the results for each model individually. In this section we discuss the results and the meaning of the results with a view on all models and compare them amongst each other, to find an answer to our initial question. This initial question was if there is an initial robustness to one or both of the attacks for models, which are either using human-like intermediate results such as the *Concept Bottleneck* models or for models, which are providing a way to understand and interpret intermediate results of the decision-making process. Subordinate questions were, how the models behave when being attacked, how the influence of different kinds of attacks is, depending on how much of the input they change and if the removal of information, which is not required from a human perspective, helps to achieve inherent robustness.
The models we used were an Inception-v3 model as baseline, Concept Bottleneck models, variations of Scouter, which is using a variant of slot attention, Vision Transformers, and our self-developed Mask Bottleneck model.

Slot Attention & Set Transformer Our combination of slot attention and set transformers was not successful. We assumed multiple reasons for this. At first, the slot attention model as it was proposed originally was never tested on real data, but only on artificial toy data. While slot attention provided strong results on these toy datasets, we were not able to train a version of the model, which has an additional bottleneck layer on the real-world CUB dataset. A very likely assumption for the unsuccessful training is one the one side the real data, which provides many more features than the toy dataset. This might require a much larger model to be able to handle the feature representations. A second reason is the imbalance of the present concepts compared to the absent concepts. While this imbalance does not provide any difficulties for other models, the slot attention approach strongly amplifies the imbalance towards the absent concepts and the model chose to predict every concept as absent as the best metric to optimize the initial as well as the adapted losses.

Vision Transformer Vision Transformers will also be excluded in further discussion since they achieved the worst results in most cases among all models. Vision Transformers are very new models and from previous research about adversarial attacks on Vision Transformers (Shao et al. 2021) that these models show some inherent robustness on this task for large datasets. Shao et al. (2021) also used PGD as one of their adversarial attacks with 40 steps instead of the two steps we used. Their experimental results are stronger than ours, but even for them, the accuracy of the models drop a lot, but they have a testing accuracy on the unperturbed dataset above 75% for each of their models, where we were far away from. Since the performance of Vision Transformers on the small CUB dataset was not as strong as one of the other models we used, we assume two possibilities for these results. First, Vision Transformers distinguish themselves from all other models we used by not using convolutions to process the images. This unique characteristic of Vision Transformers might lead to more inaccurate results on out-of-distribution data as well as on adversarial attacks. The second possibility was already named earlier, but we will repeat it here. The training results of especially the larger models were quite bad and we assume that the CUB dataset is too small for these large models to learn good features and representations from the images and that they tend to memorize the data. This memorization harms the performance firstly on the unseen test dataset and secondly on the adversarial and out-of-distribution images.

8.1 TravelingBirds out-of-distribution Attack

Among all models, the Inception-v3 baseline performed best in almost all cases on the out-of-distribution attack applied by using the *TravelingBirds* dataset. On an absolute basis regarding the classification accuracy, the Inception-v3 baseline performs best among all models, which might partially come from the high accuracy on CUB. This test accuracy on CUB was not matched by any of the other models. This is different for the relative accuracy or the relative loss in accuracy when it comes to the out-of-distribution attacks. Here the original *Concept Bottleneck* models (Koh et al. 2020) achieved results close to the results of the baseline and the Mask Bottleneck model achieves similar relative results for the models which are not using segmented images.
Mask Bottleneck model versions, which are using segmented images, achieve better results than the respective Inception-v3-black baseline and outperform the baseline by a lot. The most likely reason is here that the Mask R-CNN of the first stage of the pipeline and the processing step afterwards remove most of the confounding information by applying the segmentation mask and removing the background by cropping, which is not the case for the Inception-v3-black baseline due to the lack of such processing step. Regarding the TravelingBirds attack, all models achieve lower results on this task compared to their in-distribution datasets. It seems that all models use background information for their class predictions. Even if they never had any information in the background, because it was black, they seem to be confounded, when the background changes. The most reliable models on this task are our own developments, the Mask Bottleneck models, which are applying the segmentation masks before cropping the image around the predicted bounding boxes. The second-best models are the Inception-v3 baseline as well as the Mask Bottleneck models, which are not applying segmentation masks before cropping around the predicted bounding boxes. The disadvantage of our models is still the large difference in initial accuracy on the in-distribution dataset, which is 25% lower compared to the respective baseline model.

