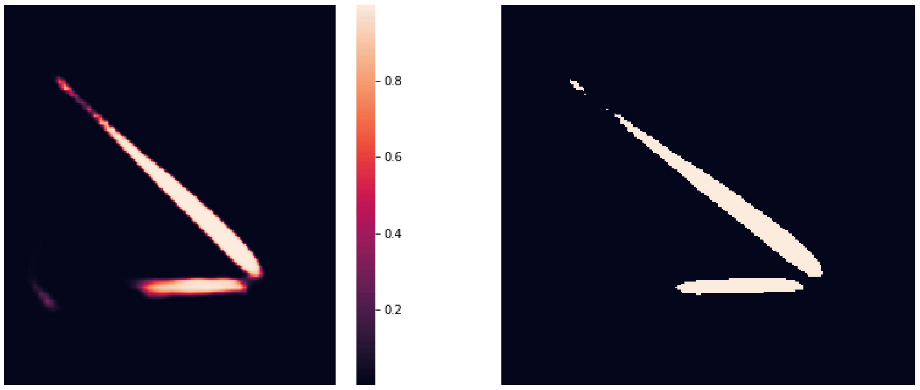

Active learning in deep convolutional neural networks for image segmentation

Evaluating data-centric approaches for improving performance in seat belt localization from images

Master's thesis in Complex adaptive systems & Computer science - algorithms, languages and logic

ISAK SCHWARTZ
WILLIAM ÅKVIST

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Active learning in deep convolutional neural networks for image segmentation
Evaluating data-centric approaches for improving performance in seat belt localization from images
ISAK SCHWARTZ
WILLIAM ÅKVIST

Supervisor: Sheng Huang, Volvo Cars
Co-supervisor: Tomas Björklund, Volvo Cars
Examiner: Giovanni Volpe, Department of Physics

Master's thesis 2022
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Left: output of final, sigmoidal layer, bounding values for each pixel between 0 and 1. Right: output converted to a boolean mask using a threshold of 0.5.

Typeset in LaTeX
Gothenburg, Sweden 2022

Active learning in deep convolutional neural networks for image segmentation
Evaluating data-centric approaches for improving performance in seat belt localization from images
ISAK SCHWARTZ
WILLIAM ÅKVIST
Department of Physics
Chalmers University of Technology

Abstract

The sitting position and seat belt orientation of passengers in automobiles can be crucial in the event of a collision. In order to warn passengers of unsafe positions, deep learning models in the form of neural networks can be used to identify the seat belt from image data. Performance of neural networks can be increased by improving the model (model-centric approaches) or by improving the data used to train the model (data-centric approaches).

In this thesis we compare the segmentation performance gains from model-centric approaches to data-centric approaches including stratified sampling, data balancing, label error reduction and active learning. No new model architecture was found that improved performance, but the model training time was sped up by four times without performance loss. Stratified sampling, data balancing and label error reduction did not improve performance significantly.

In active learning, images to be labeled were selected according to the model's uncertainty. Several uncertainty metrics were used, all leading to an improvement when using active learning. The best result showed that we achieved 95% and 99% of the best baseline performance using 19% and 23% less data respectively.

Keywords: computer vision, image segmentation, active learning, data-centric AI.

Acknowledgements

We would like to thank Sheng Huang, Tomas Björklund, and Herman Johnsson at R&D Analytics & AI at Volvo Cars for their support and guidance throughout our work, and for an efficient onboarding process. We would also like to thank AI Sweden for organizing a network of thesis students, and for access to office space at Lindholmen. Finally, thanks to Giovanni Volpe for acting as examiner of this thesis.

Isak Schwartz & William Åkvist, Gothenburg, June 2022

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AL	Active Learning
CNN	Convolutional Neural Network
DCNN	Deep Convolutional Neural Network
FCN	Fully Convolutional Network
FPN	Feature Pyramid Network
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
LC	Least confidence
MCBN	Monte Carlo batch normalization
MCDO	Monte Carlo dropout
IoU	Intersection over Union
LOOCV	Leave-One-Out Cross Validation
QBC	Query by committee
SGD	Stochastic Gradient Descent

Contents

List of Figures

List of Tables

1
Introduction

This chapter presents the background and context of the thesis.

1.1 Background

In 2012, AlexNet [4], a Deep Convolutional Neural Network (DCNN) achieved a state-of-the-art result in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [5]. Since then, most of the research in computer vision has been focused on deep learning and DCNN's. Deep learning has shown great promise in computer vision tasks such as image classification and object detection [1]. Figure 1.1 illustrates the rapid increases in performance at the ILSVRC challenge from 2011-2017.

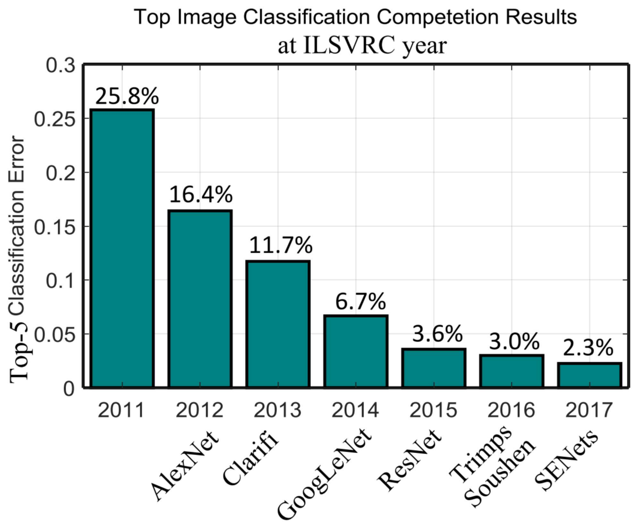

Figure 1.1: Classification error from the winning entries 2011-2017 in the image classification task at the ILSVRC challenge [1].

In each edition of the ILSVRC challenge, breakthroughs in DCNN architecture have driven the increases in performance. The ResNet architecture introduced skip connections for learning in networks with hundreds of layers [6]. GoogLeNet, or Inceptionv1, introduced Inception modules, allowing more efficient computation and enabling deeper networks [7]. We refer to the practice of developing new models in order to increase performance as a model-centric approach.

Although there is still room for increases in performance, in particular for object detection, based on the rate of improvement following ResNet in 2015 it is clear that we face diminishing returns from efforts to develop new model architectures. Thus it is worth considering alternative ways to improve performance. In particular, improving the quality of the training dataset is one approach that has recently seen increasing interest from researchers in the field. We refer to this practice as a data-centric approach.

The traditional, model-centric approach is to keep the data fixed and iteratively improve the model, wheras the data-centric approach is to keep the model fixed and iteratively improve the data.

1.2 Purpose

We will employ techniques on a supervised learning task at Volvo Cars. Volvo Cars are using image data for semantic segmentation, segmenting the passenger seat belt, as seen in figure 1.2.

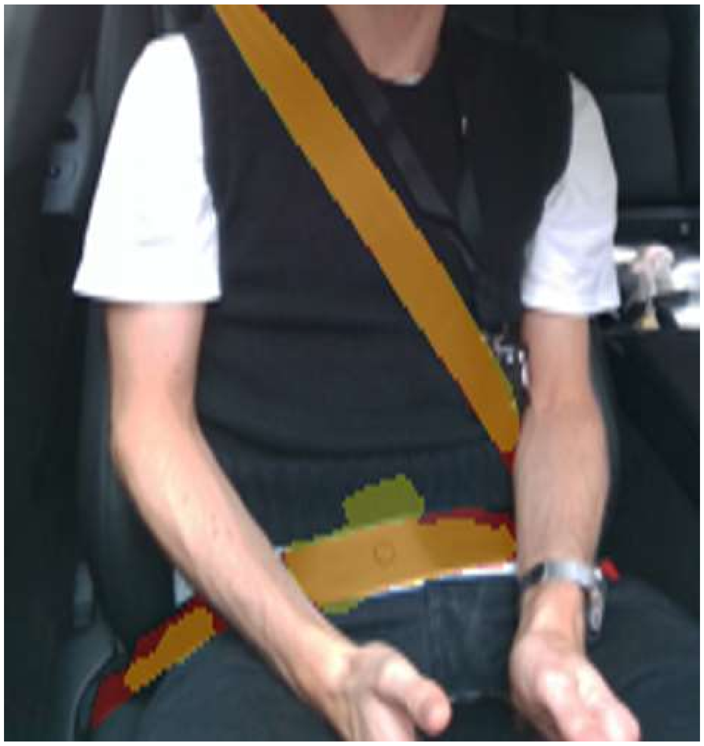

Figure 1.2: Segmentation of passenger's seat belt. Red markings are the true, annotated location and yellow markings are the predicted location of the seat belt.

To develop a successful machine learning model, one would ideally improve both the model and the dataset used to train the model. However, in practice it might be more feasible to concentrate efforts on one of the two approaches. Due to the extensive research that has already been done on model-centric approaches, we will primarily investigate data-centric approaches. The posed research questions are as follows:

- What increases in performance in semantic segmentation follow from model-centric approaches?
- What are the possible data-centric approaches?
  - What increases in performance follow from removing uncertainty in the data?
  - What increases in performance follow from modelling the uncertainty in the data?

The results will be generated using a computer vision model developed at Volvo Cars, applied to a proprietary dataset. Thus we can evaluate our practices in a real-world scenario.

1.3 Previous work

Andrew Ng has recently proposed that if one aims to improve the accuracy of a machine learning model, it might be more beneficial to improve the dataset used to train the model rather than fine tuning the model itself [8][9]. The data can be improved quantitatively or qualitatively.

Improving data quantitatively entails collecting more data. One study has shown that performance in computer vision tasks such as segmentation and classification increases logarithmically with dataset size [10]. However, Andrew Ng exemplifies that spending effort on increasing the amount of data by 200% may only be as beneficial as improving the quality of 12% of the dataset. This result demonstrates that qualitative assessments are important as well. The importance of data collection compared to annotation for image segmentation has been studied in previous work [11]. However, this comparison benchmarked dataset sizes starting from 10000 data points, meaning there is still a need for benchmarks on smaller datasets.

Andrew Ng presents three ways to ensure high data quality. The first two approaches say that data labels should be consistent and accurate. Consistent labeling could be exemplified by two images of cats with labels in the form of bounding boxes, where the bounding boxes should have the same margin around the cats in both pictures. Accurate labels would mean that the bounding boxes are present in every cat picture and that they are not misplaced. The third approach is related to both quality and quantity and says that the dataset should include all important cases. For a cat dataset, important cases could be kittens, adult cats, white cats, black cats and cats without tails.

A data-centric approach with recent developments that can be classified as both quantitative and qualitative is active learning (AL). Active learning entails iteratively labeling data based on what data is informative to a model, rather than randomly labeling a subset of the data initially. Active learning coupled with unsupervised pre-training can reduce time spent on manually labeling in medical image classification tasks by 90% [12]. Similarly, in traffic segmentation tasks different active learning strategies have been benchmarked with results showing possible labeling time reductions of up to 33.6% [13].

1.4 Goals

Our overarching goal is to investigate whether model-centric approaches or data-centric approaches are more beneficial for seat belt segmentation. The goal can be divided into two sub-goals including (1) investigating how much different approaches improve the accuracy of the model, and (2) evaluating how practical each approach is based on a trade-off between accuracy and resources needed.

The approaches we will investigate include:
1. Modifying neural network model architecture.
2. Increasing dataset quality by removing poorly labeled data points.
3. Increasing dataset balance using stratified sampling and balancing.
4. Active learning.

We will benchmark models with different architectures, backbones and numbers of parameters. We will also benchmark the impact of dropout regularization.

As suggested by Andrew Ng, qualitative improvements to our dataset should include ensuring labels are consistent and accurate. This can be done by making sure each data point is labeled according to a contract that establishes how labeling should be done in general cases and edge cases. We will improve quality by removing data points that do not follow said contract.

Data quality will also be investigated in tandem with data quantity through active learning. We will benchmark a subset of the active learning strategies benchmarked in previous work [13] with the contribution being benchmarks in a new domain.

We will investigate the importance of data balance in two ways. First we will make sure the training data contains the same number of images of every person. We will also mirror the distribution of the people in the training data to the distribution in the test data that represents a real world scenario.

1.5 Ethics

Producing a dataset that covers all scenarios that a trained model would encounter in practice can be difficult. For instance, to cover all types of passengers would require a dataset with people of every skin color, of different height and with different clothing styles. Datasets can encode gender and ethnic biases, and machine learning models trained on these datasets can then exhibit these biases themselves [14]. In two prevalent image datasets, Adience and IJB-A, a majority of the identified individuals are light-skinned (86.2% and 79.6% in Adience and IJB-A respectively). An evaluation of gender classification models showed that dark-skinned females were misclassified with error rates of up to 34.7%, wheras light-skinned males were misclassified with a maximum error rate of 0.8% [15]. This stark contrast shows that it is important to verify that the training dataset is representative of the population as a whole, and to verify that the results of the model generalize well to under-represented subpopulations in the original dataset.

2 Theory

In the following sections, the theory behind the methods used are presented.

2.1 Machine learning

A machine learning model is an algorithm that takes input data and should be able to learn and predict the value, or label/annotation, associated with that data. The input data could be images of cats and dogs which should be processed by a model that predicts whether a given image should be labeled as depicting a cat or a dog. An input image can be represented as a feature vector, with elements being the color values for each pixel, and the label or class being a number, 0 for a cat and 1 for a dog.

Sophisticated machine learning models have a set of parameters that determine the behaviour and predictions for the model given an input. These parameters are updated to produce better predictions in the training procedure for the model in a procedure that constitutes the learning process for the model. Models that learn to predict labels by training on data with given labels are called supervised models, and the goal is usually to be able to generalize and predict labels on unseen data.

In order to train and evaluate machine learning models we usually split our available data into three sets; a training set, a validation set and a test set. The training set is used to train the model, the validation set is used to evaluate and select a model during training that is expected to perform well on the test set which is used for final evaluation. The datasets can be expanded and altered in order to improve model performance using augmentation and by modifying the balance of the datasets.

2.1.1 Data augmentation

Data augmentation entails duplicating existing training data with some transformation in order to provide new diverse data when training a model. The new data points are representative for the vicinity of their original data points, hence allowing us to improve model generalization. However, domain expertise is required in order to select suitable augmentations and define a vicinity around each data point.

2.1.2 Stratified sampling and data balance

When randomly splitting a dataset with multiple classes into training, validation and test sets there is a risk that the classes will have different distributions in each set. This could mean that the validation set cannot provide an evaluation that is representative of the test set, and a model trained with said training set might not be able to generalize to the dissimilar validation set. The datasets can be modified to have the same class distribution using stratified sampling [16]. This is done by dividing the entire dataset into subgroups, or strata, one for each class, and then sampling randomly inside each group. The number of data points of each group should reflect the size of the group. A dataset can be stratified over not just classes but also features and characteristics. For a dog breed recognition task one could stratify over the classes (breeds) but also characteristics like fur color.

After stratifying data one could also balance the data which means making sure each class has the same number of data points inside some set. Balancing can be beneficial for models that weigh each data point equally, since minority classes might otherwise be ignored [17]. However, balancing is typically only used on the training set, meaning the distribution will be different to the validation and test set. Hence, balancing might reduce performance. Balancing is typically not used in validation or test sets since the balance in these sets should reflect the way the data is balanced in reality.

Balancing can be done using e.g. oversampling or undersampling [18][19]. Oversampling means replicating data points from scarce classes either directly or using augmentation. Undersampling entails removing data from abundant classes. Undersampling introduces loss of information and diversity from the dataset, but the loss can be minimized by only removing data points with similar features.

2.1.3 Principal component analysis

Principal component analysis (PCA) is a method used to reduce dimensionality of data [20]. Dimensionality reduction for points in 3D space could simply entail projecting the points down to 2D space. In general, a target dimensionality (number of components) is chosen based on how much information in the data is aimed to be preserved, in other words how much variance is explained. Dimensionality reduction is useful for illustrating the characteristics of a given dataset that resides in higher dimensions, for example a collection of images.

2.1.4 K-means clustering

Machine learning can be used to cluster data into multiple separate groups. In k-means clustering an arbitrary number k is chosen which is the number of groups the data should be clustered into [21]. The clustering is initiated by placing k center points (centroids) in the data space and assigning each data point to a group corresponding to its closest centroid. The clustering is improved iteratively by repositioning the centroids until the total sum of distances from each centroid to their assigned data points are minimized. K-means clustering can be done on data without knowledge of the label associated with each data point, which is why it is referred to as an unsupervised machine learning method. Clustering can be useful for getting an understanding of the characteristics of a given dataset.

2.2 Neural networks

Neural networks are machine learning models that consist of interconnected nodes, inspired by the network of neurons in the brain. Each node receives input from some set of nodes, processes it, and sends an output to another set of nodes according to figure 2.1.

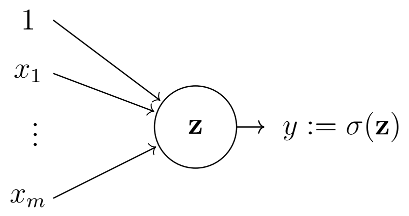

Figure 2.1: Example of a node with input $x_1, \ldots, x_m$, activation function σ, and output y. The activation function is denoted by σ and applied on the weighted sum $\mathbf{z}$ of the input $\mathbf{x}$ and a bias 1 to form its output.

The output of each node is its activation function σ applied to a weighted sum of its inputs including a constant bias term w_0 according to equation 2.1. Each node has an individual weight vector $\mathbf{w}$. These weights are adaptable and are how the network learns.

$$y = \sigma(\mathbf{z}) = \sigma(w_0 + \sum_{j=1}^{m} w_j x_j) \tag{2.1}$$

Nodes are typically aggregated into layers, with each layer performing a different transformation on its input. Formally, a fully connected neural network consists of L layers where layer l applies a non-linear transformation on its input $z^{(l)}$ to produce its output $v^{(l)}$, a vector of units $v_i^{(l)}$:

$$v_i^{(l)} = \sigma^{(l)}(z_i^{(l)}) = \sigma^{(l)}(\sum_{j=0}^{n_{l-1}} w_{i,j}^{(l)} v_j^{(l-1)}) \tag{2.2}$$

An artificial neural network, or multi-layer perceptron, is illustrated in figure 2.2. σ is an activation function. Commonly used activation functions include Rectified Linear Unit (ReLU), equation 2.3, and sigmoid functions, equation 2.4.

$$\sigma(x) = max(0, x) \tag{2.3}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

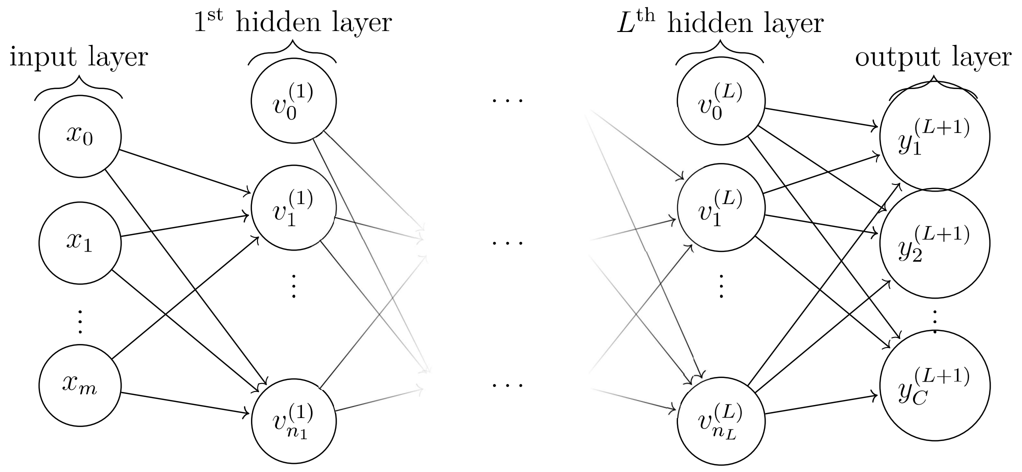

Figure 2.2: Network graph of an $(L+1)$-layer perceptron (artificial neural network) with D input units and C output units. The lth hidden layer contains n_l hidden units. Each node i in hidden layer l is calculated according to $v_i^{(l)} = \sigma^{(l)}(z_i^{(l)}) = \sigma^{(l)}(\sum_{j=0}^{n_{l-1}} w_{i,j}^{(l)} v_j^{(l-1)})$.

