TECHNICAL UNIVERSITY OF MUNICH

TUM Data Innovation Lab

Uncertainty Estimation for Deep Medical Image Segmentation

Authors	Mohammad Alamleh, Guillem Brasó, Yu Chen, Sven Elflein, Martin Hermann
Mentors	Dr. Ghazal Ghazaei, Dr. Alexander Urich (Carl Zeiss AG)
Co-Mentor	Dr. Tobias Köppl (Department of Mathematics)
Project Lead	Dr. Ricardo Acevedo Cabra (Department of Mathematics)
Supervisor	Prof. Dr. Massimo Fornasier (Department of Mathematics)

Jul 2020

Abstract

Predictive uncertainty estimation on medical image segmentation tasks can provide risk analysis in particular when applying deep learning methods for segmentation and is a new and exciting branch of research. In medical image segmentation, an automatic process of semantic segmentation on Optical Coherence Tomography (OCT) images (B-scans) of the retina detects fluids representing different retinal diseases and thus can help to facilitate prognosis. Currently, modern machine learning methods including deep learning have achieved state-of-the-art performance on semantic segmentation tasks. Specifically, U-Net-based neural networks have dominated medical image segmentation, and have been successfully applied on the OCT image segmentation challenge RETOUCH, which aims at segmenting and detecting three fluid types in retinal regions of human eyes. Although benchmark segmentation results have been established on the challenge, a credible estimation of the uncertainty of the segmentation due to the input-output black-box characteristic of U-Net is still missing. Therefore, we present a range of uncertainty estimation methods on the RETOUCH segmentation task to obtain a reliable estimation of both epistemic and aleatoric uncertainties and then evaluate the quality of this estimation using calibration metrics to compare the quality of these methods.

Contents

1 Introduction

1.1 Image Segmentation for OCT Scans

Macular edema resulting from the accumulation of leaked fluids within the retina causes sudden and severe vision loss. Fortunately, this disease can be effectively treated by several personalized treatment regimens, such as pro re nata (PRN) and treat & extend (T&E). The injection decision in these regimens relies on the re-occurrence of retinal fluid accumulation which can be readily imaged using OCT imaging[5]. The OCT visualizes a high-resolution 3D image which includes a range of B-scans[11]. In the OCT images, three types of fluid including Intraretinal Fluid (IRF), Subretinal Fluid (SRF) and Pigment Epithelial Detachment (PED) (Fig. 1) are relevant to the imaging of biomarkers for visual acuity and the personalized treatment regimens[5]. Clinical studies have proven that quantifying the amount of the three fluid types as well as measuring their change of volume and area are helpful to guide the treatment regimens[1, 22, 7].

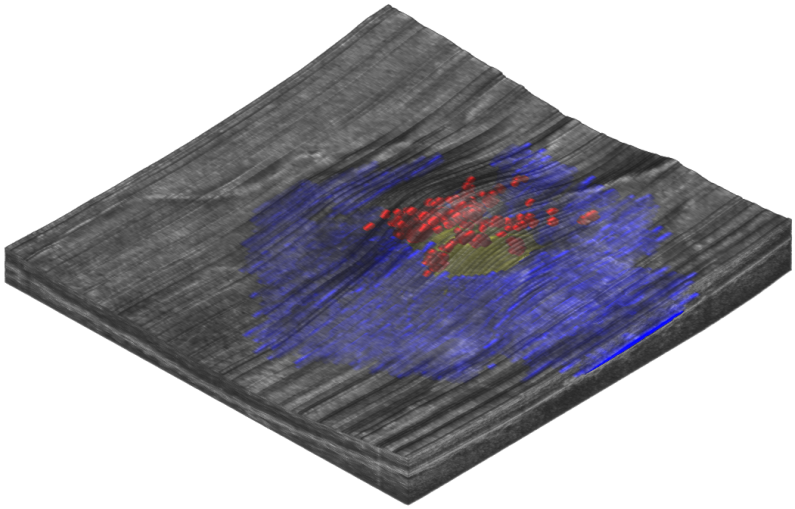

Figure 1: The three fluid types on a 3D OCT image corresponding to specific colors: Intraretinal Fluid (IRF), Subretinal Fluid (SRF), and Pigment Epithelium Detachment (PED) (Source: https://retouch.grand-challenge.org/Background/)

Unfortunately, manual quantification of the three fluids and their change via OCT scans is nearly impossible due to the huge amount of information in the scans. Therefore, it is necessary to establish automatic tools to achieve reliable quantification and analysis of OCT images. In 2017, the RETOUCH challenge had been launched aiming to acquire a state-of-the-art algorithm that detects and segments the three fluid types on a common OCT dataset known as RETOUCH[5] which is also the dataset used in our project. In the challenge, all eight participating teams applied deep-learning-based methods, in particular convolutional neural networks (CNN), which are a prevalent type of deep neural networks (DNN) to implement segmentation. Besides, five teams among the eight built their model based on fully convolutional neural networks (FCN)[19], in particular a very effective variant of FCN named U-Net[23], which has dominated medical image segmentation tasks since 2015 (including the first prize team SFU from Simon Fraser University,

Canada who also used a U-Net based model in the RETOUCH challenge[20]). Although modified architectures such as Attention UNet [14], UNet++[18] and UNet 3+[12] have been proposed and achieve the state-of-the-art on their medical image segmentation tasks, the improvements they achieved are minor when compared to U-Net.

1.2 Uncertainty Estimation for Neural Networks

However, the segmentation results obtained from CNNs on OCT images still lack predictive uncertainty due to the mechanism of CNN architectures which output a predictive distribution learned from the training dataset. Specifically, the distribution over observed data may be different from that of training data because of noise (e.g. low signal-to-noise ratio (SNR) on OCT images due to speckle[5]), and geometric transformations[26] (e.g. eye motion artifacts when obtaining OCT scans). Additionally, out-of-distribution (OOD) inputs (e.g. an unseen fluid type in the retina) may fool the prediction of DNNs, since DNN models such as the aforementioned U-Net may not encounter OOD inputs during training. These characteristics make the use of DNNs a potential risk in medical diagnoses, especially decision making based on the output of DNNs, since one does not know to what extend the output can be trusted[21]. Hence, to quantify these risks, it is crucial to develop uncertainty estimation for the safe deployment of DNNs on medically decision-making applications.
A range of methods has been proposed to estimate two types of uncertainties – epistemic and aleatoric uncertainty in DNNs. Firstly, epistemic uncertainty comes from the lack of training data in parts of the input domain, so it can be reduced via the collection of additional training data. This type of uncertainty can be quantified using approximate Bayesian approaches such as variational inference[4, 9] and Monte-Carlo Dropout[9], since full Bayesian neural networks, which would also provide such an estimation, are computationally almost intractable. In addition, non-Bayesian approaches, especially ensembling probabilistic DNNs[17], also play significant roles for quantifying not only epistemic but also aleatoric uncertainty[21]. Moreover, Test Time Augmentation[26] and Learned Loss Attenuation[15] estimate the effect of different transformations on the input images and thereby quantify aleatoric uncertainties. Finally, the probability of a model being wrong about its prediction can also reflect the uncertainty of the prediction, as we will discuss in later.