8.2 Adversarial Attacks

We will now move on to the adversarial attacks. All models experienced a large loss in accuracy, when they were attacked. But there are several models, which achieved decent results here, especially regarding smaller ϵ values with $\epsilon <= 0.1$. The first one to name is the Scouter variant with negative supports using one slot per class and $\lambda = 10$. This model achieved strong results with an accuracy of above 35% for $\epsilon < 0.1$ and 25% for $\epsilon = 0.1$. This accuracy was only surpassed by the Mask Bottleneck model without using segmentation masks but using the predicted bounding boxes for cropping, with an accuracy on all images above 40% for $\epsilon < 0.1$ on *FGSM*. For the other two attacks (*BIM, PGD*), our own model achieved the best results compared to all other models. In this case the models not using segmentation masks before cropping are more robust to adversarial attacks.

8.3 General Discussion

We investigated several models, all with different characteristics. The Inception-v3 model provided a very strong baseline, which also beat most of the models investigated. Our own development, the Mask Bottleneck pipeline, consisting of a high-level object detection model, namely the Mask R-CNN, a following processing step and a concept bottleneck model as the second stage seems be the model, which provides the most inherent robustness to both out-of-distribution inputs as well as inputs, which were perturbed by adversarial attacks.
We assume that most of the inherent robustness is achieved by two characteristics of the Mask Bottleneck pipeline. The object detection step searches for an abstract representation of a bird in general, without taking the information of specific bird species into consideration. The Mask R-CNN seems to have learned very good and very high-level feature representations for this task to be very unsusceptible to out-of-distribution data as well as adversarial attacks. Since it performed very well on both tasks, but especially well on the out-of-distribution data.
Secondly, the processing step, which includes cropping (and potential segmentation, depending on the task) removes many of the perturbations in each image. These perturbations include big parts of the background information, which enables the second stage of the pipeline to focus on the bird itself, without obfuscating background information. From the *TravelingBirds* out-of-distribution experiments we already know that the background information is obfuscating for the different models. But there is more obfuscating information in the image, which is at least partially removed during the processing step. Due to the cropping, the image sizes vary from each other in size. To put these processed images in the second stage of the pipeline, a resizing step is required, which resizes the images to 224x224 pixels to be of the right size to fit into the ResNet-34, we used for our concept models. Due to the cropping and although there is no fixed input size for the Mask R-CNN, most cropped images are smaller than 224x224 pixels and the images have to be increased in size. For this resizing, interpolation is used, which might also smooth out some of the perturbations made by the adversarial attacks.
We assume here that our Mask Bottleneck model will also provide even better results and a stronger robustness, when increasing the accuracy on the in-distribution datasets. From our experiments, we know that the concept predictions are very accurate, but that the predictions using the concepts to predict the bird classes are quite error prone in general and even more error prone when changing very few of the predicted concepts by a lot. This behavior most likely results from the sparsity of unique attributes for each bird class, but we will leave this investigation and possible solutions for future work.

8.4 Limitations

To evaluate the robustness of the models on different tasks, we always used the accuracy as our metric of choice. While this might be a good choice for end-to-end classification networks, this metric has its

limitations regarding the intermediate concept predictions and on the object detection and instance segmentation of Mask R-CNN.
The limitations on the concept predictions are that there are several concepts at once, all of which get predicted by a score between 0 and 1. To calculate the accuracy of these prediction, we have to binarize them and loose some information here. Some kind of loss evaluation might have been another good choice here. But it is then difficult to evaluate this loss again, since the information on which concepts got predicted as present or absent is better included in the accuracy than in the loss. On the other hand, the loss includes the information about the certainty of each prediction where values closer to the borders 0 or 1 indicate a higher certainty than values close to 0.5.
Regarding the evaluation of the Mask R-CNN, additional metrics might have given a more detailed information about the behavior of the model under attack. Even in this thesis, we left out several of the calculated metrics, since they would inflate the size of this work by a lot. These metrics for example include the losses of box regression (how well the bounding boxes were predicted) and segmentation mask (how well each segmentation mask was predicted).