Figure 2.2 illustrates a fully connected neural network, where each node in any given layer is connected to each node in the previous layer. Each connection between nodes has a unique weight, leading to a large number of parameters in the network, which must be learned in order to approximate some function. Furthermore, in images with spatial features such as edges and corners, a fully connected neural network would inefficiently have to learn the same feature on each occurrence in the image. Therefore, in order to utilize neural networks on large inputs such as image data, convolutional neural networks (CNN) were introduced.

Many state-of-the-art deep learning models used in computer vision today are some form of convolutional neural network. A convolutional neural network is a neural network that contains at least one convolutional layer.

Due to the large number of inputs in image data (a 224×224 RGB image contains $224 \times 224 \times 3 = 150528$ values), it is unfeasible to utilize fully connected layers. Instead, convolutional layers are used, which are fully connected locally. These layers learn filters that identify features such as edges, corners and other spatial patterns. See figure 2.3.

The example in figure 2.3 illustrates a single 2D filter. In practice, since an image has 3 channels (RGB values), the filter would also operate along the depth of the image. Each convolutional layer also typically has several filters, thus producing a 3D output, with depth corresponding to the number of filters. The resolution of the output can be smaller or equal to the input. The output resolution is smaller when the filter strides over multiple pixels, meaning skipping some possible patches of pixels. The resolution can be preserved by padding the input with zero value elements such that the filter can fit its center over every pixel in the original input,

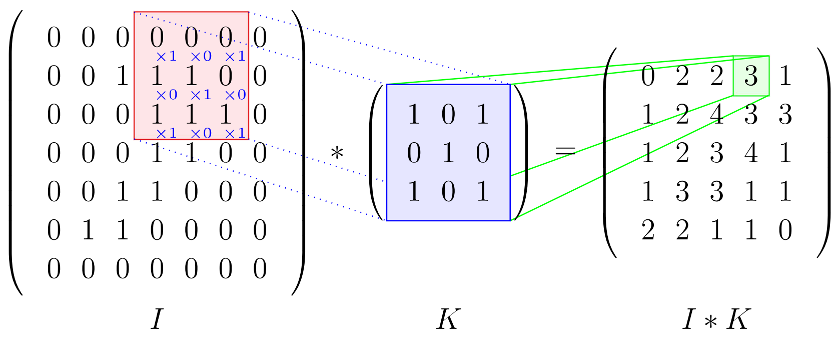

Figure 2.3: Example of a 2D filter K applied to a single-channel, 2D image I. The resolution is preserved between the input image I and the output image $I * K$ even when applying the filter K to different regions of the image by padding I with zeros. The red region is convoluted by the blue filter to produce the value highlighted in green.

most importantly the edge pixels, and stride over all of these exactly once. CNN architectures typically contain several stacked convolutional layers with each layer containing an increasing number of filters. Thus in each convolutional layer, the output feature map is deeper than that of the input, but of a lower resolution.

CNNs also typically contain some form of pooling layer, which downsamples inputs without a learnable filter, instead aggregating by choosing the average or maximum value as output. Downsampling essentially summarizes what features each region contains which reduces the amount of parameters needed in following layers. Figure 2.4 illustrates a maximum pooling layer.

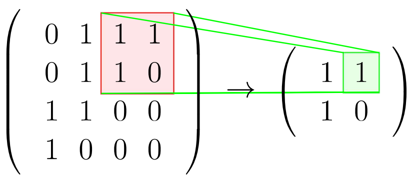

Figure 2.4: Example of a maximum pooling layer.

CNNs were further developed in the form of residual networks [6], which utilize skip connections, as seen in figure 2.5.

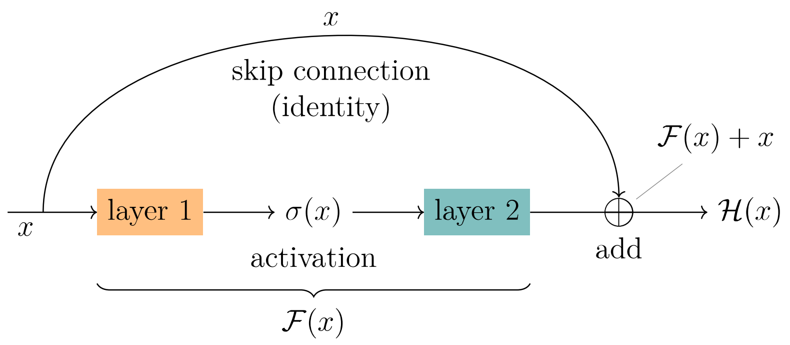

Figure 2.5: Skip connection.

Rather than designing the network so that layers directly fit some mapping, we let the layers fit a residual mapping. Denoting the desired underlying mapping as $\mathcal{H}(x)$, we design the network so that the stacked nonlinear layers fit a mapping of $\mathcal{F}(x) := \mathcal{H}(x) - x$. Utilizing a skip connection, the original mapping is recast into $\mathcal{F}(x) + x$. The idea is that it is easier to optimize the residual mapping $\mathcal{F}(x)$ than to optimize the original mapping $\mathcal{H}(x)$. In the extreme case where the identity mapping would be optimal, it would be easier to push the residual mapping $\mathcal{F}(x)$ to zero than to fit an identity mapping using nonlinear layers. Thus, skip connections enable much deeper networks.

2.2.1 Backpropagation

In order for a neural network to learn a function, one typically uses gradient methods to find model parameters (network weights) that minimize a loss function. These methods require computation of the gradients of the model parameters w.r.t. the loss function. Backpropagation is an algorithm to compute these gradients efficiently in neural networks using the chain rule. Computational effort is saved by reusing values from forward propagation and from previously backpropagated layers in the chain rule. Performing backpropagation on the entire dataset once is referred to as en epoch. The entire dataset can be recycled multiple times for multiple epochs to train a model for a longer period of time.

Consider a neural network $f(\mathbf{x}, \mathbf{W})$ with input vector (including bias) $\mathbf{x} = [1, x_1, ..., x_m]$ and weights $\mathbf{W} = \left[W^{(1)} \dots W^{(L)}\right]$. We can compute the gradient of the loss function $L(\mathbf{W}; \mathbf{x}, \mathbf{y})$ with respect to the weights $W^{(L)}$ in layer L as follows:

$$\nabla_{W^{(L)}}\mathcal{L}(\mathbf{W};\mathbf{x},\mathbf{y}) = \frac{\partial \mathcal{L}}{\partial W^{(L)}} = \frac{\partial L}{\partial f}\frac{\partial f}{\partial W^{(L)}} \tag{2.5}$$

$$\nabla_{W^{(L-1)}}\mathcal{L}(\mathbf{W};\mathbf{x},\mathbf{y}) = \frac{\partial \mathcal{L}}{\partial W^{(L-1)}} = \frac{\partial L}{\partial f}\frac{\partial f}{\partial z^{(L-1)}}\frac{\partial z^{(L-1)}}{\partial W^{(L-1)}} \tag{2.6}$$

$$\vdots$$

$$\nabla_{W^{(i)}}\mathcal{L}(\mathbf{W};\mathbf{x},\mathbf{y}) = \frac{\partial L}{\partial f}\frac{\partial f}{\partial z^{(L-1)}}\frac{\partial z^{(L-1)}}{\partial z^{(L-2)}}\cdots\frac{\partial z^{(i+1)}}{\partial z^{(i)}}\frac{\partial z^{(i)}}{W^{(i)}} \tag{2.7}$$

2.2.2 Optimization

Having computed gradients using backpropagation, gradient methods can be used to optimize the weights with respect to the loss function. One often uses stochastic methods such as stochastic gradient descent (SGD). These methods are stochastic in the sense that network weights are updated after backpropagating a randomly selected portion of the data in the form of a batch. Network weights are updated with a step size proportional to the gradient, using a learning rate η:

$$\mathbf{W} \leftarrow \mathbf{W} - \eta_t \nabla_{\mathbf{W}}\mathcal{L}(\mathbf{W};\mathbf{x},\mathbf{y}) \tag{2.8}$$

In this project, we utilize the Adam optimizer [22]. Rather than keeping the learning rate η fixed as in gradient descent, Adam computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. The algorithm calculates an exponential moving average of the gradient and the squared gradient, with parameters β_1 and β_2 controlling the decay rates of these moving averages. After correcting the averages for bias, they are used to update the parameters.

2.2.3 Regularization

A common issue when training machine learning models is how to balance underfitting and overfitting. Underfitting means the model is too simple and unable to learn complexities in data. Overfitting is the opposite problem where the model is so complex that it can learn diverse training data, but also too complex to manifest a general understanding that translates well to new data. Underfitting can be mitigated by using more parameters in the model while overfitting is mitigated using what is called regularization strategies.

Regularization can be done by punishing complex models, by exposing models to more diverse data and by designing models to be less complex inherently. A Complex model can be punished in the loss function that scores a model by reducing score based on the length of the model parameter vector. Diverse data used to force models to generalize can be introduced using augmentation. Two ways of designing an inherently less complex model is by using batch normalization and dropout.

Algorithm 1 Adam

```
Require: $\alpha$: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters $\theta$
Require: $\theta_0$: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize first moment vector)
  $v_0 \leftarrow 0$ (Initialize second moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while $\theta_t$ not converged do
      $t \leftarrow t + 1$
      $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep $t$)
      $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ (Update biased first moment estimate)
      $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ (Update biased second raw moment estimate)
      $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
      $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
      $\theta_t \leftarrow \theta_{t-1} - \alpha \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return $\theta_t$ (Resulting parameters)
```

The dropout procedure entails randomly omitting nodes in a neural network, as illustrated in figure 2.6. In each forward pass, and in each dropout layer independently, nodes to be dropped out are selected randomly (according to a Bernoulli distribution) with probability p. The idea behind dropout as a regularization technique is that we force the model to be able to make accurate predictions even if some nodes are omitted. Why these omissions may produce a less complex model is that nodes might not be able to specialize on a specific type of feature in the training data, instead nodes may have to generalize and be able to detect more features [23].

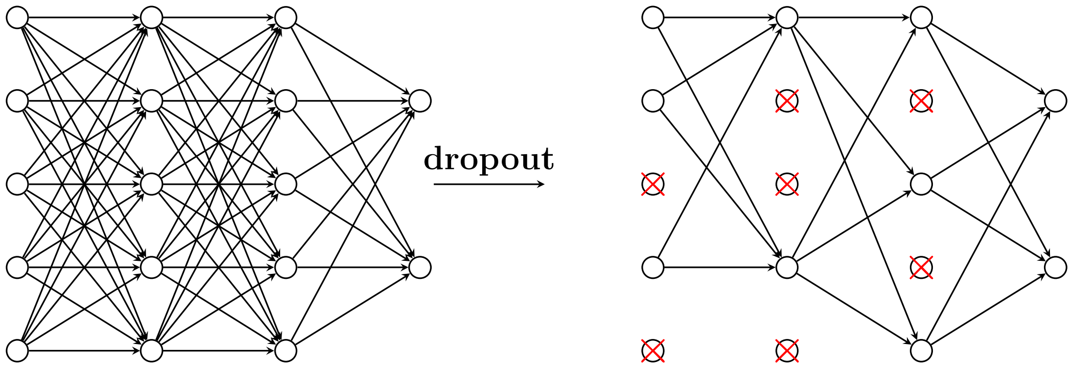

Figure 2.6: Fully connected network before (left) and after (right) dropout in layers 1, 2, and 3.

Batch normalization regularizes models by randomly altering the values that make up each training data point. Randomizing data point values may reduce the model's ability to memorize data and exploit this knowledge the next time the data point is used in training. The values for each training data point is altered by normalizing over every image inside each training batch. Normalization can be done by shifting and scaling the values for the data points such that are distributed with 0 as mean and standard deviation 1 [24].

2.3 Image segmentation

In computer vision, image segmentation is the task of partitioning an image into multiple segments, typically locating objects and boundaries. A class label is assigned to each pixel in the image. The number of output classes is pre-determined. In this project, there are 2 possible class labels: seatbelt and background.

2.3.1 Fully convolutional networks (FCN)

CNNs such as AlexNet, the VGG net and GoogLeNet were trained for the task of image classification, with a single image-level label as output. Fully convolutional versions of these networks, in contrast, predict dense outputs from arbitrary-sized inputs, i.e. they make a prediction at each pixel [25]. This is done by replacing fully connected layers near the end of the network with convolutional layers, preserving spatial coordinates.

One prevalent family of fully convolutional networks is DeepLab, the latest version being DeepLabv3+ [26].

Fully convolutional networks are an application of transfer learning, using a pre-trained model for a task different from what it was originally trained for. Transfer learning enables the use of deep learning in tasks with limited data. In this project, we train on our model on a dataset of 259 images, whereas the pre-trained ResNet encoder was trained on ImageNet, a data set consisting of more than 14 million labeled images. Thus, transfer learning enables us to quickly train deep learning models on small data sets and achieve good results.

2.3.2 Feature Pyramid Networks (FPN)

Feature Pyramid Networks [2] are a type of fully convolutional network. They extend pre-trained CNNs (referred to as encoder) and combine high resolution features with low resolution features using top-down and skip connections as seen in figure 2.7. Rather than creating a single, high resolution feature map to make predictions, predictions are made on each feature level separately and aggregated.

We use an FPN model implemented using the PyTorch framework (see appendix A), with a ResNet-18 encoder pre-trained on the ImageNet dataset [27].

The neural network used in this work is thus a function:

$$\sigma : \mathbb{R}^{H \times W \times 3} \to [0,1]^{H \times W} \tag{2.9}$$

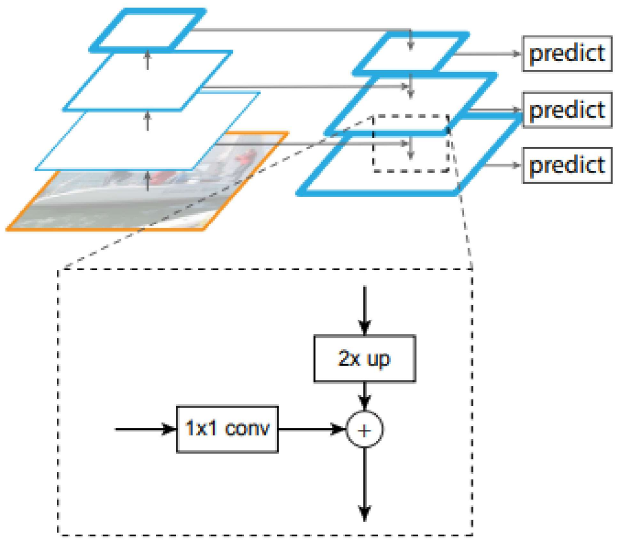

Figure 2.7: Top-down and skip connections between feature layers [2]. Blue borders indicate feature strength, thicker borders indicate stronger features.

The output of the final layer with a sigmoidal activation function and the conversion to a boolean mask using a threshold value of 0.5 is illustrated in figure 2.8.

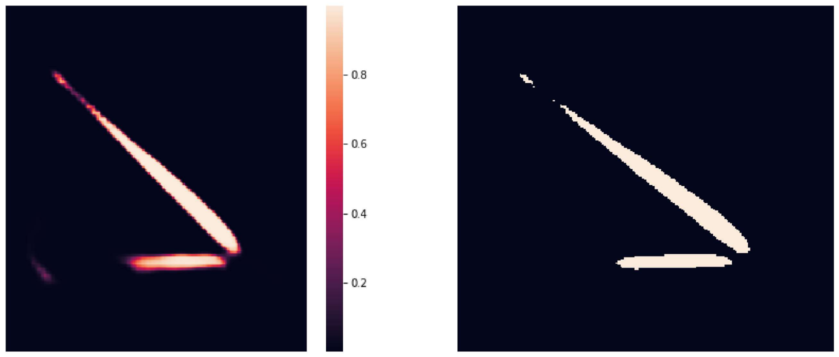

Figure 2.8: Left: output of final, sigmoidal layer, bounding values for each pixel between 0 and 1. Right: output converted to a boolean mask using a threshold of 0.5.

2.3.3 Loss function

In order to quantify how well a neural network produces the desired output given an input we use a loss function. The loss function used during training in this project leverages Binary Cross Entropy (BCE), according to equation 2.10.

$$Loss_{BCE} = Y_i * \log \hat{Y}_i + (1 - Y_i) * \log (1 - \hat{Y}_i) \tag{2.10}$$

Y_i is the true label and $\hat{Y}_i$ is the predicted value. Since we have a single output class, we can calculate the BCE loss at each pixel and take the average across the whole image as the loss for an image. We then utilize the average across all images in a batch in our backpropagation algorithm.

2.3.4 Metrics

In object detection and image segmentation, a commonly used metric is Intersection over Union (IoU), also known as the Jaccard index. If we consider sets A and B as the sets of pixels classified as true in the ground truth annotation and the model's output respectively, the IoU metric can be calculated as follows:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{TP + FP + FN} \tag{2.11}$$

The IoU metric is bound between 0 and 1 by design. The calculation of the metric is further illustrated in figure 2.9.

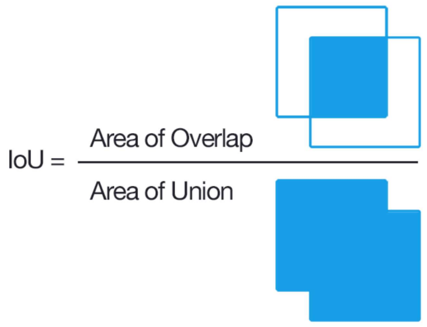

Figure 2.9: Illustration of the Intersection over Union (IoU) metric [3]. The boxes represent portions of the image classified as positive.

2.3.5 Image label quality

The performance of a model can be bottlenecked by two types of label quality issues, inaccuracy and inconsistency. If all training data points are labeled inaccurately, those inaccuracies are likely to manifest in the predictions of the trained model. The second issue is inconsistent labels, which introduces ambiguity. If different humans annotate the dataset such that label standards deviate to some percentage, then these deviations will likely be present in model predictions as well [28].

Label accuracy and consistency can be quantified by comparing the labels to a given labeling standard manually by eye and documenting deviations. Alternatively, one can make new annotations for data points with existing annotations and compare these two to quantify quality numerically using IoU. One can either measure how inconsistent each single labeler is when repeatedly annotating some image or how much annotations from different labelers conflict on some image.