1.3 Project Objectives

Currently, nobody has established a benchmark of uncertainty estimation for the RETOUCH segmentation challenge. As a result we mainly aim at constructing a baseline framework which compares different uncertainty estimation methods including Test Time Augmentation, Monte Carlo Dropout, Deep Ensembles, Dropout Deep Ensembles, Learned Loss Attenuation, and Direct Error Prediction which is a method that we propose for this task.

2 Image Segmentation

Image Segmentation is a basic computer vision task that consists of partitioning a digital image into disjoint regions (or segments). In this project, we focus on semantic image segmentation, which aims at labeling every individual pixel with one of the classes from a fixed set $\{1, \dots, C\}$.
In this chapter, we focus on how we build the segmentation model that we will later perform uncertainty estimation on in Chapter 3. We first describe the dataset that we will work with, introduce notation, and define our metrics of interest. We then move on to describe our model and results in detail, including our network's architecture, loss functions, and training procedure.

2.1 The RETOUCH Dataset

The RETOUCH dataset [5] is a popular database of OCT scans of the retina for the detection of three types of fluid: Intraretinal Fluid (IRF), Subretinal Fluid (SRF) and Pigment Epithelial Detachment (PED). These three fluid types, together with a *background* class, form the set of classes $\{1, \dots, C\}$ in our problem at hand (i.e. $C = 4$).
The training set consists of a total of 70 OCT volumes. Each volume (i.e. 3D scan of the retina) consists of a total of either 128 or 49 2D B-scans with pixel-wise annotations by experts. The entire dataset amounts to a total of 7064 B-scans. Since test data is not publicly available, we leave out one third of the training data for testing purposes. Out of the remaining two thirds, we use one tenth as the validation set. To avoid leakage of patient information, data splits are always done at the volume level, hence preventing two B-scans of the same patient from being used both during training and validation.
The 70 OCT volumes were acquired from three different device manufacturers, with 24, 24, and 22 volumes obtained from Cirrus, Spectralis, and Topcon devices, respectively[1]. This variability introduces a challenge for segmentation methods, as volumes from different manufacturers differ significantly in terms of noise and resolution.
Another challenging feature of this dataset is its severe class imbalance. Since our target retinal fluids typically accumulate sparsely in small regions, the vast majority of pixels in B-scans do not contain any fluid and are labeled as background. In fact, only 1% of pixels belong to either of the three fluids.

2.2 Notation

Before going any further, we formalize our problem at hand and introduce the notation that we will use for this and all the upcoming chapters.
As explained in the introduction, our general task is to label each pixel from images in our dataset with a class from a fixed set $\{1, .., C\}$. Formally, given a $d \times d$ image $X \in \mathbb{R}^{d \times d}$, our goal is to learn a mapping $F : \mathbb{R}^{d \times d} \to \{1, .., C\}^{d \times d}$ that outputs the per-pixel classes. In practice, we model F as a neural network $F_{NN} : \mathbb{R}^{d \times d} \to \mathbb{R}^{d \times d \times C}$ that does not output the hard-assignments but, instead, a probabilistic mapping that determines the per-class *scores* at every pixel. More specifically, for every $1 \leq i \leq d$

[1] To be consistent with the original dataset configuration, we make sure that our own training, validation and test sets' volumes are also uniformly distributed among manufacturers.

and $1 \leq j \leq d$, $f_{ij}(X) := (F_{NN}(X))_{i,j} \in \mathbb{R}^C$ is a C-dimensional vector representing the activation of the network for each class for the pixel at coordinates (i, j). In cases where the input image X is fixed and clear from the context, we write $f_{ij} := f_{ij}(X)$. To obtain a prediction $\hat{Y} \in \{1, .., C\}^{d \times d}$, we just take:

$$\hat{Y}_{i,j} \coloneqq \left(\arg \max_{c \in (1,...,C)} (f_{ij})_c(X) \right)_{i,j=1}^{d}$$

i.e. assign to each pixel the class with the highest activation. Also, for normalization purposes it is common to apply the *softmax* function

$$\sigma : \mathbb{R}^C \to \mathbb{R}^C, \sigma(f_{ij})_c = \frac{\exp{(f_{ij})_c}}{\sum_{c'=1}^{C} \exp{(f_{ij})_{c'}}}$$

to the outputs. This does not change the prediction, as the relative ordering of activations will stay the same, but proves to be useful for uncertainty estimation. In the following, unless stated otherwise, we will assume that such a softmax has been applied to every tuple f_{ij}, which in particular implies that the elements are non-negative and sum up to 1.

F_{NN} will be learned from our training dataset, which we refer to as $\mathbb{D} \coloneqq (\mathbb{X}, \mathbb{Y})$, where $\mathbb{X} \subset \mathbb{R}^{d \times d}$ is our set of images, and $\mathbb{Y} \subset \{1, .., C\}^{d \times d}$ our set of ground truth masks. For every mask $Y \in \mathbb{Y}$, we denote with $Y^{one-hot}$ its *one-hot* representation.

2.3 Evaluation Metrics

We now proceed to review the metrics we will use to evaluate the quality of segmentation results.

Accuracy. We use *accuracy* to measure pixel-wise prediction errors. We define the average accuracy for an image as:

$$Accuracy(\hat{Y}, Y) \coloneqq \frac{1}{d \times d} \sum_{i=1}^{d} \sum_{j=1}^{d} \mathbb{1}[\hat{Y}_{i,j} = Y_{i,j}]$$

Note that, since our classification task is severely imbalanced, *accuracy* is not a particularly meaningful metric. Indeed, since only $\approx 1\%$ of pixels belong to a target fluid, an average accuracy of $\approx 0.99\%$ can be achieved by simply not predicting any fluid at all. Hence, we mainly resort to alternative more informative metrics for our task.

Dice Score. The Dice Score is the main metric we use to evaluate the segmentation quality of our output masks. For two sets A, B, such that either $A \neq \emptyset$ or $B \neq \emptyset$, we define.

$$Dice(A, B) = \frac{|A \cap B|}{|A| + |B|}$$

Whenever $A = B = \emptyset$, we define $Dice(A, B) = 1$. Now, for every class c, let us denote by $Y^{one-hot} = c$ (resp. $\hat{Y} = c$) the set of pixels in $Y^{one-hot}$ (resp. $\hat{Y}$) belonging to class c. We now define:

$$AvgDice(\hat{Y}, Y) \coloneqq \frac{1}{\sum_c w_c} \sum_{c=1}^{C} w_c Dice(\hat{Y} = c, Y^{one-hot} = c)$$

Where $\{w_c\}_{c=1}^{C}$ is a set of non-negative weights that determines the contribution of each class' Dice score to the average.
To follow the RETOUCH Challenge evaluation procedure, we compute the Dice score over *voxels* (instead of pixels) of entire 3D volumes, and report the average *AvgDice* score across all test volumes[2], with $w_c = 0$ for the background class. Extending the definitions above for 3D volumes is straightforward, but not done here for notational convenience. For the experiments in Chapter 4, we set $w_c = \frac{\#\text{pixels in the dataset}}{\#\text{ background pixels in the dataset}}$, that is, the inverse ratio of background pixels. We do so because in that case dice scores are computed scanwise, and empty scans are much more prevalent than empty volumes. Hence, we need a consistent way to evaluate scans with no fluid.