9 Conclusion

In this thesis, we investigated several recently published models as well as our own developments on the two tasks of performing on out-of-distribution inputs and on performing on inputs which were perturbed by adversarial attacks.
Most models performed decently on the out-of-distribution attacks and the largest factor for this task is the initial accuracy a certain model achieves on the original dataset it was trained on. While this behavior is constant through the work on this thesis there are other important findings we made. When it comes to out-of-distribution attacks, the complexity of the background changes play an important role. More complex backgrounds or backgrounds, which differentiate more strongly from the in-distribution background hurt the decision making of a model more strongly, than backgrounds which are more uniform or more closely related to the training distributions. A second finding is the loss in prediction accuracy. This can be decreased largely by automatically reducing obfuscating background information or by removing any background information at all by a high-level object detection network during training and testing. This object detection network did in our case not even require a fine-tuning on a specialized task, as long as the is task is included in an abstract or high-level way in the original training data. In our case this abstract representation is *bird*, which is included in the original training data of Mask R-CNN.
Using the removal of obfuscating background information, the models performing the detailed classification can focus on the relevant object itself without dealing with potentially noisy background information and these models therefore provide an inherent robustness to such out-of-distribution data.

The adversarial attacks showed that all models are very susceptible to such attacks and that stronger perturbations of the input lead to worse decisions of the models. It is additionally important what proportion of the input is perturbed. Strong perturbations on small parts of the input seem to have less influence on the final decision than smaller perturbation to large parts or all parts of the input. The latter ones oftentimes even lead to the inability of the investigated models to achieve any correct predictions in the end-to-end case. On the other hand, strong perturbations on parts of the images do hurt models, which are using sparse data representations as input more, than these models are hurt by small perturbation on all input data. During the evaluation, we found, that removing obfuscating background information is not only important for out-of-distribution data, but also for adversarial attacks. During the adversarial attack experiments, we found that the Mask Bottleneck model, which is automatically removing background information is inherently more robust to adversarial attacks in many cases.
All in all, we found in this thesis that the susceptibility to both attacks can be reduced by introducing a high-level object detection network. The predictions of this object detection network can be used to remove background information, which then improves the performance of the models. We were able to achieve the best results amongst all models, by using the object detection model to remove obfuscating and from a human point of view irrelevant information from the images. After the removal of such information, a specialized model can then solve the respective specialized task in a better way. In our case this specialized

task was the classification of bird species.

10 Future Work

Future work emerging from our work can be found on several tasks, experiments and models.
We will start with the approach of combining Slot Attention and Set Transformer. This approach would require further testing of models of different sizes as a backbone for the Slot Attention module. Additionally, the capacity of the Slot Attention module has to be adapted potentially to be able to perform on tasks involving real data. A second part of future work on this combined model would be to find a way, which circumvents the high imbalance in the bird attributes, when it comes to two or more slots and all attributes are summarized in one foreground slot.
Regarding Vision Transformers, future work includes more training and testing of these large models on small datasets to first achieve a better accuracy and to find ways to circumvent potential memorization of training data. Following from this, further research could also include more testing of Vision Transformers on out-of-distribution data as well as on adversarial attacks.
Future research on Concept Bottleneck models, including the ones we used for our Mask Bottleneck model would be to find unique concept representations, which are less sparse than the current ones used. Additionally, the high gap between the accuracy of the concept predictions from images and the accuracy of class predictions from the concepts should be investigated and hopefully narrowed down.
Future research regarding our own full Mask Bottleneck pipeline would be to find models, which achieve a higher accuracy than our currently used models in the second stage of the pipeline. Additionally, testing our Mask Bottleneck model on different classification tasks on in-distribution data would be part of future research to assess the performance of our model on their generalization possibilities. Additional research would also include to use adversarial training on the models, to evaluate if it is possible to increase its inherent robustness to adversarial and out-of-distribution attacks any further.

Bibliography

Arnab, Anurag, Ondrej Miksik, and Philip H. S. Torr (2017). "On the Robustness of Semantic Segmentation Models to Adversarial Attacks". In: *CoRR* abs/1711.09856. arXiv: 1711.09856. URL: http://arxiv.org/abs/1711.09856.

Bär, Andreas et al. (2021). "The Vulnerability of Semantic Segmentation Networks to Adversarial Attacks in Autonomous Driving: Enhancing Extensive Environment Sensing". In: *CoRR* abs/2101.03924. arXiv: 2101.03924. URL: https://arxiv.org/abs/2101.03924.