2.4 Weakly supervised learning

Weakly supervised learning, also known as data programming, is an alternative to manually labelling all data that can be used to quickly create large training datasets [29].

Weak supervision has been used in computer vision for image classification, using image-level annotations as input to the models. In image classification, an image-level annotation declares the class of object that is present in the input image. Weak supervision is especially popular for semantic segmentation, where labelling data is time consuming, expensive, and prone to errors due to ambiguous labelling instructions [30]. One study crowd sourced the annotation of images, utilizing "block annotations", where only a portion of the image is annotated. The study found that annotations could be crowd sourced at half the cost and with a lower error rate, achieving equivalent performance as in the fully supervised case [31].

2.4.1 Annotations

There are several categories of annotations for image segmentation [32]. A full annotation contains information about segment location, shape, and class. In contrast, weak annotations only contain parts of this information: image-level tags provide class information; point supervision provides class and approximate location, bounding boxes and scribbles provide class, approximate location and extent.

Weak annotations are more versatile and easier to collect, and thus better for increasing the size of the training set. Since weak annotations do not provide exhaustive information, weak supervision requires specific training strategies.

2.5 Semi-supervised learning

Another approach for image classification is semi-supervised learning, where one has a labeled and an unlabeled dataset. The approach involves a teacher model that is trained on the labeled dataset, and a student model that is trained on the top data points of the teacher model's predictions on the unlabeled dataset, for each class in the labeled dataset [33]. When it comes to image segmentation, semi-supervised learning implies that the model is trained using either a combination of labeled and unlabeled data [34], or a combination of strong (pixel-level) and weak (image-level) labels [35].

It is important to distinguish between weak and semi-supervised learning for the task of image segmentation, and for the task of image classification. In [33], the backbone (ResNet) performance in image classification is improved even further by using a semi-supervised approach while training on an additional unlabeled dataset. While using such a backbone as part of a model such as DeepLab has been shown to increase performance in image segmentation [30], the problem of semi-supervised or weakly supervised learning for image segmentation is a separate problem.

2.6 Active learning

Traditionally in machine learning, data points to be labeled are chosen randomly and exhaustively. Training a model on randomly selected data points could lead to a model with low predictive accuracy since data points that do not help the model improve could be selected for labeling. Instead, one can use active learning, where data points to be labeled are selected based on their informativeness, i.e. how much useful information a data point is expected to provide the model during training in order to improve performance.

Active learning consists of two entities, a human user called the "teacher" who labels data points and the "learner" which is an algorithm that trains a model and selects data points that are queried to the teacher for labeling. The different methods used to select data points for querying are known as query strategies. Once a data point has been queried and labeled, it is added to the training set, and the machine learning model is trained and evaluated. If the learner is not satisfied with the results, more data points will be queried and the model will be retrained and evaluated on the new training set. This iterative procedure continues until the learner decides to stop.

With the right query strategy, active learning can outperform passive learning, where data points to be labeled are chosen randomly. Most importantly, active learning is an efficient way to make sure that time is not wasted on labeling non-informative instances thus saving both time and money. This aspect is particularly important in semantic segmentation, since labeling pixels in images accurately is especially time consuming.

Whether a given query strategy selects for informative data points has been estimated using IoU as a heuristic in previous work [36]. The idea behind this heuristic may be that tough data points with low IoU contain information or concepts the model has not learned yet. Therefore, a good active learning approach should prioritize data points with low IoU. But since we cannot know what the IoU is for a data point before we have labeled it, we need to employ query strategies that can rank unlabeled data points with a metric that correlates with IoU.

2.6.1 Query strategies

Common query strategies include uncertainty querying, error/variance reduction, model change maximization and query by committee/disagreement (QBC/QBD).

Querying can be done based on expected model change, meaning how much a labeled data point would update the model weights in back propagation. The expected model change for a data point can be calculated as the expected length of the model parameter gradient given every possible labeling of the data point according to equation 2.12 [37]. L is the set of all possible labels l, $p(l)$ is the probability the model assigns to label l and $\nabla_l E(\mathbf{W})$ is the gradient given that the label l is assigned to the data point in question.

$$\textit{Expected gradient length} = \sum_{l \in L} p(l) ||\nabla_l E(\mathbf{W})| \tag{2.12}$$

Computing every possible gradient length can be feasible for tasks like classification where only a few labels are possible. But for our segmentation task, expected model change estimation is not feasible without major simplification since the number of possible labels for a single data point is 2^N where N is the number of pixels in the data point. The number of possible labels can be reduced arbitrarily to 2^M by dividing each data point into M patches that are assigned one label each, instead of one label per pixel. The complexity of the problem can be reduced further by employing graph-cut algorithms, but the use of patches remains [38]. Using patches means the outputs from the model are very rough for small M, meaning it is not feasible for our safety-critical task.

Similarly, querying based on error reduction or variance reduction builds on the premise of estimating expected change for every possible labeling for each data point [39][40]. Once again, iterating over many possible annotations is infeasible for us.

Query by committee (QBC) is a query strategy wherein multiple models trained in parallel are used to predict the label of a data point [41]. The data point with the highest disagreement among the models is chosen for labeling, the same way as for uncertainty querying.

Uncertainty querying means choosing to label data points for which the model produces uncertain predictions. Uncertainty can either be estimated from a single prediction or as the disagreement between multiple predictions. Uncertainty querying can be combined with representation querying in which data points that are similar to many other data points are ranked higher [36]. We do not use representation querying since our available training data pool has already been constructed such that data points are dissimilar to each other. However, we do use uncertainty querying.

2.6.2 Model uncertainty

In active learning, we seek to label the most informative data points, and we can make use of model uncertainty about the unlabeled data points in order to determine informativeness. Thus, we require a model that produces model uncertainties.

Model uncertainty is highly useful, but most deep learning models do not offer this information. We can however get model confidence. In classification problems, models output a vector where each element represents the confidence in the corresponding label. These confidence scores are simply the outputs from the final softmax/sigmoidal layers, but are often misinterpreted as model certainty [42]. In fact, a model can be uncertain despite a high output confidence. Even so, two querying methods are based on output confidence vectors, least confidence querying (LC) and smallest margin querying. In least confidence querying the data point with the smallest confidence in the predicted label is chosen, meaning the data point with the smallest softmax/sigmoidal output for the predicted label. Smallest margin querying is identical to least confidence querying in binary classification where data points are queried based on the margin between the labels with the most confidence and second most confidence.

A different way to capture model uncertainty is using a family of neural networks studied extensively during the 1990s, Bayesian neural networks. These models have not caught on (likely due to their limited practicality) and are not commonly used in state-of-the-art deep learning in computer vision [43].

A third way to estimate model uncertainty is using stochastic regularization techniques like dropout and batch normalization [44]. These scale well with large amounts of data and can be applied to complex, state-of-the-art models such as CNNs and RNNs. The idea behind querying strategies based on these techniques is to introduce randomness in a model and predict the label for a data point multiple times and then quantify the uncertainty as the disagreement between the predictions. For Monte Carlo dropout (MCDO), randomness is introduced by randomly dropping different nodes for each prediction. For Monte Carlo batch normalization (MCBN) we introduce randomness by randomly normalizing the data point with a new batch for each prediction.

We implement Monte Carlo dropout by applying dropout to ResNet residual blocks 2, 3, 4, and before every upsampling layer in the feature pyramid. In existing CNN literature dropout layers are primarily implemented after inner-product layers at the end of the model. In our model, using multiple dropout layers, we approximate Bayesian inference over the full model.

The technique is then to perform T forward passes through the network, yielding varying predictions due to the dropout applied independently for each forward pass, from which we obtain pixelwise means and variances. We refer to this method as Monte Carlo (MC) dropout. Although the training time is unchanged compared to the model without dropout, the test time is scaled by a factor T.

Monte Carlo dropout works on the principle of variational inference. The goal is to obtain a posterior distribution $p(\mathbf{W} \mid \mathbf{X}, \mathbf{Y})$ over the weights $\mathbf{W}$ of the neural network, given the training data points $\mathbf{X}$ and the corresponding labels $\mathbf{Y}$. However, the posterior is intractable and hence an approximation to the posterior $q(\mathbf{W})$, known as the variational distribution, is defined. We aim to minimize the Kullback-Leibler (KL) divergence (equation 2.13) between the true posterior and the variational distribution.

$$KL(q(\mathbf{W}) \mid\mid p(\mathbf{W} \mid \mathbf{X}, \mathbf{Y})) \tag{2.13}$$

The authors of [45] showed that training a neural network with dropout and standard cross-entropy loss minimizes the Kullback-Leiber (KL) divergence where $q(\mathbf{W})$ is chosen to be a Bernoulli distribution, which is the effect of the dropout layers.

2.6.3 Uncertainty metrics

Epistemic uncertainty, also known as model uncertainty, represents what the model does not know due to insufficient training data. This kind of uncertainty can be decreased by adding data to the training set. Aleatoric uncertainty is caused by noise in the data and can be decreased by increasing the sensor precision. Together, the two uncertainties combined compose the predictive uncertainty of the network [43].

The predictive entropy, $\mathbb{H}$, is the average amount of information contained in the predictive distribution, and captures predictive uncertainty which combines both epistemic and aleatoric uncertainties. It can be defined as follows:

$$\mathbb{H}[y|\mathbf{x}, \mathcal{D}_{train}] = -\sum_c p(y = c|\mathbf{x}, \hat{w}_t) \log p(y = c|\mathbf{x}, \hat{w}_t) \tag{2.14}$$

and approximated by averaging the probabilities of the class over T forward passes as follows:

$$\hat{\mathbb{H}}[y|\mathbf{x}, \mathcal{D}_{train}] = -\sum_c \Big(\frac{1}{T}\sum_t p(y = c|\mathbf{x}, \hat{w}_t)\Big) \log \Big(\frac{1}{T}\sum_t p(y = c|\mathbf{x}, \hat{w}_t)\Big) \tag{2.15}$$

where $p(y = c|\mathbf{x}, \hat{w}_t)$ is the softmax probability of input $\mathbf{x}$ being in class c, and $\hat{w}_t$ are the network weights on forward pass t.

The predictive entropy attains its maximum value when all classes are predicted to have equal uniform probability, and its minimum value of zero when one class has probability 1 and all others probability 0 (i.e. the prediction is certain).

The mutual information, $\mathbb{I}$, between the prediction, y, and the posterior over the model parameters, w, captures epistemic or model uncertainty and is defined as

$$\mathbb{I}[y, w|\mathbf{x}, \mathcal{D}_{train}] = \mathbb{H}[y|\mathbf{x}, \mathcal{D}_{train}] - \mathbb{E}_{p(w|\mathcal{D}_{train})}\Big[\mathbb{H}[y|\mathbf{x}, w]\Big] \tag{2.16}$$

and approximated as

$$\hat{\mathbb{I}}[y, w|\mathbf{x}, \mathcal{D}_{train}] = \hat{\mathbb{H}}[y|\mathbf{x}, \mathcal{D}_{train}] + \frac{1}{T}\sum_{c,t} p(y = c|\mathbf{x}, \hat{w}_t) \log p(y = c|\mathbf{x}, \hat{w}_t) \tag{2.17}$$

where we are taking the expectation of the predictive entropy by averaging over the T forward passes. Note that in equation 2.15 we substitute class probabilities, $p(y = c|\mathbf{x}, \hat{w}_t)$, with the average over T forward passes, whereas in equation 2.17 we average the predictive entropy itself over T forward passes.

Data points $\mathbf{x}$ that maximize the mutual information are points on which the model is uncertain on average, yet there exist model parameters that erroneously produce predictions with high confidence.

It is our view that mutual information is the most applicable metric for uncertainty sampling, since it captures the epistemic uncertainty. Our intuition is that it makes sense to label data points where the epistemic uncertainty is high, since this means that the model is uncertain due to a lack of knowledge and not due to e.g. poor image quality.

3 Methods

This chapter presents the methods used in order to generate results.

We detail preparation methods related to data collection and processing, as well as general experiment configurations related to model training. Lastly we explain how active learning benchmarks are conducted in order to evaluate this data-centric approach.

3.1 Data collection

The data used in this project is a collection of proprietary datasets collected by Volvo Cars for research purposes. Images were taken using a Intel RealSense Depth Camera D415, although the depth information is not used in this project.

The dataset was gathered by putting test subjects in the passenger seat and going for a 1 hour drive around Volvo Cars HQ in Torslanda, Sweden. Each dataset, collected in summers of 2020 and 2021 respectively, contains thousands of images of 11 test subjects who went for a ride. For each test subject, a subset of frames were selected for annotation, explained further in the next section.

Each test subject has signed a waiver allowing Volvo Cars to process their personal data (images) for research purposes. One challenge to expand the data set is then data privacy, since each new test subject involves actions to be taken with regards to data privacy.

3.2 Data splitting

The data used was split prior to our contribution. The labeled data is split into three sets, training, validation and test with 259, 66 and 217 images respectively. The data contains images of 11 subjects that are included in each of the three subsets. The distribution of the subjects in each subset is illustrated in figure 3.1.

The images used for the training and validation sets have been selected using PCA and k-means clustering. First PCA was performed for each subject with every frame in greyscale format. The number of components used were the amount of components needed to explain 75% of the variance in each subject's images. Then k-means

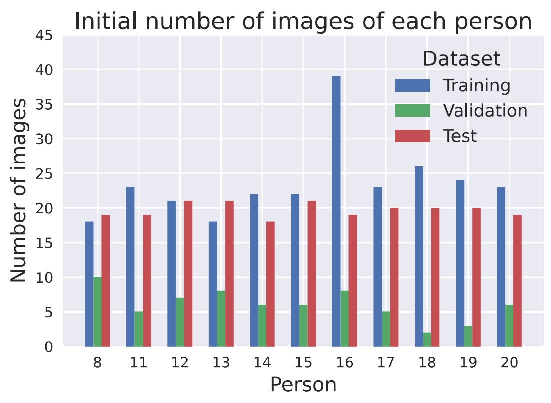

Figure 3.1: Initial number of images of each person in the training, validation and test sets.

clustering was performed on the data which showed that around 30 clusters was optimal for grouping the images. Hence, one image was picked randomly from each cluster, in an attempt to represent the entire dataset with high diversity. These 30 images were divided between training and validation set with a 80:20 split.

The images for the test set were chosen as every 500th frame of the recorded footage of each person. This test set was used in order to evaluate if the model can segment the seat belt for a person throughout an entire driving session.

3.3 Data processing

Images were labeled using COCO annotations as JSON files [46]. See appendix B. These annotations are used to create a ground truth mask of the seat belt for each annotated image. These are used as labels in order to train the model and calculate IoU scores.

Images are resized from 480 x 848 to 224 x 224 to speed up the training process and standardized to $\mathcal{N}(0, 1)$ as is required for inputs to ResNet.

In this project, typical augmentations for image data are used; resizing, rotation, cutout of random image patches, spatial distortion and adjustment of brightness, contrast, hue and saturation. Augmentations were implemented using the library Albumentations [47].

3.4 Model training

For each experiment we train models using a mostly fixed set of hyperparameters, as seen in table 3.1.

Hyperparameter	Value
Batch size	4
Learning rate	0.0025
Encoder learning rate	0.0020
Number of epochs	80
Encoder depth	5

Table 3.1: Hyperparameters used during training.

The encoder, ResNet-18, is pre-trained and the learning rate is set slightly lower than that of the decoder, which is trained from scratch.

The models are trained and tested on fixed datasets in order to get a fair comparison, except for data-centric approaches where only the test set is fixed.

For model-centric approaches different encoders as well as different model architectures, i.e. encoder-decoder networks, are compared. To compare architectures we train the models for 50 epochs to see which has better IoU score and is faster to train. The architectures benchmarked include ResNet and DeepLabv3 of different sizes, as shown in section 4.1.

3.5 Data-centric approaches

Data-centric approaches keep the model fixed and alter the data used to train the model.

3.5.1 Cross-validation

We aim to obtain an estimate of the model's performance in a general setting, i.e. the model's performance on images captured with the same camera and setup on unseen test subjects. If we validate the model's performance against a dataset containing test subjects that were also in the training set, the estimate of the model's performance will be biased. In order to obtain an unbiased estimate, we utilize Leave-One-Out Cross Validation (LOOCV) by validating on one held-out test subject and training on the remaining test subjects in each split.

The procedure used is illustrated in figure 3.2.

For each split, a new model is trained on the training set of the split, and evaluated on the validation set of the split. Thus, the IoU on the validation set is an indicator of the model's performance on unseen test subjects. The results are shown in section 4.1.1.

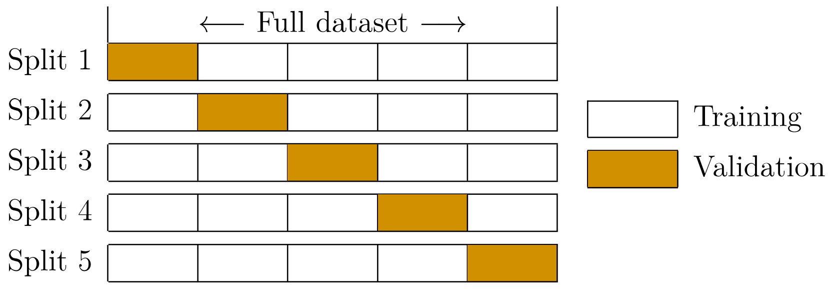

Figure 3.2: Visualization of the cross validation procedure. Each cell represents the set of all images of a test subject.

3.5.2 Active learning

To evaluate active learning as a data-centric approach, we compare the test set IoU of a model trained on a data set queried with active learning to a model trained on a randomly queried data set (referred to as the baseline). Both querying methods sample images from the original training set which means we use existing labels instead of making new annotations. Each time images are queried, we pick the same number of images of each test person in order to keep the training set balanced.

In order to evaluate the uncertainty metrics presented in section 2.6.3, we visualize the uncertainty and performance of the model on each image in the training set in the form of scatter plots. We expect to see some negative correlation between model uncertainty and model performance, i.e. the model should perform poorly on images with high uncertainty.

We will benchmark ten different active learning query strategies. We will implement least confidence querying and every combination of mutual information, predictive entropy and variance in conjunction with Monte Carlo batch normalization, Monte Carlo dropout and query by committee. Each experiment will be repeated ten times to get a robust result, except for query by committee experiments which will be repeated five times.

We use a dropout backbone for all out active learning benchmarks since it is required for Monte Carlo dropout query methods. The dropout backbone also constitutes a regularization that could increase performance by itself. Thus we must investigate the performance impact of the dropout backbone and active learning separately in order to tell if active learning is actually useful. We first investigate the performance impact of dropout by comparing two baselines with and without dropout without using active learning. After the performance impact of dropout is established we compare active learning performance to the baseline (with or without dropout) that performs the best. The dropout benchmark is shown in section 4.1 and the active learning benchmarks are shown in section 4.2.2.

3.5.2.1 Uncertainty metrics

We use dropout rates of $p = 0.5$ in both the encoder and decoder dropout layers and for Monte Carlo methods we perform $T = 25$ forward passes of each image in order to sample the posterior distribution of network weights. For query by committee, we train five models in parallel. For each pixel in each input image, we then calculate the uncertainty metrics presented in section 2.6.3 and the variance of the sampled predictions. Using these uncertainty maps, we draw heatmaps in order to visualize the location of uncertainties in each image, and aggregate pixelwise uncertainties by means of averaging in order to obtain overall uncertainty of each image. The results are shown in section 4.2.1.