2.4 Approach

In the following sections, we describe the design choices, implementation details, and results of our segmentation model, which we later will use for the task of uncertainty estimation in Chapter 3.

2.4.1 Architecture

We use a U-Net architecture [23] for our segmentation model F. This is a common baseline for biomedical image segmentation, and has shown good *off-the-shelf* performance for our task. An illustration of the architecture is given in Figure 13.
We only make two small modifications upon the original architecture. The first one is to add Batch Normalization [13] after every convolutional layer. Empirically, we observe that this addition allows for much faster and stable training. The second change we make is adding dropout [25] after every convolutional *block*. Despite this change yielding a slight improvement in segmentation quality, its main purpose is to allow Monte Carlo-based uncertainty estimation methods to be implemented on top of our network, as will be explained Chapter 3.

2.4.2 Loss Functions

We have experimented using two standard loss functions as well as combinations of both for our segmentation task: *Cross-Entropy* and *Dice Loss*. We now proceed to define both of them.
Cross-Entropy Loss. We can use the Cross-Entropy Loss to optimize for *pixelwise* classification performance. We define it as:

$$CrossEntropy(F_{NN}(X), Y) \coloneqq -\frac{1}{d \times d} \sum_{i=1}^{d} \sum_{j=1}^{d} \sum_{c=1}^{C} w_c Y_{i,j,c}^{one-hot} \log\left(\sigma(f_{i,j,c})\right)$$

where w_c denotes a non-negative parameter used to weight pixels from class c in the overall loss. The standard way to define the Cross-Entropy Loss is by setting $w_c = 1$ for every $c \in \{1, \ldots, C\}$. However, with such definition, just like with accuracy, this loss function may not be best suited for our task. Given our dataset's class imbalance, most of the

[2]Volumes with no fluid annotations are skipped during evaluation.

terms in the sum above will correspond to background regions and hence, fluid regions will have a comparably smaller contribution to the overall loss. To address this issue, w_c is typically given a value proportional to the inverse frequency of the given class in the dataset. We will experiment with these parameters in Section 2.4.4

Dice Loss. Since our main segmentation metric is the Dice score, it's natural to ask whether there's a differentiable surrogate for it, and the answer is affirmative. We first define, for two arbitrary 2D tensors $A, B \in [0,1]^{d \times d}$:

$$DiceLoss(A, B) := 1 - \frac{\sum_i \sum_j (A_{i,j} * B_{i,j})}{\sum_i \sum_j (A_{i,j} + B_{i,j}) + \epsilon}$$

Where $\epsilon > 0$ is a small constant used for numerical stability. Note that whenever A and B are binary, $DiceLoss(A, B) = 1 - Dice(A, B)$. Now, given an arbitrary tensor $Z \in \mathbb{R}^{d \times d \times C}$, let $Z_{:,:,c}$ denote the tensor corresponding to fixing the third dimension at index c. We define:

$$AvgDiceLoss(F_{NN}(X), Y) := \frac{1}{\sum_{c=1}^{C} w_c} \sum_{c=1}^{C} w_c * DiceLoss((\sigma(F_{NN}(X)))_{:,:,c}, Y_{:,:,c})$$

where w_c denotes a non-negative parameter used to weight class c in the overall loss and in our experiments in Section 2.4.4 we will mainly experiment with setting $w_c = 0$ for the background class.

2.4.3 Training

We train our network end-to-end using an Adam optimizer [16] with a starting learning rate 0.003 and betas set to default values. We multiply the learning rate by 0.5 every 10 epochs, and train for a total of 40 epochs, which takes around 8 hours on an NVIDIA Titan X GPU. Images are resized to 512×512 and batch size is set to 4. During training, dropout probability is set to $p = 0.2$ and no weight decay regularization is used. We also use *data augmentation* during training, which means we apply random affine transformations (rotations between $-20°$ and $20°$ in 50% and horizontal mirroring in 25% of cases) to each image, which is known to help with generalization.

At each epoch, we compute the average dice score over fluid classes in the validation set and select the model corresponding to the epoch with the highest value.

2.4.4 Results

In this section, we present an ablation study in which we investigate the choice of our model's loss function. We experiment using either the Cross-Entropy Loss, the Dice Loss, or a combination of both. Results are summarized in Table 1

Cross-Entropy Loss Variants. For the Cross-Entropy loss, we consider using its standard version (i.e. with $w_c = 1$ for every class), or its weighted variant, in which every w_c is set to the inverse frequency of class c among all pixels from images in the dataset and rescaled so that $\sum_{c=1}^{C} w_c = 1$. Surprisingly, we observe that the latter works worse.

Dice Loss Variants. For the Dice Loss, we consider either setting $w_c = 1$ for every c to 1 or setting $w_c = 0$ for the background class and 1 for the remaining ones. We observe

Table 1: Performance comparison among different model variants and a competing method.

Setting	Metrics	Dice Score			
		Mean	PED	IRF	SRF
CE Loss	$w_c = 1 \; \forall c$	0.580	0.603	0.537	0.600
	Weighted by Inv. Frequencies	0.524	0.523	0.494	0.556
Dice Loss	$w_c = 1 \; \forall c$	0.644	0.652	0.556	0.725
	$w_{background} = 0$	0.644	0.635	0.640	0.658
Dice + CE Loss	$w_c = 1 \; \forall c$	0.646	0.604	**0.653**	0.680
	$w_{background} = 0$	0.660	0.639	0.648	0.695
Helios team	U-Net + heavy engineering	**0.680**	**0.730**	0.610	**0.700**

that this change does not improve the mean Dice Score, but does yield a more uniformly distributed performance among classes.

Combining both losses. Lastly, we train the model with both loss functions (Cross-Entropy and Dice) in an attempt to further boost performance. We experiment with setting the background weight for the Dice loss to either 0 or 1 and observe again, that the former works best and yields our top performing model.

External Comparison. Since we do not have access to the test set, we cannot directly compare our method with most competing teams in the RETOUCH challenge. However, one of the competing teams, *Helios*, which finished in 6th overall position, did use the same test set as us for their ablation studies[3]. Hence, we report their metrics as a reference. Their approach also involves a U-Net (trained with Cross-Entropy loss), but further includes heavily engineered preprocessing and postprocessing pipelines which we do not use. Overall, our method performs comparably while being significantly simpler, end-to-end trainable, and having access to less training data.