Bastani, Osbert et al. (2016). "Measuring Neural Net Robustness with Constraints". In: *CoRR* abs/1605.07262. arXiv: 1605.07262. URL: http://arxiv.org/abs/1605.07262.

Bhambri, Siddhant et al. (2019). "A Study of Black Box Adversarial Attacks in Computer Vision". In: *CoRR* abs/1912.01667. arXiv: 1912.01667. URL: http://arxiv.org/abs/1912.01667.

Buzhinsky, Igor, Arseny Nerinovsky, and Stavros Tripakis (2020). "Metrics and methods for robustness evaluation of neural networks with generative models". In: *CoRR* abs/2003.01993. arXiv: 2003.01993. URL: https://arxiv.org/abs/2003.01993.

Cordts, Marius et al. (2016). "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *CoRR* abs/1604.01685. arXiv: 1604.01685. URL: http://arxiv.org/abs/1604.01685.

Dosovitskiy, Alexey et al. (2020). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *CoRR* abs/2010.11929. arXiv: 2010.11929. URL: https://arxiv.org/abs/2010.11929.

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy (2015). *Explaining and Harnessing Adversarial Examples*. arXiv: 1412.6572 [stat.ML].

Guo, Chuan et al. (2019). "Simple Black-box Adversarial Attacks". In: *CoRR* abs/1905.07121. arXiv: 1905.07121. URL: http://arxiv.org/abs/1905.07121.

He, Kaiming, Georgia Gkioxari, et al. (2017). "Mask R-CNN". In: *CoRR* abs/1703.06870. arXiv: 1703.06870. URL: http://arxiv.org/abs/1703.06870.

He, Kaiming, Xiangyu Zhang, et al. (2015). "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385. arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

Hirschberg, Julia and Christopher D Manning (2015). "Advances in natural language processing". In: *Science* 349.6245, pp. 261–266.

Howard, Andrew et al. (2019). "Searching for MobileNetV3". In: *CoRR* abs/1905.02244. arXiv: 1905.02244. URL: http://arxiv.org/abs/1905.02244.

Kim, Hoki (2020). "Torchattacks : A Pytorch Repository for Adversarial Attacks". In: *CoRR* abs/2010.01950. arXiv: 2010.01950. URL: https://arxiv.org/abs/2010.01950.

Kim, Taesup, Sungwoong Kim, and Yoshua Bengio (2020). "Visual Concept Reasoning Networks". In: *CoRR* abs/2008.11783. arXiv: 2008.11783. URL: https://arxiv.org/abs/2008.11783.

Koh, Pang Wei et al. (2020). "Concept Bottleneck Models". In: *CoRR* abs/2007.04612. arXiv: 2007.04612. URL: https://arxiv.org/abs/2007.04612.

Kurakin, Alexey, Ian J. Goodfellow, and Samy Bengio (2016). "Adversarial examples in the physical world". In: *CoRR* abs/1607.02533. arXiv: 1607.02533. URL: http://arxiv.org/abs/1607.02533.