3.5.3 Stratified sampling and data balancing

We balance and stratify our data to see if this cheap approach can improve performance. We perform two experiments, one where we only stratify the data and one where we stratify and balance the data. We repeat each experiment ten times to get a robust result showing the IoU score for each dataset.

We stratify the data by matching the distribution of the data in each set. We match the distributions by altering the training and validation set such that the relative number of images of each person is the same as for the test set. We alter the number of images without losing any data using duplicate upsampling, meaning we duplicate images of some people in the training and validation set until they have enough images.

We balance data in the training set using duplicate upsampling as well. We upsample by first finding the highest number of images of any person in the training set, and upsample for the other people to match this maximum. The results are shown in section 4.2.3.

3.5.4 Removing data points with bad annotations

We perform one experiment where images with bad annotations are removed from the training set. To produce a robust benchmark indicating whether this cheap data-centric approach can improve performance we measure the IoU ten times and compare it to a baseline with the original training set. The images we remove are chosen by manually inspecting each data point and discarding those that deviate from the intended labeling. The results are shown in section 4.2.4.

4
Results

In this chapter, results of the proposed approaches, model-centric and data-centric, are presented. For each approach we present the results of several experiments, comparing results to a baseline.

4.1 Model-centric approaches

Performance of different models and network architectures are illustrated in figure 4.1 and figure 4.2.

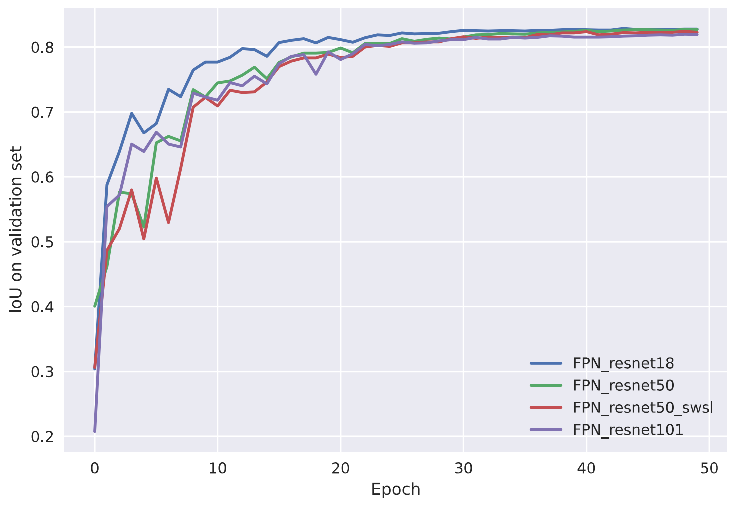

Figure 4.1: IoU scores on validation set during training for FPN models.

The FPN model architecture with ResNet-18 as the encoder performs best and converges the fastest on the training set, and also performs best on the test set, as seen in table 4.1. The model is also trained significantly faster, training at a rate of 23 iterations/second, whereas the same architecture with ResNet-101 as the encoder trains at a rate of 6 iterations/second. The training rate is inversely proportional to the number of parameters in the encoder network.

Although the models using an encoder with more parameters perform slightly worse on the test set, more work is necessary to evaluate performance. However, moving

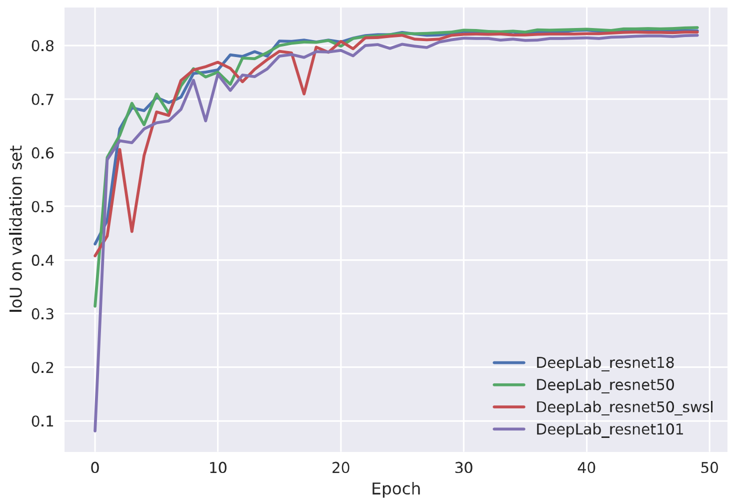

Figure 4.2: IoU scores on validation set during training for DeepLab models.

Model architecture	Encoder	No. of parameters	Test set IoU
FPN	ResNet-18	11M	80.67%
FPN	ResNet-50	23M	80.37%
FPN	ResNet-50-swsl	23M	79.86%
FPN	ResNet-101	42M	80.07%
DeepLabv3+	ResNet-18	11M	80.65%
DeepLabv3+	ResNet-50	23M	80.28%
DeepLabv3+	ResNet-50-swsl	23M	80.08%
DeepLabv3+	ResNet-101	42M	79.85%

Table 4.1: IoU scores on the test set for different models.

forward, ResNet-18 is used as encoder to allow faster training.

To see if dropout affects model performance we compare IoU score for a ResNet-18 model trained with dropout in both the encoder and decoder to a ResNet-18 model without dropout. The results are shown in figure 4.3. Both models were trained ten times to account for variance in model performance and to get a robust result. The IoU was calculated with between 5% and 100% of the training data. For each training subset we show the mean IoU across the ten repetitions as well as the five to 95 percentile range of the ten IoUs.

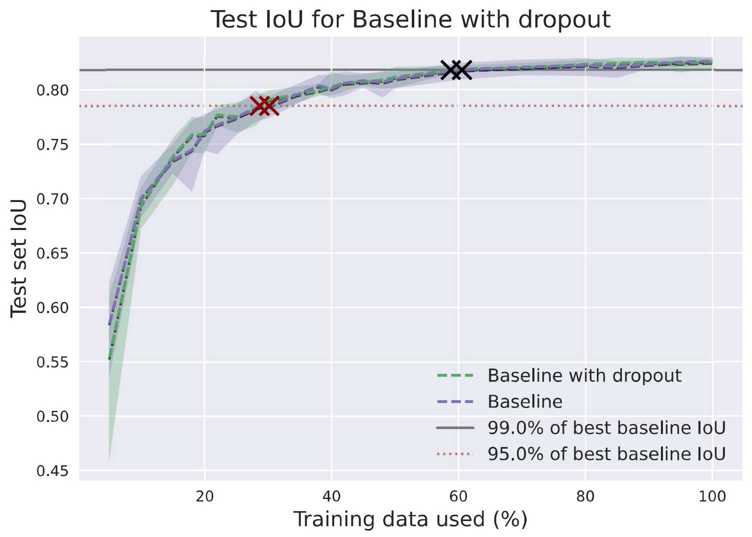

Figure 4.3: Test set IoU for two models using ResNet-18 with and without dropout layers in both encoder and decoder. The model with dropout reached 95% and 99% of the best IoU achieved without dropout using less data.

Introducing dropout layers in the model slightly increased performance. The performance of the models can be quantified by comparing how much data each model needed to achieve 95% and 99% of the highest IoU achieved by the model without dropout. The dropout model required 2.9% less data to reach the 95% IoU threshold and 4.9% less data to reach the 99% IoU threshold. We picked the most competitive baseline, the one with dropout, for the following active learning benchmarks.

4.1.1 Cross-validation

Results of the LOOCV procedure are illustrated in figure 4.4.
The results in figure 4.4 indicate that the model performs adequately on most unseen test subjects, with an average IoU of 0.67 across the 11 folds. However, test subjects 8 and 18 with IoU of 0.47 and 0.18 respectively significantly decrease the mean performance. Thus, the results indicate that a model trained on the existing dataset may perform well on some unseen test subjects, but suddenly fail when presented with a new test subject. It should be noted that the intention of this model is not to perform well on unseen test persons in an Occupant Monitoring System. Rather the purpose of this data set is to to be able to auto-annotate the remaining frames in each of the videos. So that the identified postures can be used in safety studies and simulations at Volvo Cars Safety Centre.

4.2 Data-centric approaches

Results of the methods presented in section 3.5.

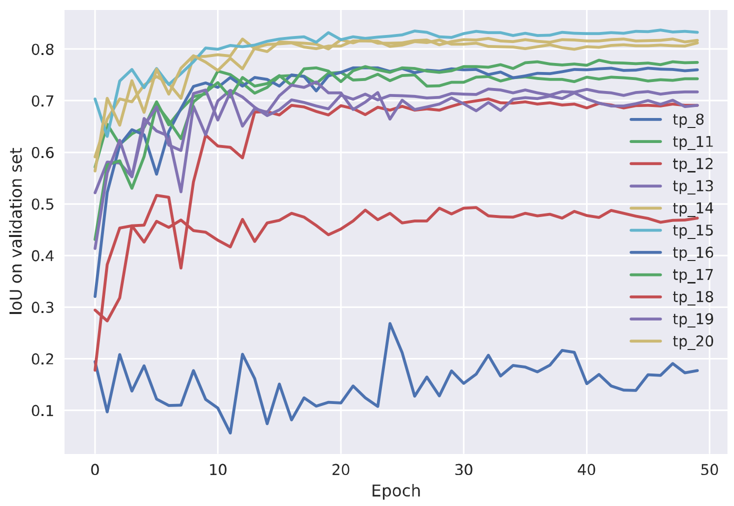

Figure 4.4: Results of Leave-One-Out Cross Validation.

4.2.1 Uncertainty metrics

Two uncertainty metrics are visualized in figures 4.5 and 4.6.

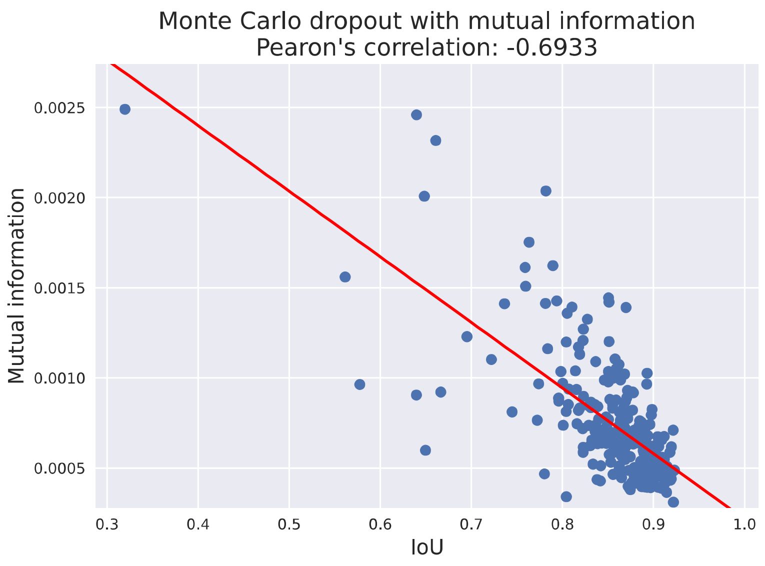

Figure 4.5: Model confidence using Monte Carlo dropout and mutual information with the corresponding IoU for each data point in the training set. A regression line is fitted by ordinary least squares.

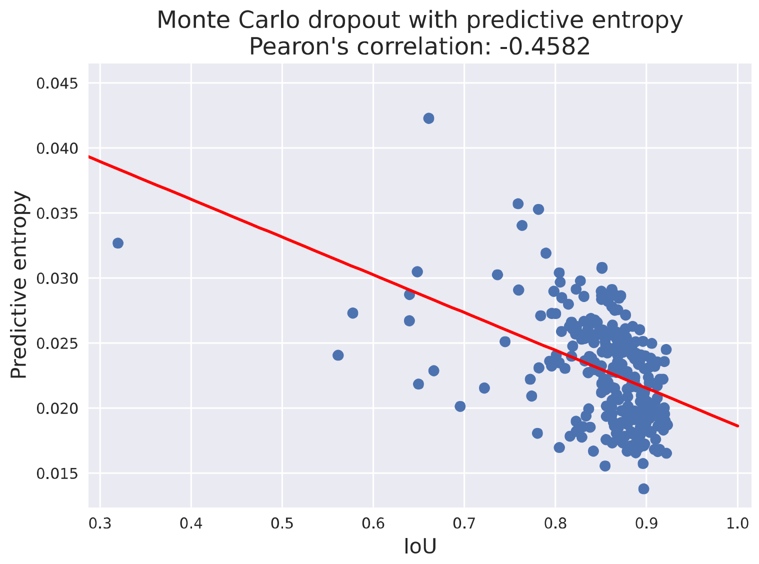

Figure 4.6: Model confidence using Monte Carlo dropout and predictive entropy with the corresponding IoU for each data point in the training set. A regression line is fitted by ordinary least squares.

Both uncertainty metrics are clearly negatively correlated with model accuracy, with Pearson correlation coefficients of -0.429 and -0.681 for predictive entropy and mutual information respectively. Thus, both predictive entropy and mutual information metrics are strong indicators of weak model performance.

The uncertainty values are rather low due to the sparsity of the model's predictions (most pixels are classified as not seatbelt), and for each image, we calculate the uncertainty as the mean uncertainty of all pixels.

Scatter plots of the other uncertainty metrics and model performance are presented in appendix C.

Two examples of uncertainty maps are visualized in figures 4.7 and 4.8. Figure 4.7 shows predictions and uncertainty maps of an image with relatively low uncertainty. In contrast, figure 4.8 shows predictions and uncertainty maps of an image with high uncertainty. The image was held out during training of the model and used for validation, according to the method detailed in section 3.5.1.

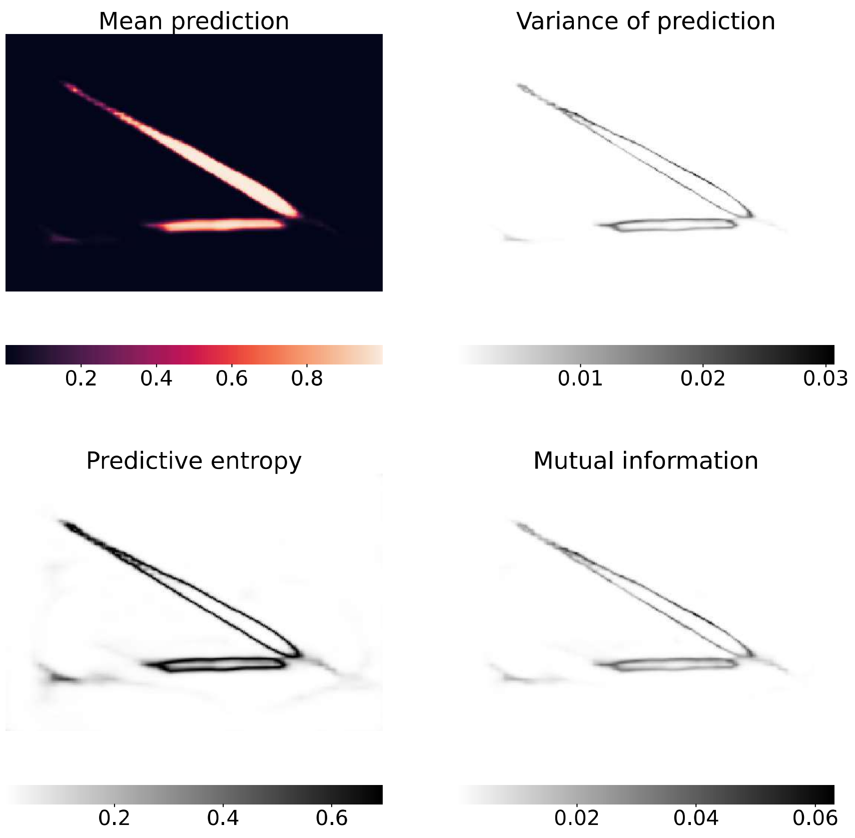

Figure 4.7: Average prediction of $T = 25$ forward passes of tp_19_1041_22.png with corresponding uncertainty metrics.

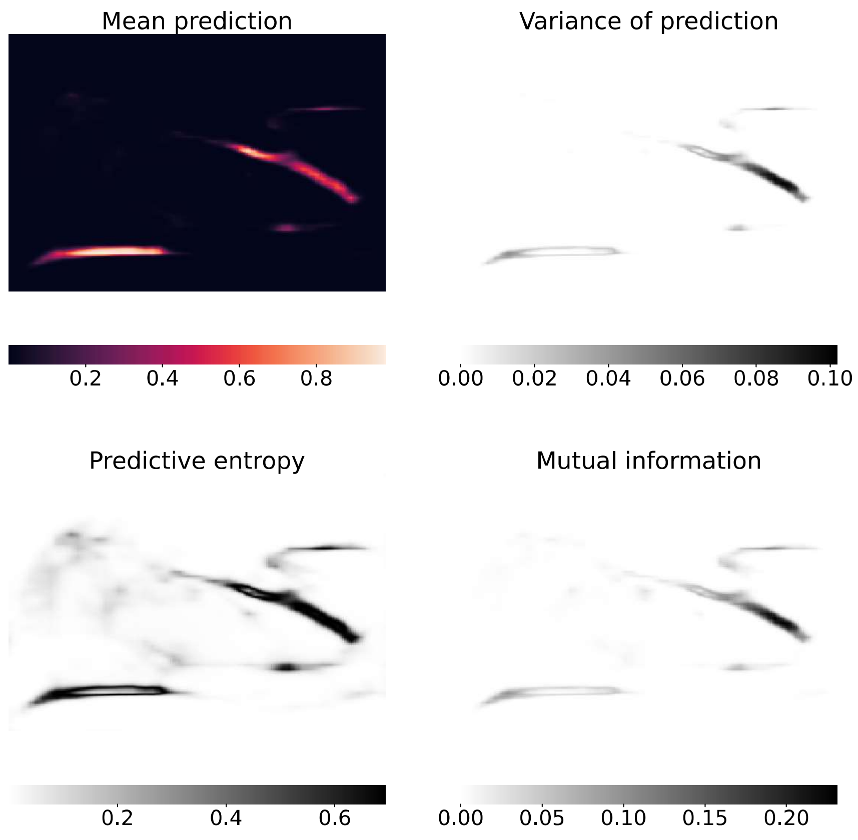

Figure 4.8: Average prediction of $T = 25$ forward passes of tp_8_1590_7.png with corresponding uncertainty metrics.

Note the sparsity in the uncertainty maps, as the vast majority of pixels have low uncertainty.

4.2.2 Active learning

The IoU scores for each querying method are shown in figure 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17 and 4.18. Each querying method is using a model identical to the baseline, a ResNet-18 with dropout in both the encoder and decoder. Comparing to this baseline let us see whether active learning provides performance benefits on top of the small performance increase attributed to the dropout model. Query by committee benchmarks are compiled over five repetitions since they took much longer than other benchmarks (16.5 hours compared to 3.5 hours). The baseline and non-query by committee benchmarks are compiled over ten repetitions. Each plot shows the mean test IoU as well as the measured IoU in a five to 95 percentile range over all repetitions. The performance of each querying method, specifically the proportion of the dataset required to reach 95% and 99% performance of a model trained on the full dataset, is collected in table 4.2.