3 Uncertainty Estimation

With growing interest in not only the quality of the prediction itself, but also the estimation and quantification of a models uncertainty about that prediction, there is an increasing number of different methods proposed for this task. As part of this project, we adapted several of these approaches for the purpose of medical image segmentation, tried to fit them in a unified framework and improve on them where possible.

Formally, the goal of uncertainty estimation is to estimate a second function U on top of the segmentation prediction F_{NN} introduced in Chapter 2, where $U : \mathbb{R}^{d \times d} \to Q$, and either $Q = \mathbb{R}$ (in which case we will talk of *imagewise* uncertainty estimation), or $Q = \mathbb{R}^{d \times d}$, which is then called *pixelwise* uncertainty estimation. Either way, we expect $U(X)$ to reflect in some way the uncertainty the network has about its prediction $\hat{Y}$. For the first few sections, we will focus on pixelwise uncertainties, while different ways to estimate imagewise uncertainty are then discussed in Section 3.5.

[3]They performed 9-fold Cross-Validation and we report their performance over 3 splits vs our performance on a single split, which means that they had access to significantly more training data.

3.1 Baseline Approach

It is tempting to think of the magnitude of the network's activation as a measure of its certainty. This leads to the naive approach of treating every entry $f_{ijc} := (f_{ij})_c$ as a predicted probability for the corresponding class c. We then have two ways to estimate the uncertainty from this: Either, we can calculate the entropy of that distribution:

$$-\sum_c f_{ijc} \log_2 f_{ijc}$$

In addition, we may use the *residual probability*, that is (using $\tilde{c} = \arg\max_{i \in (1,\dots,C)} f_{ij}$)

$$\sum_{c \neq \tilde{c}} f_{ijc} = 1 - f_{ij\tilde{c}}$$

We will see in Section 4.5.2 that this approach is not sufficient to capture most uncertainty of the network, but it can serve as a baseline to compare the more sophisticated methods against.

3.2 Sample-based Methods

Usually, $(U(X))_{ij}$ is assumed to be some moment or other quantity derived from some underlying probability distribution. This may sound rather vague, as the actual assumptions and models differ quite a lot for the different methods, but nevertheless it suggests the use of a *Monte Carlo* approach, i.e. drawing multiple samples from the distribution and then using them for the uncertainty estimation. We will call the methods resulting from this *sample-based methods*. Figure 2 shows the general pipeline for sample-based methods.
In our scenario, this translates to the following: For a given input image X, we generate $N \in \mathbb{N}$ different samples $(f_{ij}(X))_{n=1}^N$ (which we will denote by f_{ij}^n), even though the way these samples are generated will differ from method to method. The output of the model is then computed as the mean of the individual samples, i.e. $f_{ij} = \frac{1}{N} \sum_{n=1}^N f_{ij}^n$. For the uncertainty, we use some quantity derived from either the mean f_{ij} or the individual samples f_{ij}^n. Possible choices are:

- The Monte Carlo estimation of the variance, averaged across all classes: $\frac{1}{C} \sum_{c=1}^C \frac{1}{N} \sum_{n=1}^N \left(f_{ijc}^n - f_{ijc} \right)^2$
- The *entropy of averages*: $-\sum_{c=1}^C f_{ijc} \log_2 f_{ijc}$
- The *average of entropies*: $\frac{1}{N} \sum_{n=i}^N -\sum_{c=1}^C f_{ijc}^n \log_2 f_{ijc}^n$
- The residual probability $1 - f_{ij\tilde{c}}$, where again $\tilde{c}$ denotes the class with the highest value f_{ijc}:

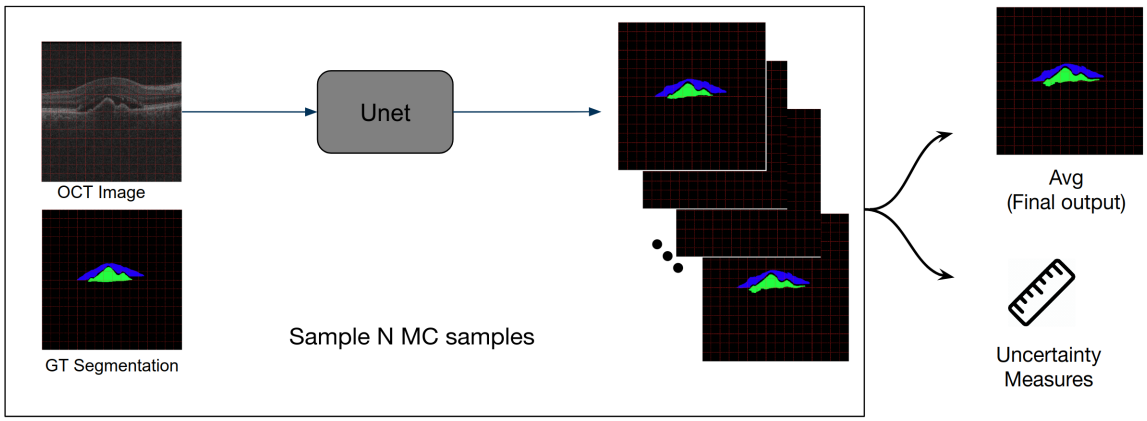

Figure 2: General principle of sample based methods

3.2.1 Monte Carlo Dropout

If we adjust the notation given in Chapter 2 to a Bayesian setting, following the outlines of [9] and [24], we now assume that we don't aim at representing an exact function output $Y = F_{NN}(X)$, but rather want to estimate the conditional probability $p(Y|X, \mathbb{D})$ (we recall that $\mathbb{D} = (\mathbb{X}, \mathbb{Y})$ is the training data). Using $\mathbf{W}$ to denote the weight matrix of the neural network, this leads to

$$p(Y|X, \mathbb{D}) = \int p(Y|X, \mathbf{W}) \times p(\mathbf{W}|\mathbb{D}) \, \mathrm{d}\mathbf{W} \tag{1}$$

The above integral can be approximated using variational inference by minimizing the KL-Divergence between the variational distributions $q(\mathbf{W})$ and $p(\mathbf{W}|\mathbb{D})$ which gives us

$$p(Y|X, \mathbb{D}) \approx q(Y|X, \mathbb{D}) = \int p(Y|X, \mathbf{W}) \times q(\mathbf{W}) \, \mathrm{d}\mathbf{W} \tag{2}$$

The weight matrix $\mathbf{W}$ in a neural network is layered. More specifically, $\mathbf{W} = (\mathbf{W}_i)_{i=1}^{L}$, where L is the maximum number of layers in the network. This means we can sample $q(\mathbf{W})$ per layer as follows:

$$\mathbf{W}_i = \mathbf{M}_i \times diag([z_{i,j}]_{j=1}^{K_i}) \quad z_{i,j} \sim Bernoulli(p_i) \quad i = 1, ..., L, j = 1, ..., K_{i-1} \tag{3}$$

Here K_i denotes the number of neurons in the ith layer and $\mathbf{M}_i$ are the variational parameters to be optimized. With this, one can estimate the integral in (2) by Monte Carlo sampling from $q(\mathbf{W})$. Gal and Ghahramani [9] showed that this can be approximated by performing dropout on the layer i of a network that has weights $(M_i)_{i=1}^{L}$, which is the central idea of Monte Carlo Dropout.
To implement this, we added a dropout layer to the U-Net after each encoder and decoder block excluding the bottleneck, as proposed in [24] and leading to the architecture depicted in Figure 3.