Lee, Juho et al. (2018). “Set Transformer”. In: *CoRR* abs/1810.00825. arXiv: 1810.00825. URL: http://arxiv.org/abs/1810.00825.
Li, Liangzhi et al. (2020). “SCOUTER: Slot Attention-based Classifier for Explainable Image Recognition”. In: *CoRR* abs/2009.06138. arXiv: 2009.06138. URL: https://arxiv.org/abs/2009.06138.
Lin, Tsung-Yi et al. (2014). “Microsoft COCO: Common Objects in Context”. In: *CoRR* abs/1405.0312. arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312.
Locatello, Francesco et al. (2020). “Object-Centric Learning with Slot Attention”. In: *CoRR* abs/2006.15055. arXiv: 2006.15055. URL: https://arxiv.org/abs/2006.15055.
Madry, Aleksander et al. (2019). *Towards Deep Learning Models Resistant to Adversarial Attacks*. arXiv: 1706.06083 [stat.ML].
Modas, Apostolos, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard (2018). “SparseFool: a few pixels make a big difference”. In: *CoRR* abs/1811.02248. arXiv: 1811.02248. URL: http://arxiv.org/abs/1811.02248.
Moosavi-Dezfooli, Seyed-Mohsen, Alhussein Fawzi, and Pascal Frossard (2015). “DeepFool: a simple and accurate method to fool deep neural networks”. In: *CoRR* abs/1511.04599. arXiv: 1511.04599. URL: http://arxiv.org/abs/1511.04599.
Ren, Shaoqing et al. (2015). “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *CoRR* abs/1506.01497. arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.
Rohanian, Omid et al. (2020). “Verbal multiword expressions for identification of metaphor”. In: ACL.
Russakovsky, Olga et al. (2014). “ImageNet Large Scale Visual Recognition Challenge”. In: *CoRR* abs/1409.0575. arXiv: 1409.0575. URL: http://arxiv.org/abs/1409.0575.
Sagawa, Shiori et al. (2019). “Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization”. In: *CoRR* abs/1911.08731. arXiv: 1911.08731. URL: http://arxiv.org/abs/1911.08731.
Shao, Rulin et al. (2021). “On the Adversarial Robustness of Visual Transformers”. In: *CoRR* abs/2103.15670. arXiv: 2103.15670. URL: https://arxiv.org/abs/2103.15670.
Stammer, Wolfgang, Patrick Schramowski, and Kristian Kersting (2020). “Right for the Right Concept: Revising Neuro-Symbolic Concepts by Interacting with their Explanations”. In: *CoRR* abs/2011.12854. arXiv: 2011.12854. URL: https://arxiv.org/abs/2011.12854.
Steiner, Andreas et al. (2021). “How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers”. In: *CoRR* abs/2106.10270. arXiv: 2106.10270. URL: https://arxiv.org/abs/2106.10270.
Stowe, Kevin, Leonardo F. R. Ribeiro, and Iryna Gurevych (2020). “Metaphoric Paraphrase Generation”. In: *CoRR* abs/2002.12854. arXiv: 2002.12854. URL: https://arxiv.org/abs/2002.12854.
Suglia, Alessandro et al. (2020). “Imagining Grounded Conceptual Representations from Perceptual Information in Situated Guessing Games”. In: *CoRR* abs/2011.02917. arXiv: 2011.02917. URL: https://arxiv.org/abs/2011.02917.
Sun, Chen et al. (2017). “Revisiting Unreasonable Effectiveness of Data in Deep Learning Era”. In: *CoRR* abs/1707.02968. arXiv: 1707.02968. URL: http://arxiv.org/abs/1707.02968.
Szegedy, Christian et al. (2015). “Rethinking the Inception Architecture for Computer Vision”. In: *CoRR* abs/1512.00567. arXiv: 1512.00567. URL: http://arxiv.org/abs/1512.00567.
Vargas, Danilo Vasconcellos and Shashank Kotyan (2019). “Model Agnostic Dual Quality Assessment for Adversarial Machine Learning and an Analysis of Current Neural Networks and Defenses”. In: *CoRR* abs/1906.06026. arXiv: 1906.06026. URL: http://arxiv.org/abs/1906.06026.
Vaswani, Ashish et al. (2017). “Attention Is All You Need”. In: *CoRR* abs/1706.03762. arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

Wah, C. et al. (2011). *The Caltech-UCSD Birds-200-2011 Dataset*. Tech. rep.

Xie, Cihang et al. (2017). "Adversarial Examples for Semantic Segmentation and Object Detection". In: *CoRR* abs/1703.08603. arXiv: 1703.08603. URL: http://arxiv.org/abs/1703.08603.

Yi, Kexin et al. (2018). "Neural-Symbolic VQA: Disentangling Reasoning from Vision and Language Understanding". In: *CoRR* abs/1810.02338. arXiv: 1810.02338. URL: http://arxiv.org/abs/1810.02338.

Yu, Youngjoon et al. (2020). "Investigating Vulnerability to Adversarial Examples on Multimodal Data Fusion in Deep Learning". In: *CoRR* abs/2005.10987. arXiv: 2005.10987. URL: https://arxiv.org/abs/2005.10987.

Zhang, Hang et al. (2020). "ResNeSt: Split-Attention Networks". In: *CoRR* abs/2004.08955. arXiv: 2004.08955. URL: https://arxiv.org/abs/2004.08955.

Zhou, Bolei et al. (2018). "Places: A 10 Million Image Database for Scene Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.6, pp. 1452–1464. DOI: 10.1109/TPAMI.2017.2723009.