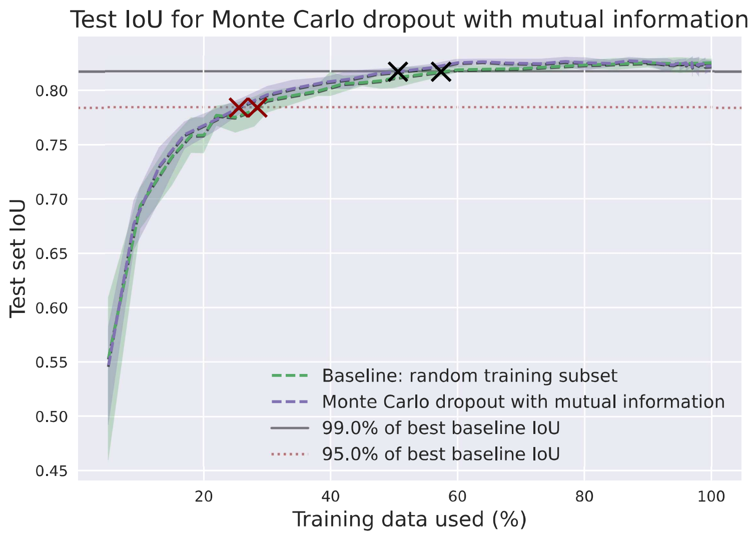

Figure 4.9: Plot of IoU score for active learning using Monte Carlo dropout with mutual information as querying method and the baseline using random querying.

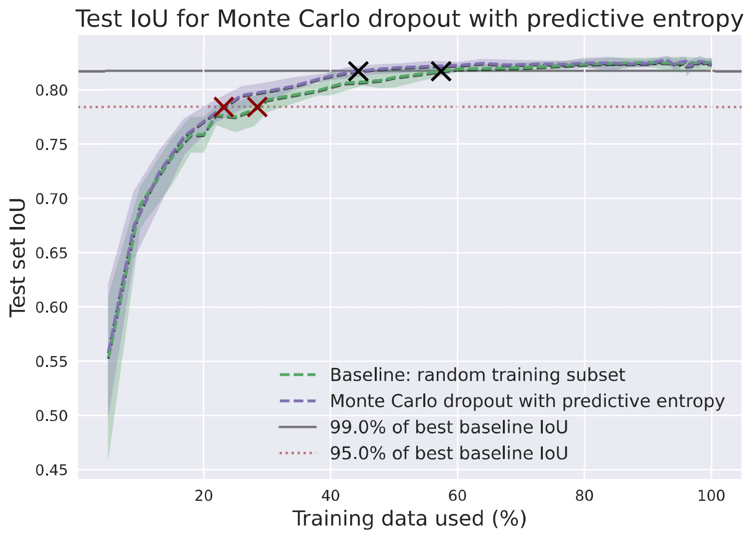

Figure 4.10: Plot of IoU score for active learning using Monte Carlo dropout with predictive entropy as querying method and the baseline using random querying.

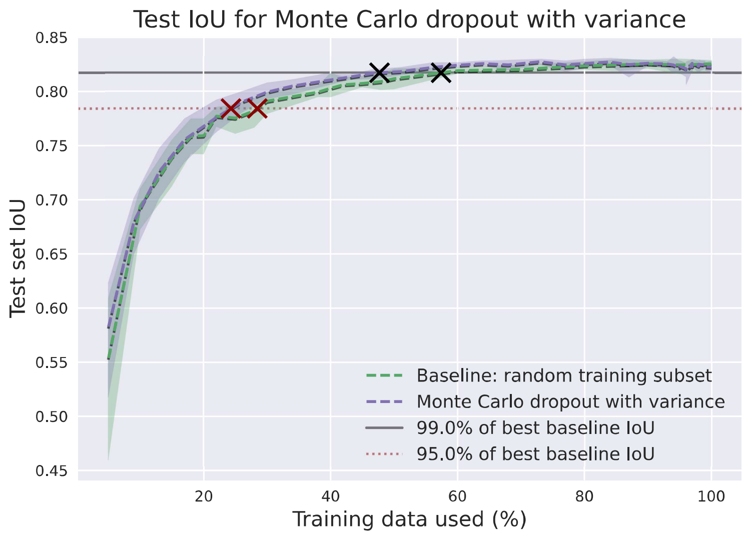

Figure 4.11: Plot of IoU score for active learning using Monte Carlo dropout and variance as querying method and the baseline using random querying.

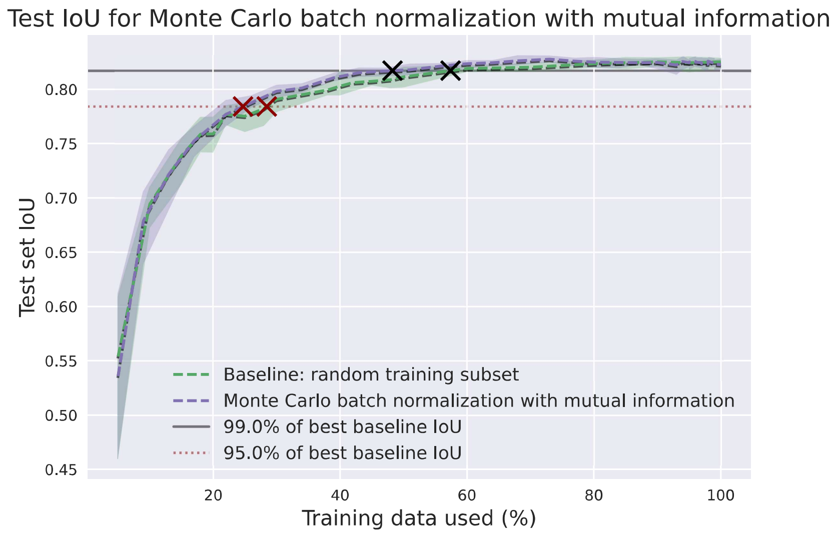

Figure 4.12: IoU score for active learning using Monte Carlo batch normalization with mutual information as querying method and the baseline using random querying.

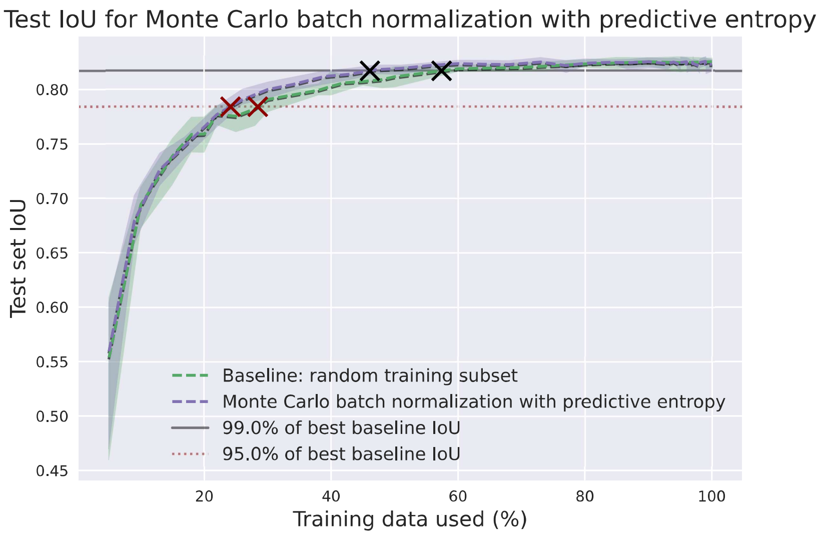

Figure 4.13: IoU score for active learning using Monte Carlo batch normalization with predictive entropy as querying method and the baseline using random querying.

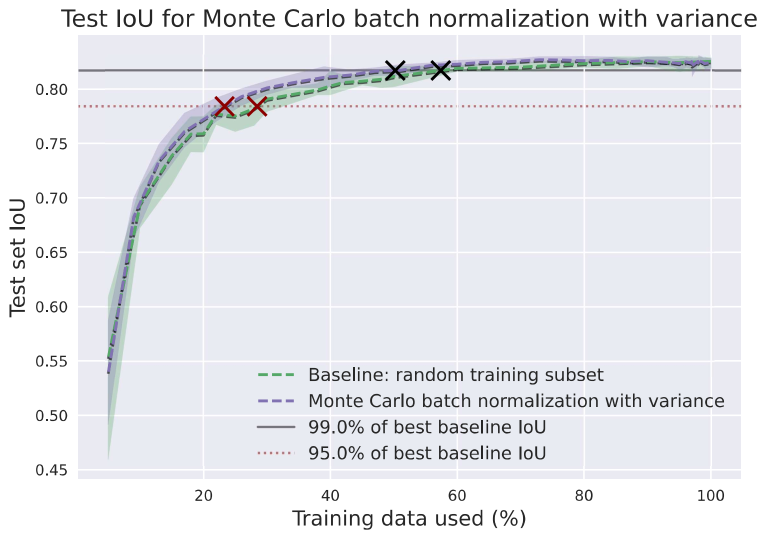

Figure 4.14: Plot of IoU score for active learning using Monte Carlo batch normalization and variance as querying method and the baseline using random querying.

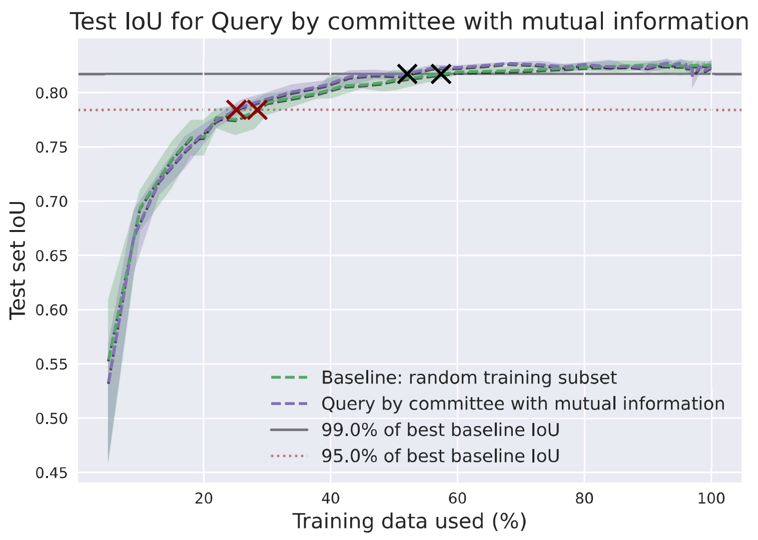

Figure 4.15: Plot of IoU score for active learning using Query by committee with mutual information as querying method and the baseline using random querying.

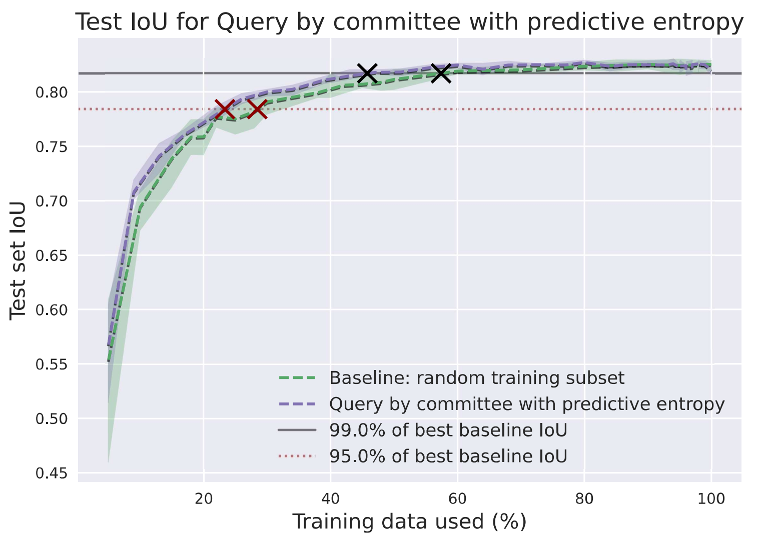

Figure 4.16: Plot of IoU score for active learning using Query by committee with predictive entropy as querying method and the baseline using random querying.

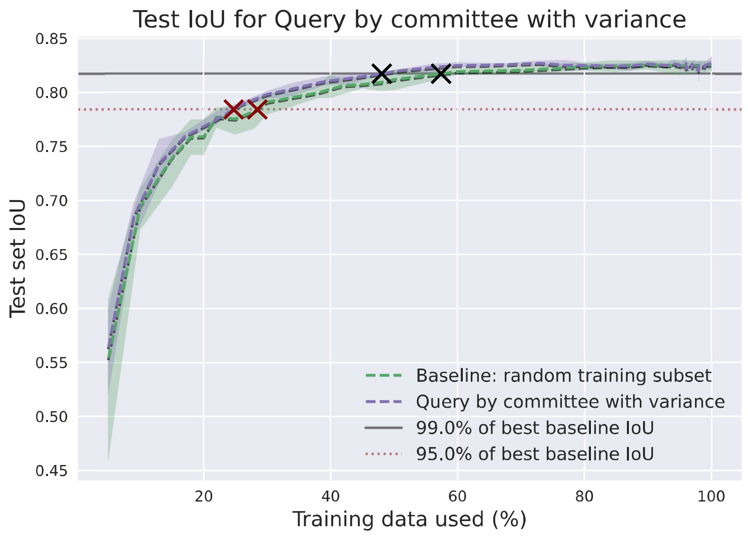

Figure 4.17: Plot of IoU score for active learning using Query by committee with variance as querying method and the baseline using random querying.

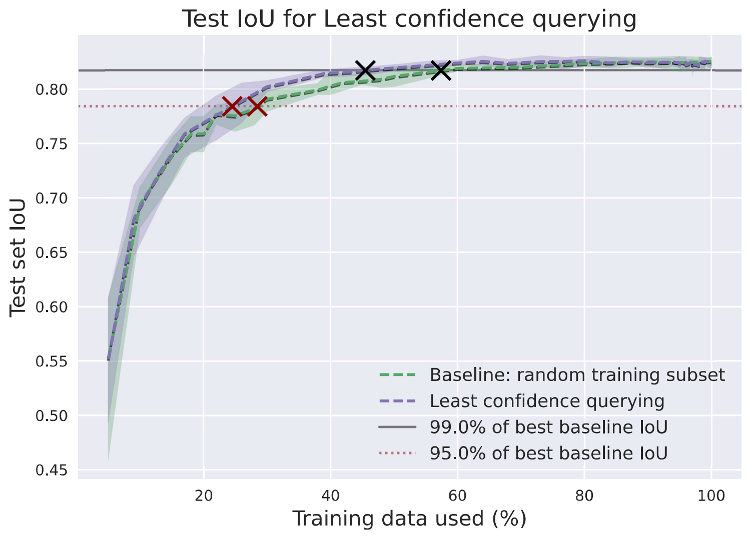

Figure 4.18: Plot of IoU score for active learning using least confidence querying and the baseline using random querying.

Query method	Uncertainty metric	Data required to reach 95% of best baseline IoU	Data required to reach 99% of best baseline IoU
Random	-	28.45%	57.40%
MCDO	Mutual information	25.54%	50.61%
	Predictive entropy	**23.17%**	**44.37%**
	Variance	24.29%	47.70%
MCBN	Mutual information	24.69%	48.27%
	Predictive entropy	24.15%	**46.10%**
	Variance	**23.32%**	50.24%
QBC	Mutual information	25.16%	52.09%
	Predictive entropy	**23.33%**	**45.79%**
	Variance	24.71%	48.05%
LC	-	24.49%	45.48

Table 4.2: Actual performance of active learning querying methods compared to a baseline using randomly selected training sets. Actual performance is quantified as the amount of training data needed for achieving 95% and 99% of the best baseline IoU. We highlight the most data efficient querying methods for each IoU threshold, for each type of querying method.

In each active learning approach, as 100% training set size was reached, the IoU converged with the IoU of the baseline. Convergence is expected since the training set was identical for the active learning approach and the baseline when the entire set was used. However, in the earlier stages of the benchmarks, each active learning approach had higher IoU than the baseline.

Every querying method reached 95% and 99% of the best baseline IoU using less training data than the baseline. Monte Carlo dropout with predictive entropy required the least data for matching 95% and 99% of the best baseline IoU, 19% and 23% less data respectively.

While all active learning querying methods had better performance than the baseline, negative correlation between uncertainty and model performance did not indicate better results in active learning. For instance, Monte Carlo dropout with mutual information uncertainty had a larger negative correlation with model performance than Monte Carlo dropout with predictive entropy, but did not showcase better performance in terms of training data needed to reach the IoU thresholds. Hence, ranking querying methods for active learning based on negative correlation with model performance is not reliable. Furthermore, to estimate expected performance using Pearson's correlation values you need to have at least some labeled data points for measuring IoU.

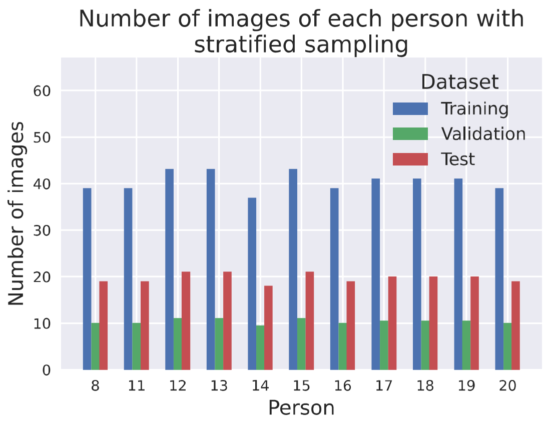

Figure 4.19: Number of images of each person after stratifying the training and validation set to match the distribution of the test set.

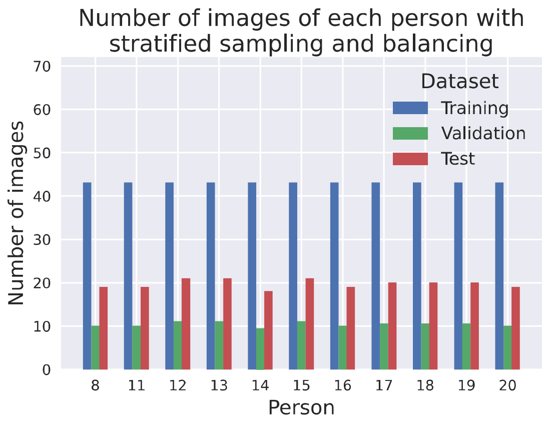

Figure 4.20: Number of images of each person after balancing the training set and stratifying the validation set.

4.2.3 Performance impact of stratified sampling and data balancing

Models were trained with two training and validation set configurations. In one benchmark the training and validation data was stratified according to the distribution of the test set, as shown in figure 4.19. In another benchmark the training set was balanced while keeping the validation set stratified, as shown in figure 4.20.

We present test IoU scores from 40 training repetitions for each dataset configuration in figure 4.21. The median and mean test IoU scores were slightly better with the modified datasets compared to the baseline. However, the measured ranges of test IoU scores were large and overlapped with the baseline to a great degree.