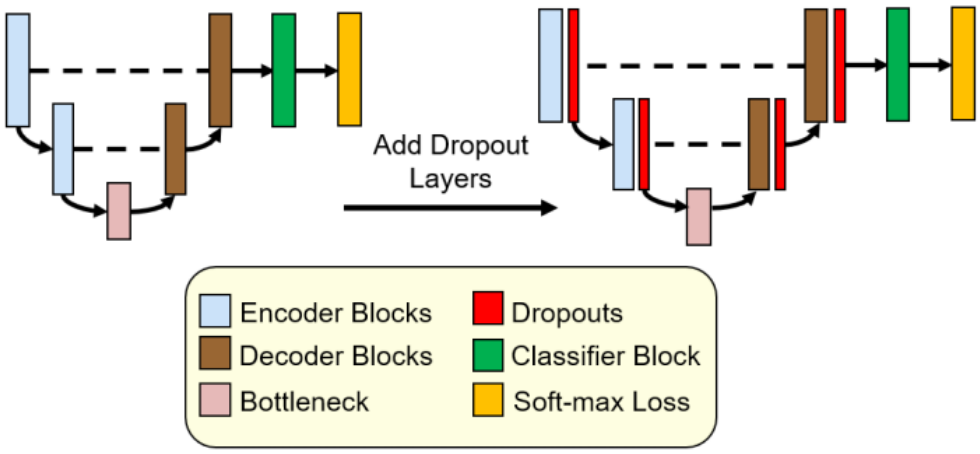

Figure 3: The U-net architecture after adding dropout layers (Source: Roy et al. [24], p. 3)

At test time, we keep the dropout layer activated, then we evaluate the model for each input image N times resulting in N samples f_{ij}^n. From there, we calculate the output and uncertainty measures as described above.

3.2.2 Test Time Augmentation

Data augmentation, that is applying some form of rotation, reflection, noise or other distortion to input images of a neural network, is a common technique to prevent overfitting and improve generalization during training. During test time however, it can be used to predict the networks aleatoric uncertainty, as described in [2] and [26].
For this, we assume that the input to the network, an image X, has been generated from an actual underlying image X_0 by some corruption (which can stem e.g. from technical or physical issues or inter-patient variability), which we will represent by some affine transformation τ_β consisting of rotation and mirroring, the amount of which is parameterized by the random variable β, and additional noise ϵ, leading to the model $X = \tau_\beta(X_0) + \epsilon$.
While we have access to the augmented image X and can (for every pixel) predict an output $f_{ij}(X)$, we are actually interested in the conditional distribution $\tau_\beta(f_{ij}(X_0))|X$ whose mean we can use as prediction and the uncertainty of which we want to estimate. The reason we apply τ_β to $f_{ij}(X_0)$[4] is that we care about the location of fluids in the actual (possibly rotated) input image X. We can sample from this conditional distribution via $X_0 = \tau_{-\beta}(X) - \epsilon$, which then enables us to use the standard Monte Carlo techniques described above.
In practice, this translates to the following procedure visualized in Figure 4: We sample from β and ϵ N times, generating N new input images $X_n := \tau_{\beta_n}(X) + \epsilon_n$[5], collect the respective predictions $F_{NN}(X_n)$, and finally reverse the rotation and mirroring to generate

[4]This is also a slight abuse of notation, as we apply τ_β to the whole image and then select the specific pixel. It would be more correct to write $(\tau_\beta(F_{NN}(X_0)))_{i,j}$, but we feel that this does not add more clarity.
[5]Where we also used that $\beta \sim -\beta$ and $\epsilon \sim -\epsilon$.

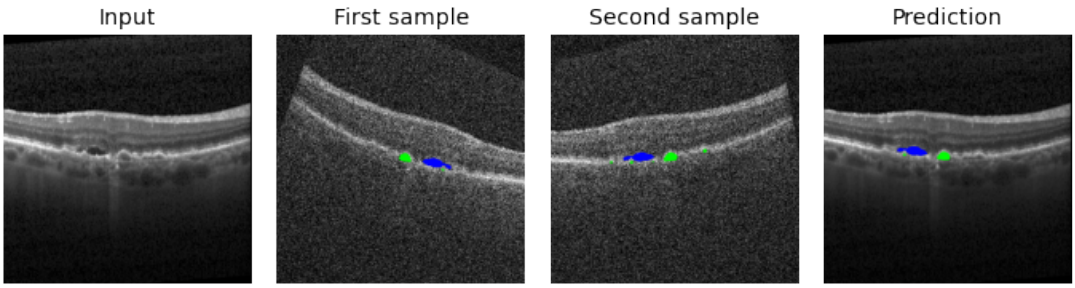

Figure 4: Test Time Augmentation with two different samples. The noise is exaggerated to be better visible.

the samples f_{ij}^n.

3.2.3 Deep Ensembles

Lakshminarayanan, Pritzel, and Blundell [17] propose deep ensembles to achieve simple and salable predictive uncertainty estimation for the following reasons: Firstly, the quality of uncertainty estimation obtained by Bayesian neural networks heavily relies on whether the assumed prior distribution is correct and the approximation extent (that is subject to computational constraints). Also, although Monte Carlo Dropout is relatively simple to implement for real-world applications, its dropout rates are not tuned based on the training data. Finally, it is assumed that an ensemble strategy is more powerful in approximating the true representable function when the function does not lie in the hypothesis space [8].
The central idea of ensemble methods is that instead of training only a single model, we instead train N different models, each with different initial weights and a different shuffling of the training data, assuming that each model will capture slightly different aspects of the feature space. In contrast to [17], we do not use *adversarial training* (introduced in [10] for encouraging local smoothness) that was implemented for robustness against model misspecification and out-of-distribution examples, since their experimental results indicate that it becomes barely helpful when ensembling more models [17] and it increases computational costs to a great extent.
After obtaining all trained models $\{\theta_1, \theta_2, ..., \theta_N\}$, we can simply generate samples $f_{ij}^n(X) := f_{ij}^{\theta_n}$, i.e. consider the output of every ensemble member as a new sample. This now fits nicely into the Monte Carlo framework that we established before and uncertainties can be calculated as described above.