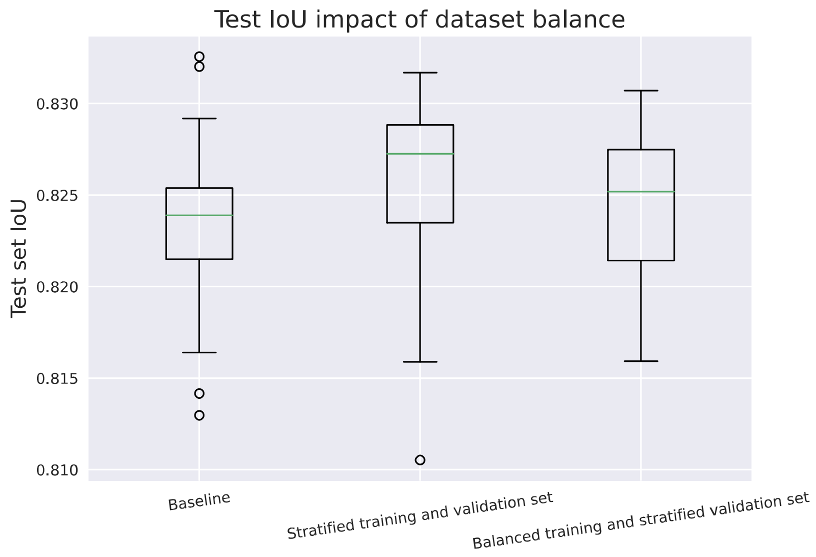

Figure 4.21: Test IoU score for a model trained with datasets modified with balancing and stratified sampling compared to a baseline model trained with an unmodified dataset.

4.2.4 Performance impact of removing poorly labeled data points

The training set was inspected manually in order to find poorly annotated data points. 20 images were identified that either had annotations deviating from the intended annotation to a great degree or that were duplicates of other images. An image with poor annotation is shown in figure 4.22 and all removed images are shown in Appendix D.

The test set IoU scores for models trained with the original dataset and the modified dataset in 40 repetitions are shown in figure 4.23. The mean and median test IoU scores were slightly better than for the baseline. However, the measured ranges of test IoU scores were large and overlapped with the baseline to a great degree.

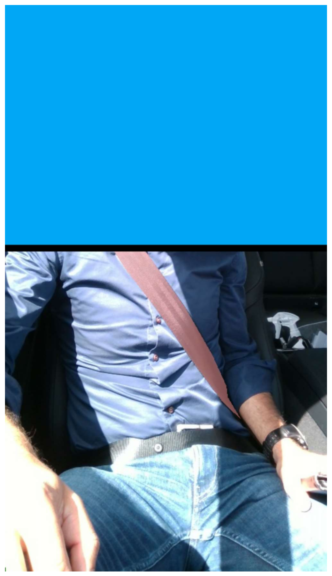

Figure 4.22: A data point with poor annotation that was removed from the training set. The lower part of the belt is not annotated.

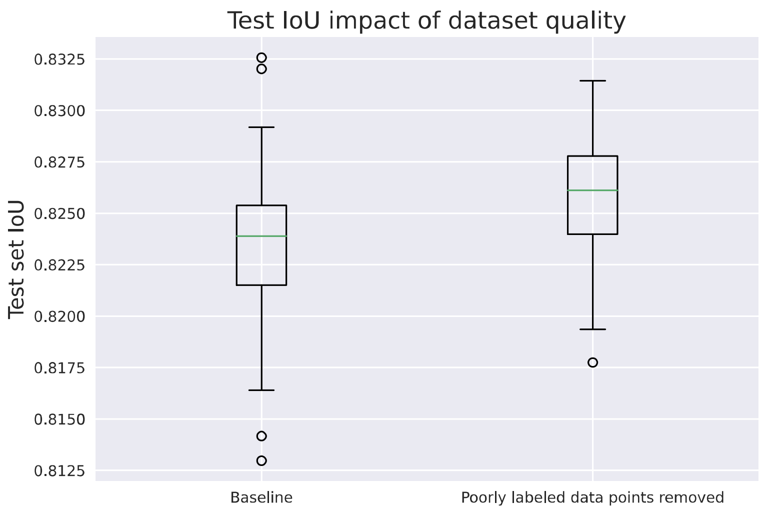

Figure 4.23: Test IoU score for a model trained on a dataset with poorly labeled data points removed and a baseline model trained with an unmodified dataset.

5

Discussion

5.1 Model-centric approaches

The simpler ResNet-18 network performed the best and was also trained fastest, with four times less trainable parameters than e.g. ResNet-101. It is unclear why larger networks with more trainable parameters fail to perform better. One reason could be that ResNet networks were originally designed for image classification with hundreds of different output classes, whereas we adapt the network for binary classification of each input pixel.

More model architectures should be benchmarked in future work to see if any significant performance gains can be seen. Altering network architecture is relatively easy compared to conducting data-centric approaches, hence further investigation is needed. More thorough hyperparameter tuning and optimizer selection should be done as well.

5.2 Data-centric approaches

All our benchmarks for data-centric approaches can both be improved and expanded upon.

5.2.1 Leave-one-out cross-validation

The results of the leave-one-out cross-validation procedure showed that the model performs well on unseen images of test subjects that are present in the training set, but the model does not always perform well on the held-out test subject in the validation set.

Thus, in order to increase model generalization, it would be wiser to increase dataset diversity by collecting images of a larger number of test subjects, rather than collecting more images of each individual test subject.

5.2.2 Active learning improvements

All querying methods and uncertainty metrics were shown to be negatively correlated with the model's accuracy, but the sample size is small and the dataset is not particularly diverse, with only 11 distinct test subjects.

The results establish the uncertainty metrics presented in 2.6.3 as suitable proxies for model performance in the unsupervised setting. Given a set of unlabeled images, the uncertainty metrics can be used to determine the images on which the model performs badly, thus indicating that these images should be prioritized for labeling.

It remains to be seen whether data points where the model is uncertain are informative, that is whether adding these data points to the training set provides the model with valuable information. The active learning results show that providing the model with the most uncertain examples first increases performance. However, the improvement is fairly small considering the effort required to conduct active learning, likely due to the limited size of the dataset.

The insufficient dataset size is troublesome when evaluating the approach used for active learning, as the search space for the query strategy is rather small. We hypothesize that given a larger dataset, the proposed query strategy would significantly outperform random sampling. Therefore, we suggest that a larger and more dataset be used in further work, in order to evaluate the active learning query strategy. An interesting investigation for future work is to compare performance improvements that follow from collecting more high quality data compared to cheaper data. Data can be collected at a low cost using scribble annotations, which could then be used to train a weakly supervised model.

To be able to make a more conclusive comparison between the different querying methods we should also perform more repetitions of each active learning benchmark. If we were to repeat each benchmark more than ten times we could increase confidence in the mean IoU scores for each benchmark, and thus more confidently rank each querying method. However, each benchmark took between three and sixteen hours, entailing high costs for computational power.

Another possible shortcoming of our comparisons between active learning and the baseline is that selecting data points randomly for the baseline is arguably not realistic and could potentially entail lower IoU score than a more realistic selection method. In reality the entire training set was chosen using PCA and k-means clustering, so rather than randomly selecting data points from this original data when constructing our smaller training sets for our benchmarks we should redo the PCA and k-means clustering to create each of the smaller training sets. To mimic the realistic selection for smaller datasets we would repeat the PCA identically, but reduce the number of centroids for the k-means clustering to equal the number of images desired of each test person in order to add up to the intended training set size. This approach could improve the baseline IoU score, because the motivation to use PCA and k-means was to be able to select one image from each k-means cluster, since that would increase the diversity in the dataset. With our random selection from a larger number of centroids we do not maximize this diversity.

Despite the method used to randomly construct the training set for our baseline, we believe that our benchmarks are valid for two reasons. Firstly, it is possible that the diversity achieved with random sampling would not greatly differ from that of k-means clustering selection. With random sampling it is always possible to select the most diverse data or close to the most diverse data. Furthermore, for the k-means clustering selection used originally, the diversity was not maximized. One data point was simply selected randomly for each centroid, which is not generally optimal if we assume the diversity is measured as the sum of distances between the data points selected. The second reason why we believe our benchmarks are valid is that our baseline could actually be considered realistic in two scenarios. The more conservative scenario is a hypothetical project in which PCA and k-means clustering is still used to collect training data the same way as discussed before. Even if this is the case, one might not be able to determine beforehand exactly how much data is needed throughout said project, meaning that a very large training set might be chosen initially. This entire dataset might be selected using k-means but throughout the project there might be a need to randomly select a subset of the data to annotate, if there are not enough resources to annotate all the data at once. This scenario reflects our baseline. The second less conservative scenario is one in which all the data is selected randomly. If we consider a baseline reflecting this construction one can expect it to perform even worse than our baseline which at least used PCA and k-means to select a few relatively diverse candidates to query from.

Another issue with our active learning implementation was that in the first few iterations the model was likely not trained well enough to query the most informative data points since it was only given 5% of the training data for the first iteration. To ensure the model can query informative data points in the early iterations we could pre-train the model with cheaply obtainable unlabeled images from our domain. However, to make a fair evaluation of active learning leveraging pre-training in our domain we should compare it to a baseline leveraging the same type of pre-training. Pre-training should not only help predict which informative data points should be queried but should also improve the model's IoU score.

Another future investigation on active learning could focus more on execution time as primary evaluation between querying methods. While Monte Carlo dropout with predictive entropy performed the best in terms of data efficiency out of the benchmarked active learning approaches, it might not always be the fastest. The execution time consists of two parts, model training and uncertainty estimation. Training is five times slower for QBC compared to the other methods, since five models are trained in parallel. For Monte Carlo methods, the uncertainty estimation is five times slower than for QBC, since 25 forward passes through the network were chosen for each image, compared to five for QBC. For least confidence querying we only perform one forward pass with a single model. This means that if a much larger unlabeled data pool is used in practice, increasing the time needed for prediction, it might be faster to use QBC over Monte Carlo methods, and least confidence querying will always be the fastest. To get a better understanding of which query method is more beneficial in different scenarios we propose two types of benchmarks for future work. The first type of benchmark to investigate is how the number of predictions done in each query method affects performance. One could benchmark active learning performance for e.g. 5, 10, 25 and 50 predictions. A lot of time can be saved if we can reduce the number of predictions used as a result of knowing how many predictions result in diminishing returns in terms of IoU during active learning. A second investigation could compare query methods for a time constrained scenario. Monte Carlo methods and QBC could be configured with a number of predictions resulting in each method requiring the same execution time. These benchmarks could be repeated for multiple data pool sizes to see which method is the most efficient for a given time constraint and data pool size.

There are more advanced querying methods from previous work we could investigate for our domain, including representation querying and pixel weighing. Representation querying could be factored into any of our current querying methods to not only select based on which data point is informative, but also if that data point is representative for many other data points and hence expected to improve overall model IoU score significantly. Pixel weighing could also be combined with any of our current approaches. The uncertainty of each pixel could be weighed based on how far it is from the contour of the seat belt. Each pixel could also be weighed more based on how certain the pixel is expected to be. The expected certainty can be predicted using a CNN that is trained with the same input as our main network but with the labeling of the data being the certainty provided by our main network.

Active learning can be a valuable asset for seat belt segmentation since it allows domain experts to spend less time annotating. However, a convenient annotation pipeline is needed to save time in practice. With a naive annotation pipeline there may be overhead in terms of operation time since users can only annotate a few data points at once before having to wait for the model to re-train and query data points for labeling in the next iteration. We suggest two ways to avoid this overhead. The first suggestion is to implement an annotation pipeline that allows users to easily exit the annotation process while the model is re-training and then resume the annotation task in the next iteration when training is done. This would allow users to work on other tasks while the model is training. However, if training is fast, this pipeline will result in users switching tasks often and may inhibit their ability to focus. A different approach is to let users label data points while the model is re-training in each iteration. To implement this pipeline we need to query the user a set of data points that is split into two subsets. The first subset is annotated by the user and given to the model for re-training like normally. The second subset is annotated by the user during the training process, preemptively expanding the training data set for the next iteration. A downside with this approach is that the data points that are queried preemptively might not be useful once the next iteration is reached. To avoid annotating many data points in vain, this pipeline should not be used for models that train slowly, since this would mean there would be a lot of time to label many data points preemptively.

5.2.3 Improving dataset balance and quality

Modifying our dataset using stratified sampling, balancing and by removing poorly labeled data points did not improve performance significantly. Furthermore, since the variances in test IoU were so high, we can not confidently say that our approaches will reliably outperform the baseline. The reason could be due to the way oversampling was used. When we oversampled, we used the same augmentations as for the original dataset, which did not alter each image by a lot. This could mean that the images that were duplicated contributed a lot to how the model parameters converged, meaning we skewed the model to make predictions specialized on the duplicated data points. This would mean that oversampling had the opposite effect to what we were trying to achieve with balancing and stratified sampling, where we tried to prevent the model from skewing towards certain test subjects in the training data.

The initial data imbalance and label quality issues were not severe, which might have been a reason why we did not see a significant impact on performance after stratifying, balancing and reducing label errors. To get a better indication of the importance of these modifications one might need to compare to a baseline with more severe data balance and quality issues. This could be done artificially by randomly introducing label errors and skewing the data.

Our dataset quality improvements might not have been good enough to increase model performance significantly. Firstly, rather than simply duplicating images to balance data, we should use augmentations that are strong enough to differentiate the copies from the originals, but also not strong enough to produce images that are unrealistic for our domain. Secondly, rather than removing poorly labeled data points, one could fix the labels.

6 Conclusion

We did not improve model performance using model-centric approaches, but training time was reduced by four times without impacting performance. Simple data-centric approaches, i.e. modifying the training set and validation set using stratified sampling and balancing, as well as error reduction through deletion of poorly labeled data points, did not improve performance significantly. These approaches are however cheap, so more thorough investigations are needed to show their value. Active learning did however show improvements over the baseline.

Active learning was evaluated by querying data points from the previously labeled training set and comparing to a baseline using random subsets of the training set in order to see how much data would be needed to reach equivalent performance. The investigated active learning querying methods include least confidence querying and every combination of the prediction methods Monte Carlo dropout, Monte Carlo batch normalization and query by committee paired with the uncertainty metrics mutual information, predictive entropy and variance. Monte Carlo dropout with predictive entropy was shown to reach baseline equivalent performance with the least data. With Monte Carlo dropout and predictive entropy we achieved 95% and 99% of the best baseline IoU using 19% and 23% less data respectively.

Future improvements to our active learning approach include repeating benchmarks more times to improve our confidence and comparing active learning performance to a baseline using a more realistic but cheap training set construction method. Future expansions to our active learning include more advanced active learning querying methods that weigh data points based on if they are representative for many other data points. Weighing can also be done pixel-wise for disagreement estimation of data based on if specific regions in the data are expected to have high disagreement. Models can also be pre-trained without labeled data to improve the initial phase of active learning algorithms. Another way to leverage data cheaply is using weakly supervised models utilizing point or scribble annotations. This could allow more test persons in the dataset and increase diversity and model generalization.

Bibliography

[1] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," 2019.

[2] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 12 2016.

[3] "A visual equation for intersection over union (jaccard index)," https://commons.wikimedia.org/wiki/File:Intersection_over_Union_-_visual_equation.png, accessed: 2022-04-13.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[5] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

[8] D. Radečić, "Data-centric vs. model-centric ai? the answer is clear," Aug 2021, accessed: 2022-06-09. [Online]. Available: https://towardsdatascience.com/data-centric-vs-model-centric-ai-the-answer-is-clear-4b607c58af67

[9] (2021, Mar) a chat with andrew on mlops: from model-centric to data-cetric ai. Accessed: 2022-06-09.

[10] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," 2017.

[11] A. Zlateski, R. Jaroensri, P. Sharma, and F. Durand, "On the importance of label quality for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1479–1487.

[12] H. Hao, S. Didari, J. O. Woo, H. Moon, and P. Bangert, "Highly efficient representation and active learning framework for imbalanced data and its application to covid-19 x-ray classification," *arXiv preprint arXiv:2103.05109*, 2021.

[13] J. Xiong, "Active learning for semantic segmentation," 2019.

[14] J. Zou and L. Schiebinger, "Ai can be sexist and racist — it's time to make it fair," *Nature*, vol. 559, no. 7714, pp. 324–326, 2018.

[15] J. Buolamwini and T. Gebru, "Gender shades: Intersectional accuracy disparities in commercial gender classification," in *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, ser. Proceedings of Machine Learning Research, S. A. Friedler and C. Wilson, Eds., vol. 81. PMLR, 23–24 Feb 2018, pp. 77–91. [Online]. Available: https://proceedings.mlr.press/v81/buolamwini18a.html

[16] W. G. COCHRAN, "Sampling techniques," 1977.

[17] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[18] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, 2018.

[19] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[20] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, vol. 2, no. 11, pp. 559–572, 1901.

[21] J. A. Hartigan, *Clustering algorithms.* John Wiley & Sons, Inc., 1975.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[23] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

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

[25] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," 2015.

[26] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," 2018. [Online]. Available: https://arxiv.org/abs/1802.02611

[27] "Resnet," https://pytorch.org/hub/pytorch_vision_resnet/, accessed: 2022-05-11.

[28] M. Boguslav and K. B. Cohen, "Inter-nnotator agreement and the upper limit on machine performance: Evidence from biomedical natural language processing," *Studies in health technology and informatics*, vol. 245, pp. 298–302, 2017.

[29] A. Ratner, C. D. Sa, S. Wu, D. Selsam, and C. Ré, "Data programming: Creating large training sets, quickly," 2017.

[30] A. Jeddi, "Tackling the problem of limited data and annotations in semantic segmentation," *CoRR*, vol. abs/2007.07357, 2020. [Online]. Available: https://arxiv.org/abs/2007.07357

[31] H. Lin, P. Upchurch, and K. Bala, "Block annotation: Better image annotation for semantic segmentation with sub-image decomposition," 2020.

[32] C. Mayer, R. Timofte, and G. Paul, “Towards closing the gap in weakly supervised semantic segmentation with dcnns: Combining local and global models,” 2019.

[33] I. Z. Yalniz, H. Jégou, K. Chen, M. Paluri, and D. Mahajan, “Billion-scale semi-supervised learning for image classification,” 2019.

[34] W.-C. Hung, Y.-H. Tsai, Y.-T. Liou, Y.-Y. Lin, and M.-H. Yang, “Adversarial learning for semi-supervised semantic segmentation,” 2018.

[35] G. Papandreou, L.-C. Chen, K. Murphy, and A. L. Yuille, “Weakly- and semi-supervised learning of a dcnn for semantic image segmentation,” 2015.

[36] L. Yang, Y. Zhang, J. Chen, S. Zhang, and D. Z. Chen, “Suggestive annotation: A deep active learning framework for biomedical image segmentation,” in *International conference on medical image computing and computer-assisted intervention*. Springer, 2017, pp. 399–407.

[37] B. Settles, M. Craven, and S. Ray, “Multiple-instance active learning,” *Advances in neural information processing systems*, vol. 20, 2007.

[38] A. Vezhnevets, J. M. Buhmann, and V. Ferrari, “Active learning for semantic segmentation with expected change,” in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3162–3169.

[39] N. Roy and A. McCallum, “Toward optimal active learning through monte carlo estimation of error reduction,” *ICML, Williamstown*, vol. 2, pp. 441–448, 2001.

[40] A. I. Schein and L. H. Ungar, “Active learning for logistic regression: an evaluation,” *Machine Learning*, vol. 68, no. 3, pp. 235–265, 2007.

[41] H. S. Seung, M. Opper, and H. Sompolinsky, “Query by committee,” in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 287–294.

[42] B. Settles, “Active learning literature survey,” 2009.

[43] Y. Gal, “Uncertainty in deep learning,” 2016.

[44] A. Mehrtash, W. M. Wells, C. M. Tempany, P. Abolmaesumi, and T. Kapur, “Confidence calibration and predictive uncertainty estimation for deep medical image segmentation,” *IEEE transactions on medical imaging*, vol. 39, no. 12, pp. 3868–3878, 2020.

[45] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” 2015. [Online]. Available: https://arxiv.org/abs/1506.02142

[46] J. Brooks, “COCO Annotator,” https://github.com/jsbroks/coco-annotator/, 2019.

[47] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, “Albumentations: fast and flexible image augmentations,” *Information*, vol. 11, no. 2, p. 125, 2020.

A
Appendix A

Listing A.1: PyTorch image segmentation model.

```
==================================================================================================================
Layer (type:depth-idx)                                  Output Shape                   Param #
==================================================================================================================
FPN                                                     [4, 1, 224, 224]               --
├─ResNetEncoder: 1-1                                    [4, 3, 224, 224]               --
|    ├─Conv2d: 2-1                                      [4, 64, 112, 112]              9,408
|    ├─BatchNorm2d: 2-2                                 [4, 64, 112, 112]              128
|    ├─ReLU: 2-3                                        [4, 64, 112, 112]              --
|    ├─MaxPool2d: 2-4                                   [4, 64, 56, 56]                --
|    ├─Sequential: 2-5                                  [4, 64, 56, 56]                --
|    |    ├─BasicBlock: 3-1                             [4, 64, 56, 56]                73,984
|    |    ├─BasicBlock: 3-2                             [4, 64, 56, 56]                73,984
|    ├─Dropout2d: 2-6                                   [4, 64, 56, 56]                --
|    ├─Sequential: 2-7                                  [4, 128, 28, 28]               --
|    |    ├─BasicBlock: 3-3                             [4, 128, 28, 28]               230,144
|    |    ├─BasicBlock: 3-4                             [4, 128, 28, 28]               295,424
|    ├─Dropout2d: 2-8                                   [4, 128, 28, 28]               --
|    ├─Sequential: 2-9                                  [4, 256, 14, 14]               --
|    |    ├─BasicBlock: 3-5                             [4, 256, 14, 14]               919,040
|    |    ├─BasicBlock: 3-6                             [4, 256, 14, 14]               1,180,672
|    ├─Dropout2d: 2-10                                  [4, 256, 14, 14]               --
|    ├─Sequential: 2-11                                 [4, 512, 7, 7]                 --
|    |    ├─BasicBlock: 3-7                             [4, 512, 7, 7]                 3,673,088
|    |    ├─BasicBlock: 3-8                             [4, 512, 7, 7]                 4,720,640
├─FPNDecoder: 1-2                                       [4, 128, 56, 56]               --
|    ├─ModuleList: 2-12                                 --                             1,623,808
|    ├─Conv2d: 2-13                                     [4, 256, 7, 7]                 131,328
|    ├─FPNBlock: 2-14                                   [4, 256, 14, 14]               --
|    |    ├─Conv2d: 3-9                                 [4, 256, 14, 14]               65,792
|    |    ├─Dropout2d: 3-10                             [4, 256, 14, 14]               --
|    ├─FPNBlock: 2-15                                   [4, 256, 28, 28]               --
|    |    ├─Conv2d: 3-11                                [4, 256, 28, 28]               33,024
|    |    ├─Dropout2d: 3-12                             [4, 256, 28, 28]               --
|    ├─FPNBlock: 2-16                                   [4, 256, 56, 56]               --
|    |    ├─Conv2d: 3-13                                [4, 256, 56, 56]               16,640
|    |    ├─Dropout2d: 3-14                             [4, 256, 56, 56]               --
|    ├─ModuleList: 2-12                                 --                             1,623,808
|    |    ├─SegmentationBlock: 3-15                     [4, 128, 56, 56]               590,592
|    |    ├─SegmentationBlock: 3-16                     [4, 128, 56, 56]               442,880
|    |    ├─SegmentationBlock: 3-17                     [4, 128, 56, 56]               295,168
|    |    ├─SegmentationBlock: 3-18                     [4, 128, 56, 56]               295,168
|    ├─MergeBlock: 2-17                                 [4, 128, 56, 56]               --
├─SegmentationHead: 1-3                                 [4, 1, 224, 224]               --
|    ├─Conv2d: 2-18                                     [4, 1, 56, 56]                 129
|    ├─UpsamplingBilinear2d: 2-19                       [4, 1, 224, 224]               --
|    ├─Activation: 2-20                                 [4, 1, 224, 224]               --
|    |    ├─Identity: 3-19                              [4, 1, 224, 224]               --
==================================================================================================================
Total params: 13,047,233
Trainable params: 13,047,233
Non-trainable params: 0
Total mult-adds (G): 13.60
==================================================================================================================
Input size (MB): 2.41
Forward/backward pass size (MB): 241.75
Params size (MB): 52.19
Estimated Total Size (MB): 296.35
==================================================================================================================
```

B
Appendix B

Listing B.1: Example COCO annotation file for semantic segmentation.

```
{"images": [
        {
            "id": 1148,
            "dataset_id": 16,
            "category_ids": [
                1,
                2
            ],
            "path": "/datasets/sapa-training/tp_16_3437_3
              437.png",
            "width": 480,
            "height": 848,
            "file_name": "tp_16_3437_3437.png",
            "annotated": true,
            "annotating": [],
            "num_annotations": 2,
            "metadata": {},
            "milliseconds": 19812,
            "events": [
                {
                    "_cls": "SessionEvent",
                    "user": "redacted",
                    "milliseconds": 19812,
                    "tools_used": []
                }
            ],
            "regenerate_thumbnail": false,
            "is_modified": false
        }
    ],
"categories": [
        {
            "id": 2,
            "name": "seat_belt",
            "supercategory": "",
            "color": "#70f404",
            "metadata": {},
            "creator": "system",
            "keypoint_colors": [],
            "keypoints": [
                "shoulder_belt",
                "mid_belt",
                "lower_belt"
            ],
            "skeleton": [
                [
                    1,
                    2
                ],
                [
                    2,
                    3
                ]
            ]
        }
    ],
"annotations": [
        {
            "id": 1247,
            "image_id": 1148,
            "category_id": 2,
            "dataset_id": 16,
            "segmentation": [
                [
                    80.2,
                    170.5,
                    87.3,
                    161.9,
                    119.5,
                    213.7,
                    142.2,
                    240.4,
                    176.8,
                    276.5,
                    208.9,
                    318.8,
                    236.4,
                    351,
                    256.8,
                    383.2,
                    316.4,
                    469.5,
                    339.2,
                    519.7,
                    413,
                    695.5,
                    405.1,
                    692.4,
                    399.5,
                    684.5,
                    407.5,
                    696.3,
                    388.6,
                    683,
                    343.9,
                    668.9,
                    317.2,
                    658.7,
                    297.6,
                    657.1,
                    172,
                    675.1,
                    87.3,
                    690.8,
                    90.4,
                    677.5,
                    132.8,
                    664.9,
                    150.1,
                    656.3,
                    169.7,
                    650,
                    250.5,
                    633.5,
                    289.8,
                    632,
                    320.4,
                    634.3,
                    338.4,
                    642.2,
                    379.2,
                    654.7,
                    384.8,
                    663,
                    357.2,
                    610,
                    330.6,
                    563.7,
                    298.4,
                    513.5,
                    267,
                    460.1,
                    185.4,
                    336.9,
                    136.7,
                    252.9,
                    121.8,
                    222.3,
                    78.7,
                    170.5
                ]
            ],
            "area": 21995,
            "bbox": [
                80,
                162,
                333,
                534
            ],
            "iscrowd": false,
            "isbbox": false,
            "creator": "system",
            "width": 480,
            "height": 848,
            "color": "#cd35f0",
            "keypoints": [
                159,
                273,
                2,
                276,
                451,
                2,
                402,
                680,
                2
            ],
            "metadata": {},
            "milliseconds": 0,
            "events": [],
            "num_keypoints": 3
        }
    ]
}
```

C

Appendix C

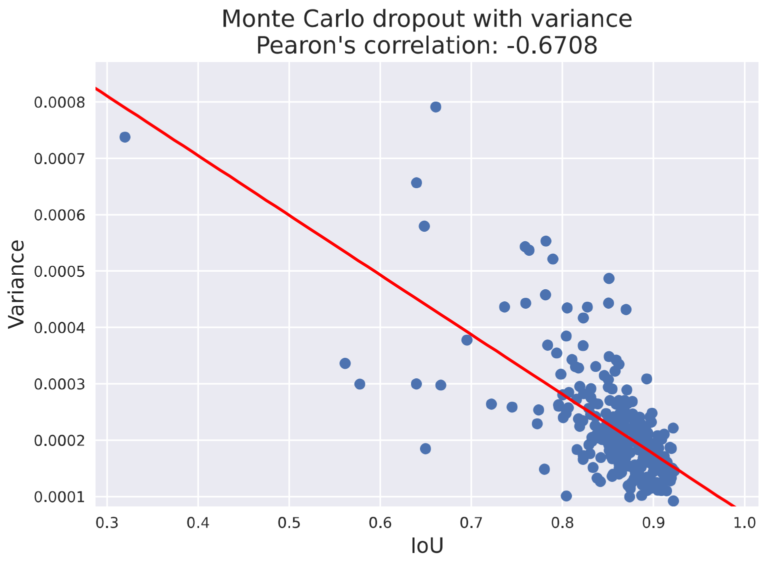

Figure C.1: Model confidence using Monte Carlo dropout and variance with the corresponding IoU for each data point. A regression line is fitted by ordinary least squares.

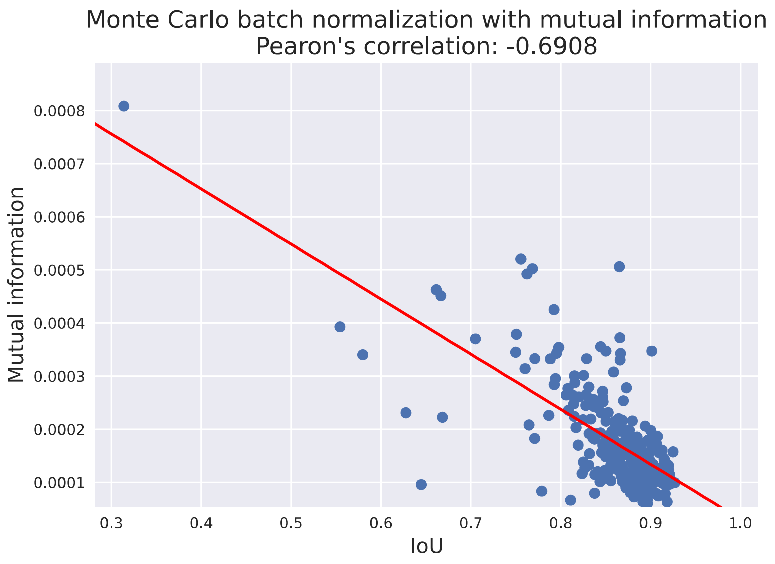

Figure C.2: Model confidence using Monte Carlo batch normalization and mutual information with the corresponding IoU for each data point. A regression line is fitted by ordinary least squares.

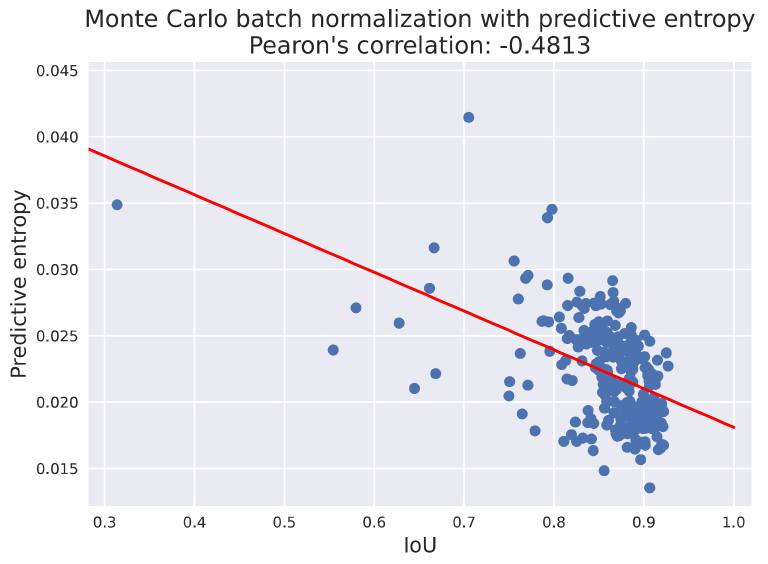

Figure C.3: Model confidence using Monte Carlo batch normalization and predictive entropy with the corresponding IoU for each data point. A regression line is fitted by ordinary least squares.

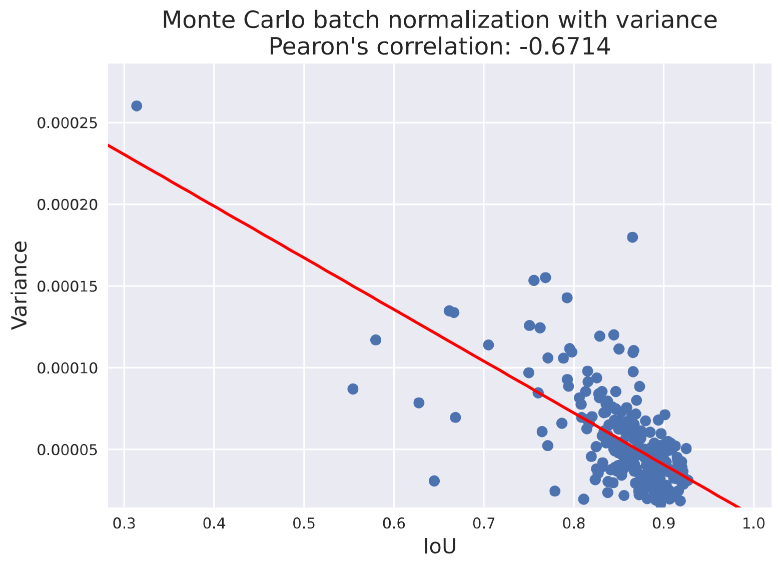

Figure C.4: Model confidence using Monte Carlo batch normalization and variance with the corresponding IoU for each data point. A regression line is fitted by ordinary least squares.

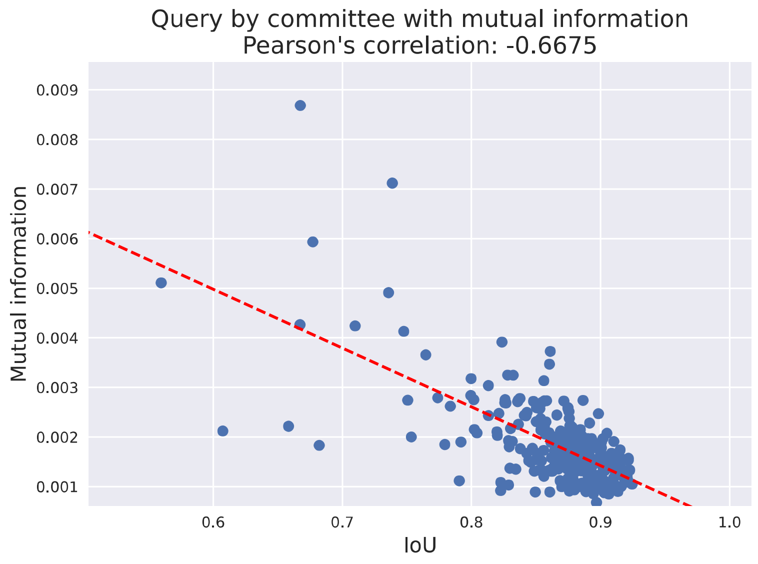

Figure C.5: Model confidence using query by committee and mutual information with the corresponding IoU for each data point. A regression line is fitted by ordinary least squares.

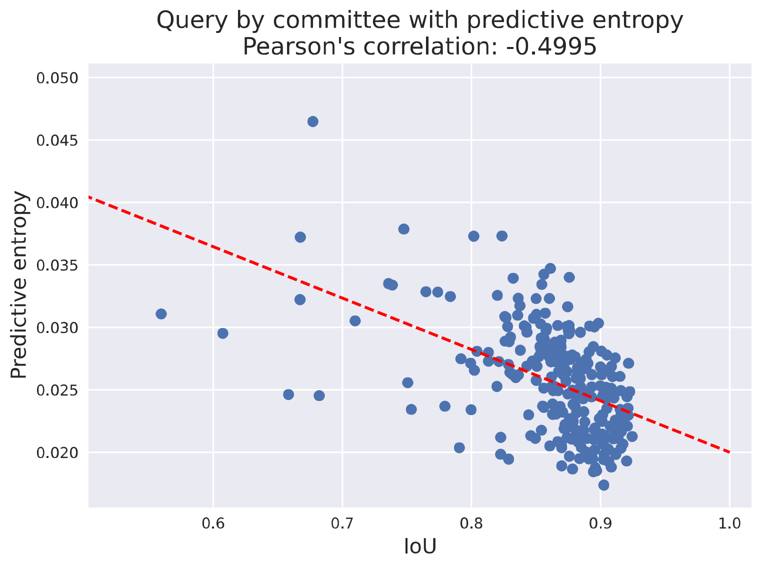

Figure C.6: Model confidence using query by committee and predictive entropy with the corresponding IoU for each data point. A regression line is fitted by ordinary least squares.

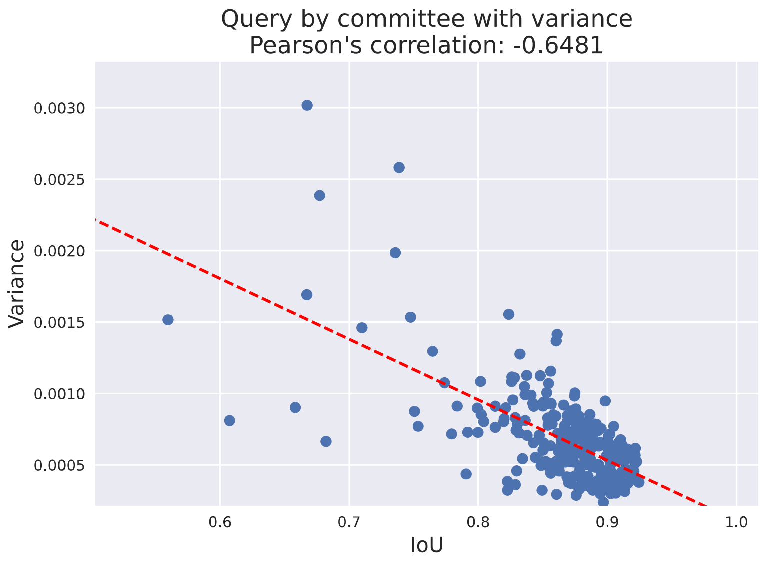

Figure C.7: Model confidence using query by committee and variance with the corresponding IoU for each data point. A regression line is fitted by ordinary least squares.