3.2.4 Dropout Ensembles

Based on Monte Carlo Dropout discussed in Section 3.2.1 and Deep Ensembles in Section 3.2.3, Bachstein [3] proposes to combine both methods. This is achieved by using an ensemble of models and additionally enabling dropout layers at test time for each of them. One can then sample from each model within the ensemble alternatingly. The premise is that by combining both methods one obtains a better approximation of the overall weight distribution in comparison to considering Deep Ensembles or Monte Carlo Dropout independently.

3.3 Learned Loss Attenuation

Kendall and Gal [15] propose a sampling-free approach for aleatoric uncertainty estimation for regression and classification tasks. In particular, the approach focuses on heteroscedastic regression, i.e., it assumes that observation noise varies depending on the input image. As we consider a segmentation task, we implement the part of the approach designated for classification. For this, the uncertainty is modeled over the logit space of the output of the neural network for each pixel and class, which in particular means that, in contrast to the previous methods, we assume that **no** softmax function has been applied to f_{ij}. We now model the uncertainty by a parametric distribution over f_{ij}, optimizing the parameters of the distribution during training.

Model Adjustments In order to incorporate this approach within our standard model, we change each logit output f_{ij} of the network to predict two vectors instead: A mean vector μ_{ij} and log-variance $\log \sigma_{ij}^2$. This amounts to a change in the final feature map dimensions from $d \times d \times C$ to $d \times d \times 2C$. In order to ensure positivity of the variance values, an additional exponential activation is applied to the log-variance output to obtain the final output variance σ_{ij}^2.

Training Using the outputs $\mu_{ij}, \sigma_{ij}^2 \in \mathbb{R}^C$, one can parameterize the distribution over the logits for each pixel $f_{ij} \sim N(\mu_{ij}, diag(\sigma_{ij}^2))$. The overall objective then becomes maximizing the log likelihood of this model given by

$$\log \mathbb{E}_{f_{ij} \sim \mathcal{N}(\mu_{ij}, diag(\sigma_{ij}^2))} \left[f_{ij\tilde{c}} \right] \tag{4}$$

where $\tilde{c}$ is the groundtruth class for pixel i. Since this expectation is intractable, Kendall and Gal [15] propose to approximate the expectation using Monte Carlo integration by sampling from the logit distribution N times[6]. This is fast, as only a single forward pass is required to obtain the means and variances of the logit distribution. Afterwards, only sampling from the analytic distribution is required to approximate the integral, which is comparatively cheap. Formally, this means we calculate $f_{ij}^n = \mu_{ij} + \sigma_{ij}\epsilon_n$, where $\epsilon_n \sim \mathcal{N}(0, I)$, and then use $\frac{1}{N} \sum_{n=1}^{N} f_{ij}^n$ to calculate the loss. Since the model is able to adjust the predicted variance for each logit output, it is able to reduce the expected loss under the predicted distribution, which can be interpreted as a learned loss attenuation. The average predicted variance $\frac{1}{C} \sum_{c=1}^{C} \sigma_{ijc}^2$ can then be used as an estimation of aleatoric uncertainty, together with the same metrics the baseline uses.

3.4 Direct Error Prediction

Arguably, a good uncertainty estimate should reflect the probability of a model being wrong about its prediction. A natural question then arises: Could we *learn* to output this probability directly, and use the result as an uncertainty estimate?
We propose to do so by having a new branch of our U-Net output, for every pixel, the probability that our U-Net labels it incorrectly. This can be casted as a pixelwise binary

[6] We stress that this is different from the sample based methods of the previous section. Here, we sample during **training**, not during test time.

classification problem and can be trained with the already available ground truth data. We refer to this approach as *direct error prediction* and provide an illustration of it in Figure 5.

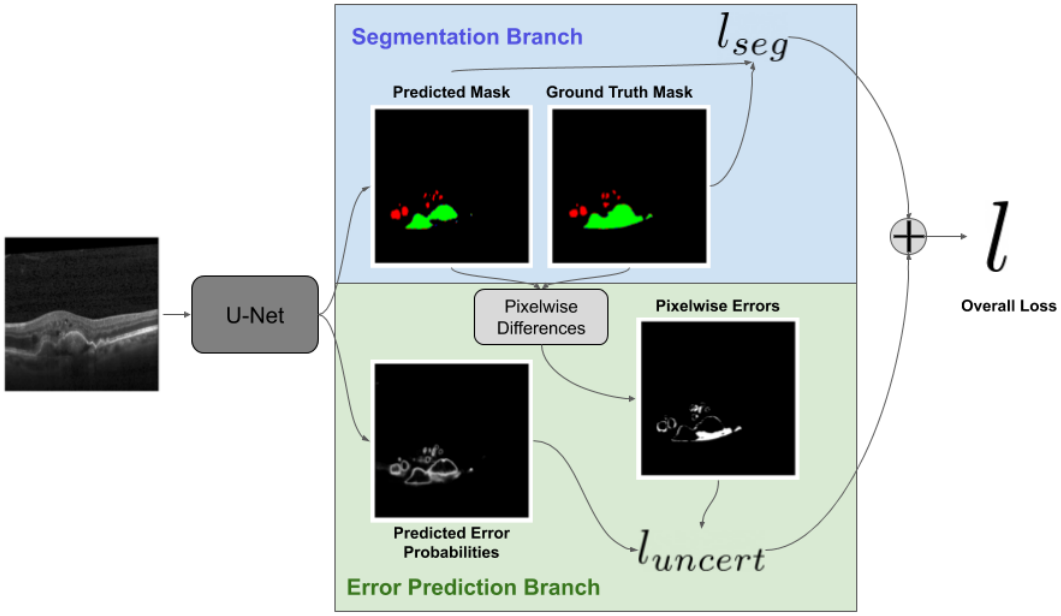

Figure 5: Training pipeline with *Direct Error Prediction*

More specifically, we add a sequence of convolutional layers after our U-Net's last feature map in order to estimate uncertainty. During training, at every iteration we first follow the pipeline described in Section 2.4.3. That is, we feed each image $X \in \mathbb{R}^{d \times d}$ through the U-Net, obtain our output mask $F_{NN}(X)$, compare it to the target mask Y and compute our segmentation loss $l_{seg}(F_{NN}(X), Y)$. After that, we compute another loss term for the new branch. We first define, for $1 \leq i \leq d, 1 \leq j \leq d$ a new target $Y_{uncert} \in \{0,1\}^{d \times d}$:

$$Y_{i,j,uncert} = \begin{cases} 1 & \text{if } \hat{Y} = Y_{ij} \\ 0 & \text{otherwise} \end{cases}$$

That is, Y_{uncert} is a a binary mask indicating the pixelwise errors in our segmentation $\hat{Y}$[7]. We then compute a new tensor $g(X)$ by feeding the last feature map of the U-Net to the newly added convolutional layers followed by a sigmoid activation, and treat the result as our uncertainty estimate[8]. We then compute the binary cross-entropy loss between the probabilities obtained and Y_{uncert}, which we denote as $l_{uncert}(g(X), Y_{uncert})$. Our final loss l is computed as $l = l_{uncert}(g(X), Y_{uncert}) + l_{seg}(F_{NN}(X), Y)$.
Note that this approach does not, in principle, require retraining an entire network from scratch. Instead, the additional convolutional layer could be trained in isolation by minimizing l_{uncert} and leaving all other layers (and hence, also l_{seg}) frozen. However, in our

[7]Recall that $\hat{Y}$ is obtained by selecting the class with highest score from $F_{NN}(X)$ pixelwise.
[8]Note that we are effectively modelling our uncertainty score as the probability of our model's segmentation being incorrect

experiments, we have observed that joint training of the segmentation and uncertainty estimation branches yields a slight improvement in segmentation performance, and hence we opt for it (see Table 2).
At test time, we obtain the uncertainty estimates with a single forward pass through the network, using the same uncertainty outputs as during training. Optionally, we can also use the same estimates the baseline uses, i.e., calculate entropy and residual probability. Note that the only additional computation corresponds to the newly added convolutional layers, and has no significant additional overhead, which makes the runtime of this new method compare favorably to the alternatives.

3.5 Calculation of Imagewise Uncertainty

Up to now we have only discussed pixelwise uncertainties, which correspond to one uncertainty value for every one of the $d \times d$ pixels in the input image. While this may be desirable, e.g., to detect uncertainty about individual structures in an image or about the size of these structures, in practice it is often more useful to have a single number describe all of the confidence the network has in its prediction. For example, a hospital may implement the policy to have an additional (human) expert review all scans with an uncertainty value above a certain threshold.
There are several ways to estimate such an imagewise uncertainty: Firstly, it is straight forward to aggregate the individual, pixelwise uncertainties which all our methods output by taking the mean or maximum value, where we additionally considered taking the average of only the top 1% of values to put less emphasis on the predominant background pixels. However, this way of compressing information from single pixels into a global picture turned out to perform very badly in our initial experiments, so we did not explore it further.
However, for sample based methods (c.f. Section 3.2), there are two additional measures available, which have been described in [24] and use global information of agreement between the samples rather than focusing on individual pixels. For this, let $\hat{Y}^n$ denote the n-th predicted segmentation (i.e. the n-th sample):

- *inverted Dice agreement in samples (iDais)*: For any pair i, j of predictions, we calculate the Dice score of i, using j as ground truth, then average all of these:

$$1 - \frac{2}{N(N-1)} \sum_{i \neq j}^{N} AvgDice(\hat{Y}^i, \hat{Y}^j)$$

- *inverted IoU of samples (iIoU)*: For each class c, we take the intersection w.r.t. this class of all samples and divide its size by the size of the union of all samples, then average between all classes:

$$1 - \frac{1}{C} \sum_{c=1}^{C} \frac{|(\hat{Y}^1 = c) \cap (\hat{Y}^2 = c) ... \cap (\hat{Y}^N = c)|}{|(\hat{Y}^1 = c) \cup (\hat{Y}^2 = c) ... \cup (\hat{Y}^N = c)|}$$

In contrast to the other uncertainty measures used, the original formulations of these two actually signal *lower* uncertainty for higher values, so can be seen as a sort of *certainty*

metric. In order to be consistent among our metrics however, we choose to invert them (as the original formulations assume values between 0 and 1, so will our inverted versions), so that now higher values again reflect higher uncertainty.
We note that for the iDais, we actually calculate the Dice score with equal weights given to all classes, even the background, and similarly the mean of the IoU of samples for individual classes is also not biased. This differs from the way the Dice score is calculated to assess segmentation quality (c.f. Section 2.3) to better reflect that agreement about which regions are fluid-free should just as much reduce uncertainty as agreement on fluid regions.

4 Results and Discussion

With a variety of different methods, each with different options for how to compute pixelwise and imagewise uncertainties, this section aims to compare their performance in our setting and evaluate pixelwise as well as imagewise uncertainty estimation.
In order to compare different methods, we need a suitable measure for the quality of the uncertainty estimation. There are a number of choices to judge this performance by a single number, such as the *Brier score* (introduced in [6]), or the negative log likelihood of the predicted class (used e.g. in [21]), but we choose an arguably simpler approach that we consider to be more generally applicable to methods with different theoretical backgrounds: Following the intuition that scans on which the model performs poorly should also be assigned high uncertainty, while low uncertainty scans should almost always be correct, we judge the performance of the uncertainty estimation by the strength of the correlation between the predicted uncertainty and the quality of segmentation. For pixelwise uncertainties, we consider all pixels of all test images, group them by their predicted uncertainty values (we use bin sizes of 0.005, i.e., all values between 0.010 and 0.015 will be considered together) and then calculate the accuracy of this group. For imagewise uncertainty estimation, we simply use the imagewise uncertainty score $U(X)$ together with the Dice loss[9] for that image.
In either case, the performance of the uncertainty estimation is then reflected in a scatter plot of these two quantities, which in the case of perfect uncertainty estimation should cluster around a diagonal line, indicating perfect correlation.

4.1 Experimental Setup

In this section, we describe the setup and settings we used to run our experiments. For sampling-based methods, the number of samples we use is 30. This number of samples is decided upon based on an experiment detailed in Section 4.3

Monte Carlo Dropout The U-Net model is trained with a dropout rate of 0.2 and the dropout layers are kept activated during test time.

Test Time Augmentation From inspection of the training data and heuristic experiments, we perform the following data augmentations: Given a test image, we mirror it

[9] which we recall to be $1 - AvgDice$, calculated on a *weighted* version of the Dice Score (c.f. 2.3)

horizontally with a probability of 0.33 and rotate it by a degree of $\beta \sim Unif([-20, 20])°$. To further simplify this, we replace the random sampling by deterministically mirroring every third image and rotating in a fixed pattern.
Additionally, for each pixel x_i, we add $\epsilon \sim Unif[-0.1, 0.1]$ representing a uniform noise where $\forall x_i, 0.1 \times max(x_i) = 0.1$.

Deep Ensembles & Dropout Ensembles For the ensemble methods, we train 10 U-Net models according to Section 3.2.3. We restrict ourselves to 10 models due to the fact that new ensemble models have to be retrained from scratch which requires significant computational resources. Furthermore, [17] shows that there is diminishing benefit in adding further models to the ensemble.
For the final Dropout Ensemble method, we consider the full 10 models, but also show results for the case of 2 and 5 models in Section 4.3. For the final sample size, each ensemble member is sampled 3 times using Monte Carlo Dropout as described above to obtain 30 samples in total. In general, for a ensemble of M models, N total samples are generated by sampling every member $\lceil \frac{N}{M} \rceil$ times.

Loss Attenuation In contrast to [15], who use a Laplace distribution in their evaluation, we instead use a Normal distribution to parameterize the logit distributions. The reason for this are numerical instabilities we experienced when using a Laplace distribution. Further, we set the number of samples approximating the expectation in Equation (4) to 10 during training.