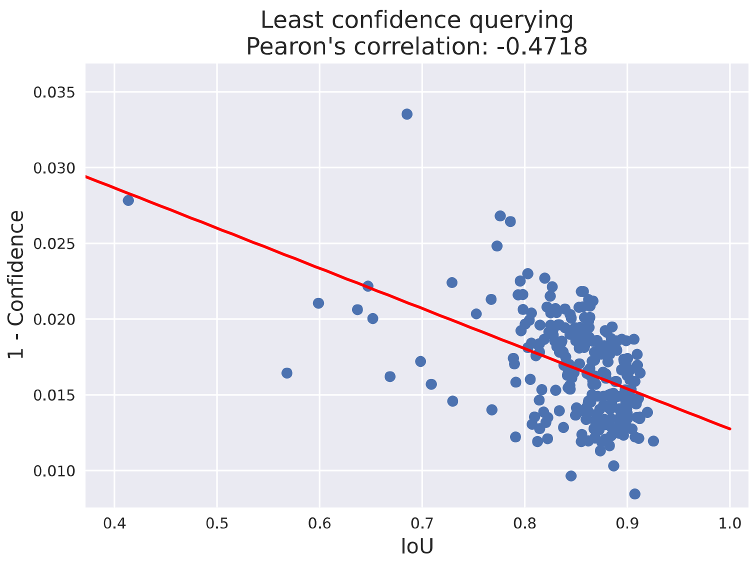

Figure C.8: Model confidence based on final layer activation and corresponding IoU for each data point. A regression line is fitted by ordinary least squares.

D

Appendix D

The figures in this appendix show images with bad annotations that were removed in one benchmark in order to investigate the importance of label quality on model performance. Some images were removed due to being a duplicate of a certain frame, rather due to the quality of the label.

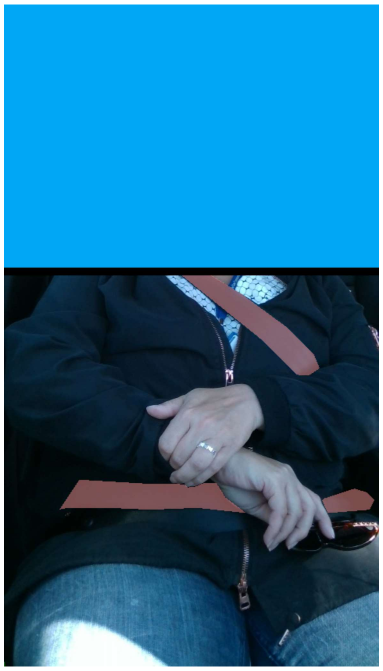

Figure D.1: An image where the lower part of the belt is annotated too high up.

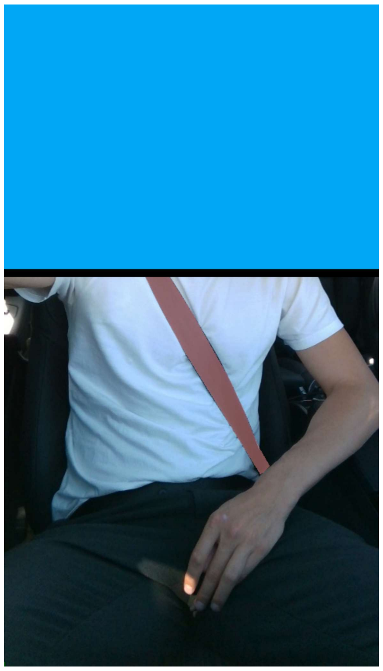

Figure D.2: An image where the lower part of the annotation is missing.

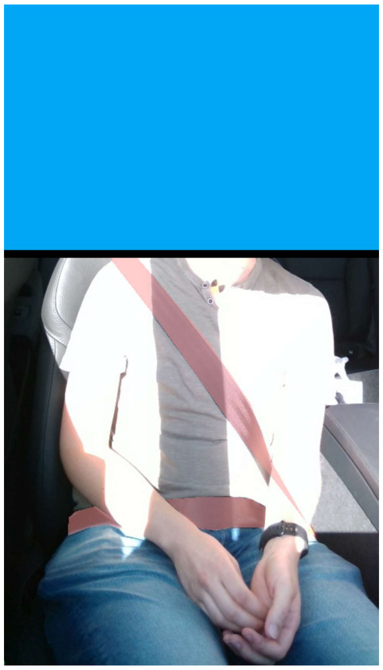

Figure D.3: An image where the annotation is on top of an arm.

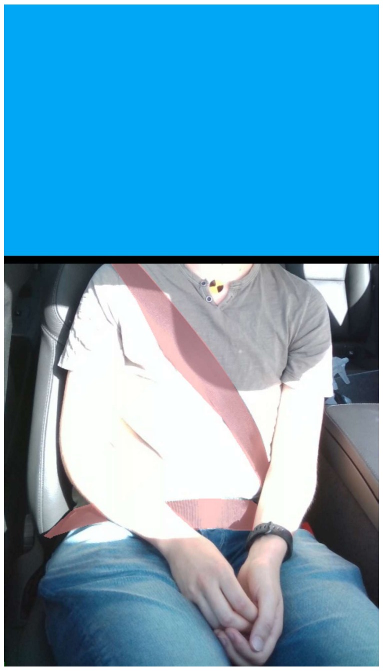

Figure D.4: An image where the annotation on the chest is too wide.

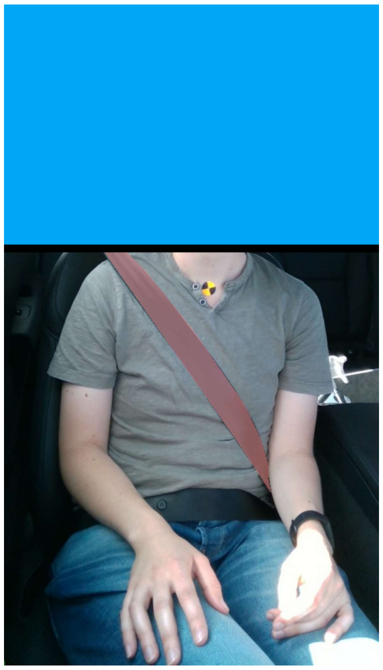

Figure D.5: An image where the lower part of the annotation is missing.

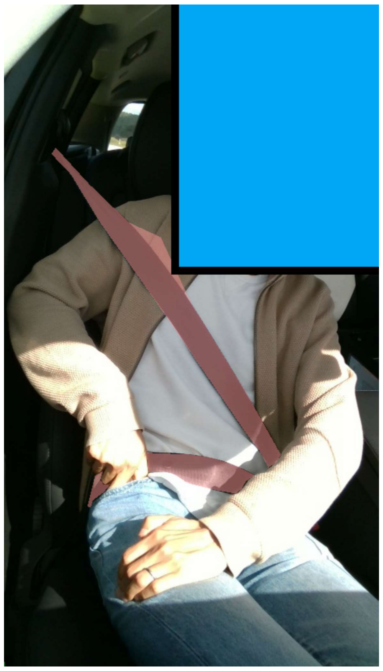

Figure D.6: An image where the annotation on the chest is too wide.

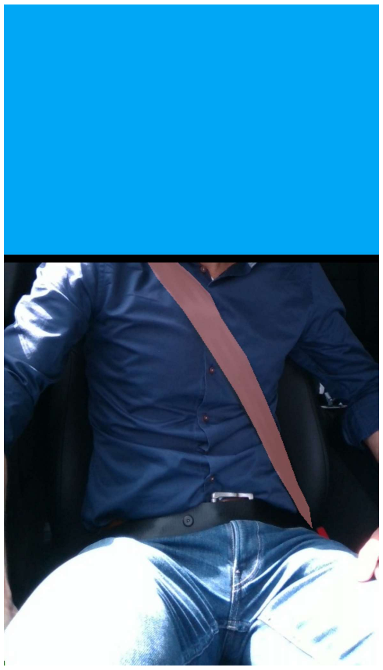

Figure D.7: An image where the lower part of the annotation is missing.

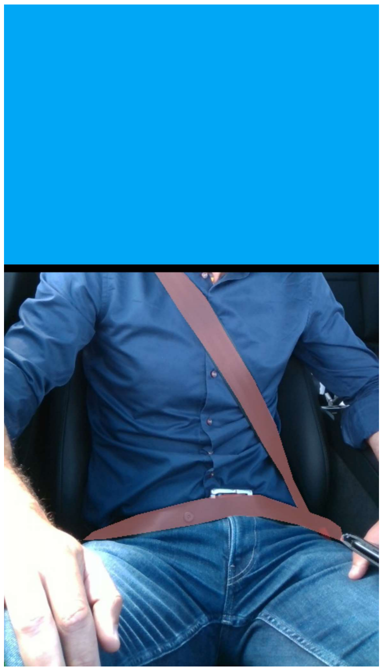

Figure D.8: An image where the lower part of the annotation stretches too far to the right.

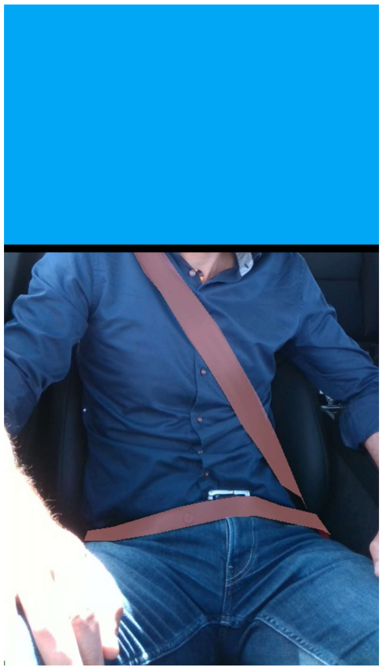

Figure D.9: An image where the upper segment of the annotation abruptly cuts off, as well as being too wide at the top.

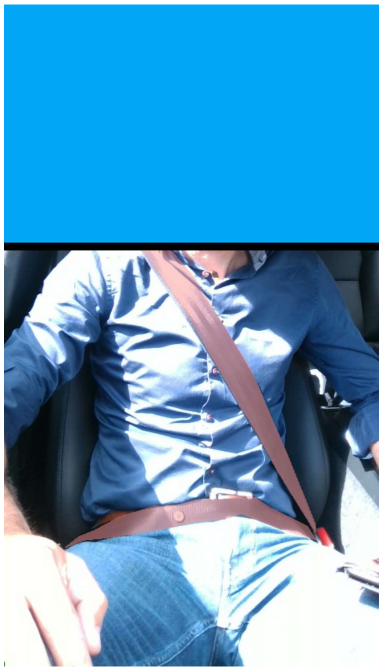

Figure D.10: A duplicate image.

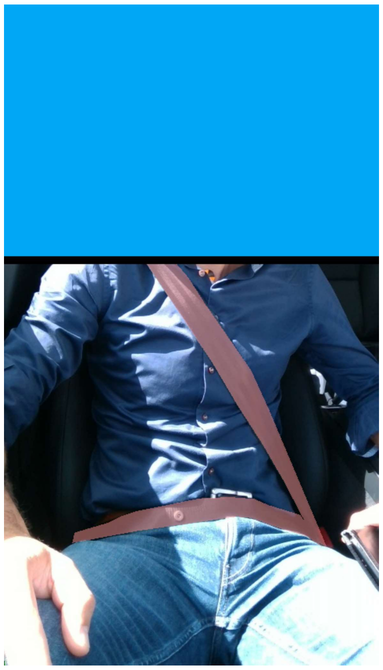

Figure D.11: An image where the annotation is too wide at the top.

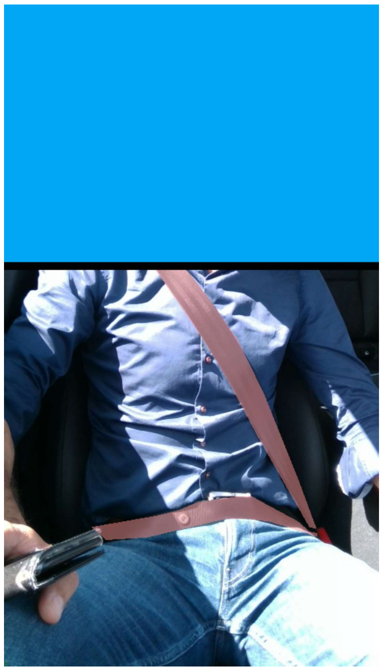

Figure D.12: A duplicate image.

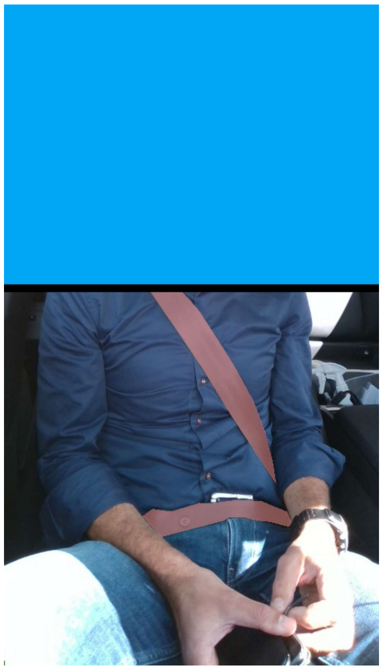

Figure D.13: A duplicate image.

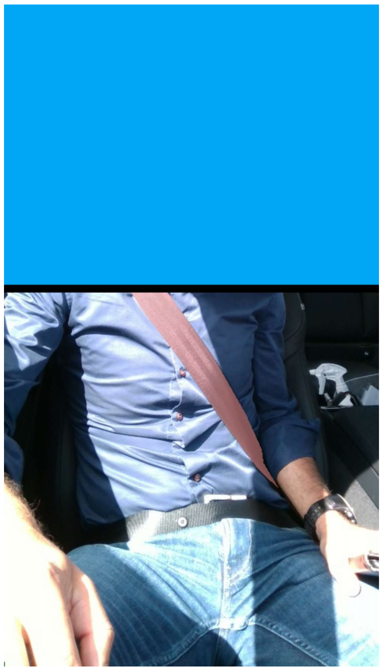

Figure D.14: An image where the lower part of the annotation is missing.

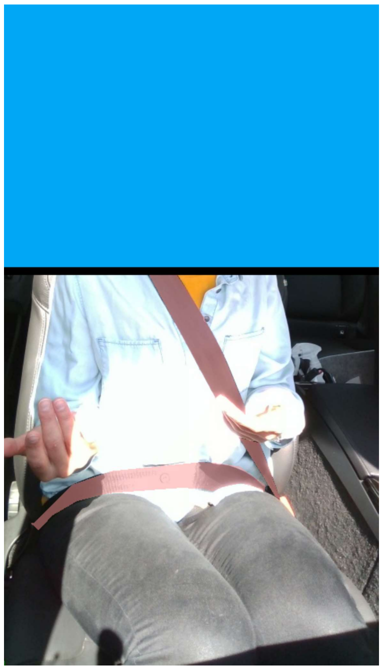

Figure D.15: An image where the left part of the lower annotation is too thin.

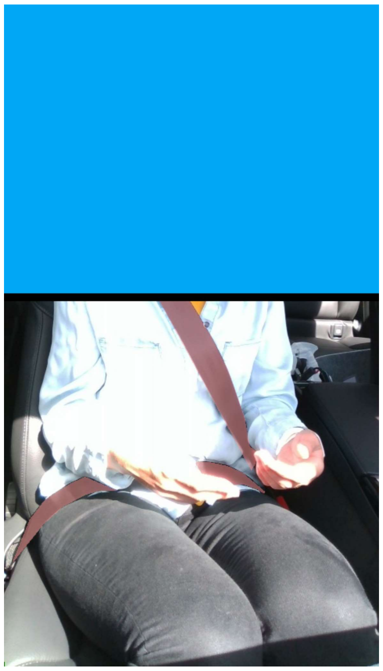

Figure D.16: An image where the lower part of the annotation is missing below the hand on the left.

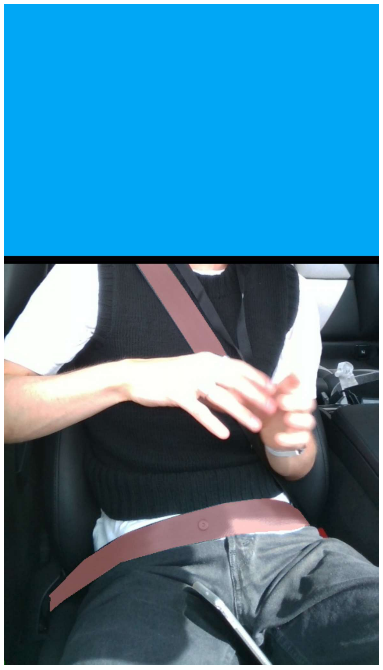

Figure D.17: An image where the annotation is missing between the hands.

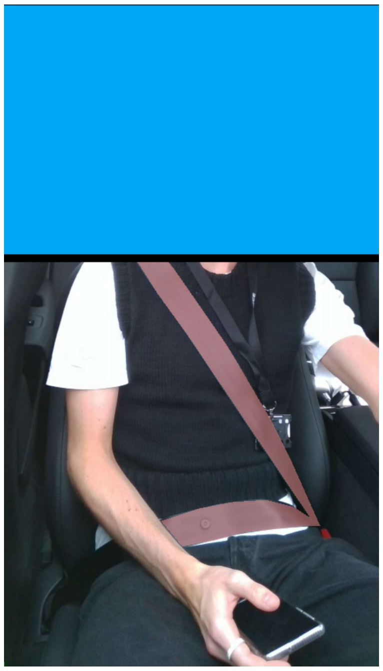

Figure D.18: An image where the left part of the lower annotation is missing.

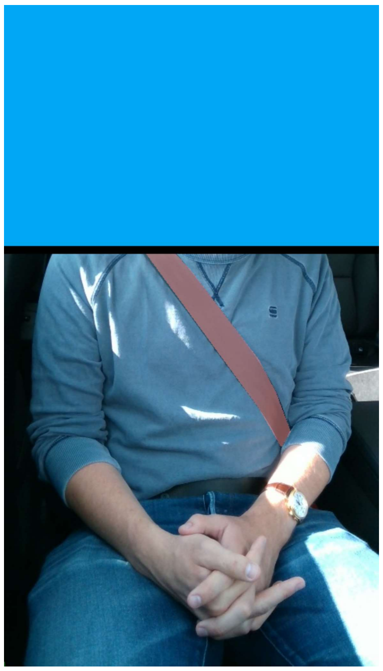

Figure D.19: An image where the lower part of the annotation is missing.

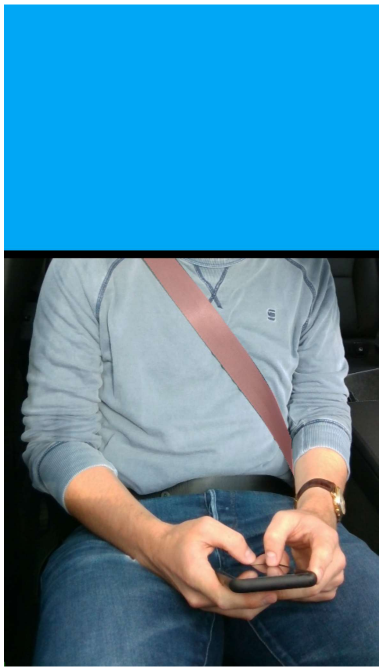

Figure D.20: An image where the lower part of the annotation is missing.