Direct Error Prediction For the *error prediction branch*, we use a single convolutional layer with kernel size 1×1 and a single output channel.

4.2 Segmentation Performance with Uncertainty Estimation

The performance of the model should not be negatively affected when applying uncertainty estimation. Table 2 shows the Dice score per fluid calculated per volume, their volumewise and scanwise mean on the OCT segmentation task. As we can see from the table, except for the case of Test Time Augmentation (TTA), the per-volume model performance is preserved or even improved when incorporating the uncertainty estimation methods. Further investigation should be done to understand why the TTA method has this behavior. It is interesting to see that when evaluating on a per image (B-Scan) basis, we see that TTA performs comparably to the other methods and that Direct Error Prediction gives results that are significantly better than the rest.
Also, it is important to notice that here the means per B-Scan values are higher as we do not skip all-background test samples. However, in the case of per-volume mean, we do skip empty B-Scans from the volumes to be comparable to the RETOUCH benchmark. This might also explain the differences in ordering amongst the methods.

4.3 Effect of Sample Size for Sample-based Methods

One of the most obvious parameters to tune is the sample size N for each of the sample-based methods of Section 3.2. There is a clear trade-off between performance in terms of

Table 2: Comparison of segmentation performance on OCT volumes after applying uncertainty estimation methods.

	Dice score				
Method	PED	SRF	IRF	Mean (Per Volume)	Mean (Per B-Scan)
Baseline	0.646	0.680	0.659	0.662	0.722
Monte Carlo Dropout	0.646	0.680	0.659	0.662	0.841
Ensemble	0.636	**0.694**	0.670	0.666	0.748
Dropout Ensemble	0.632	0.692	**0.671**	0.665	0.751
Test Time Augmentation	0.519	0.563	0.446	0.509	0.770
Loss Attenuation	0.651	0.663	0.635	0.650	0.722
Direct Error Prediction	**0.677**	0.685	0.656	**0.672**	**0.912**

the time it takes to generate the samples (i.e to evaluate the model multiple times) and the quality of uncertainty estimation.
One way to specify the number of samples needed to reliably quantify uncertainty is the following: If U_i, $U_j \in \mathbb{R}^{d\times d}$ are one of the types of uncertainty maps defined in Section 3, obtained from using i and j samples respectively, then we compute

$$E[|U_i - U_j|] := \frac{1}{d^2}\sum_{k,l}\Big|(U_i)_{kl} - (U_j)_{kl}\Big|$$

Once this value get small enough, we get the cut off value for N.
Figure 6 shows the results when using different uncertainty methods for our four types of uncertainty maps. From the graph, one can see that using more samples does not add much value at some point, as the change in the uncertainty maps becomes smaller and smaller. Based on this, we decided to have a cut-off at 30 samples as the change in the uncertainty maps becomes negligible afterwards, while time still increases linearly. We also note that Test Time Augmentation seems to converge less stably than the other methods, but also fix the sample size at 30 to be comparable.

4.4 Quality of Imagewise Uncertainty

As discussed in Section 3.5, it is often desirable to aggregate the $d\times d$ *pixelwise* uncertainty values that all methods produce into a single *imagewise* score. We explored the *inverse Dice agreement in samples (iDais)* and *inverse IoU per sample (iIoU)*, which unfortunately limit us to sample-based methods only. For all four of these, both measures are compared in Figure 7.
We see overall good performance for all combinations of measures and methods (recall again that good performance corresponds to high correlation, i.e. a tight diagonal line in the plots), but notice three horizontal clusters in the iIoU, which are most prominent in the case of Test Time augmentation, but occur in principle for all metods: These result from the fact that iIoU is a very strict metric, in the sense that the intersection is taken over all samples, so a fluid missing from a single prediction will have a heavy impact on

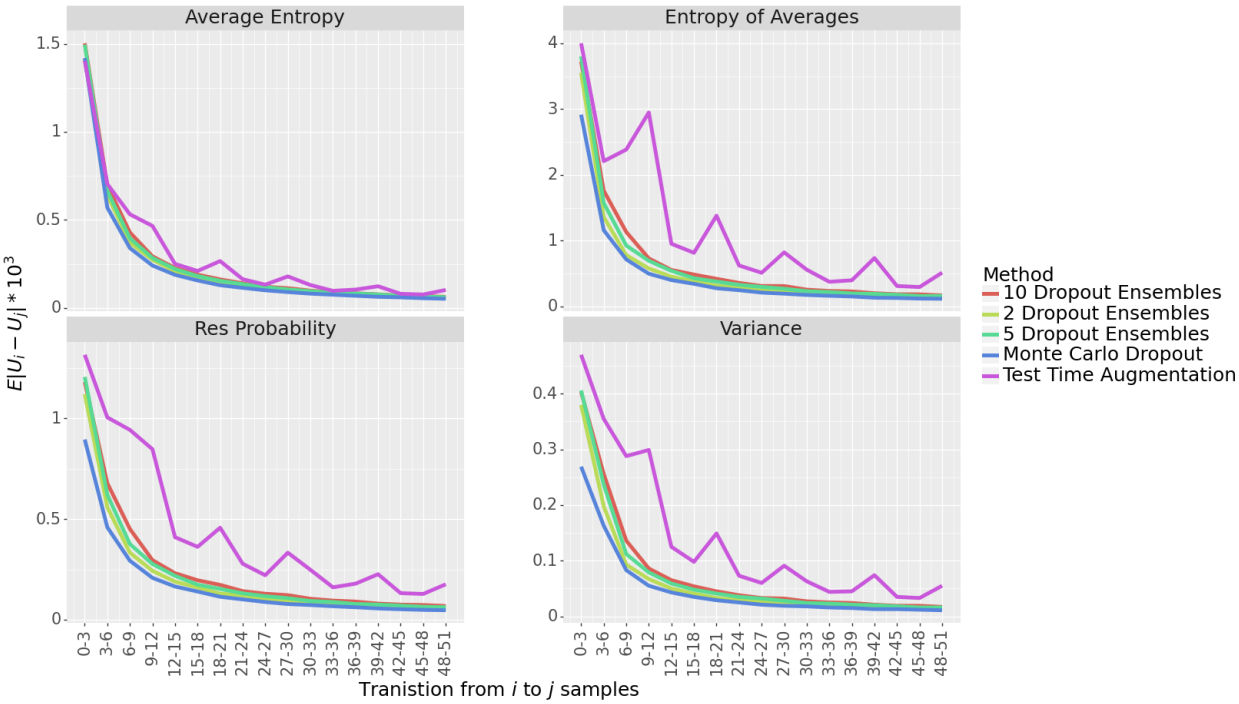

Figure 6: Mean absolute change in B-Scan Uncertainty Maps when switching from using i to j samples. We have 4 different methods to generate the uncertainty maps U_i and U_j. Each of the graphs here shows the $E|U_i - U_j| \times 10^3$ as we use add more samples to compute U_i and U_j

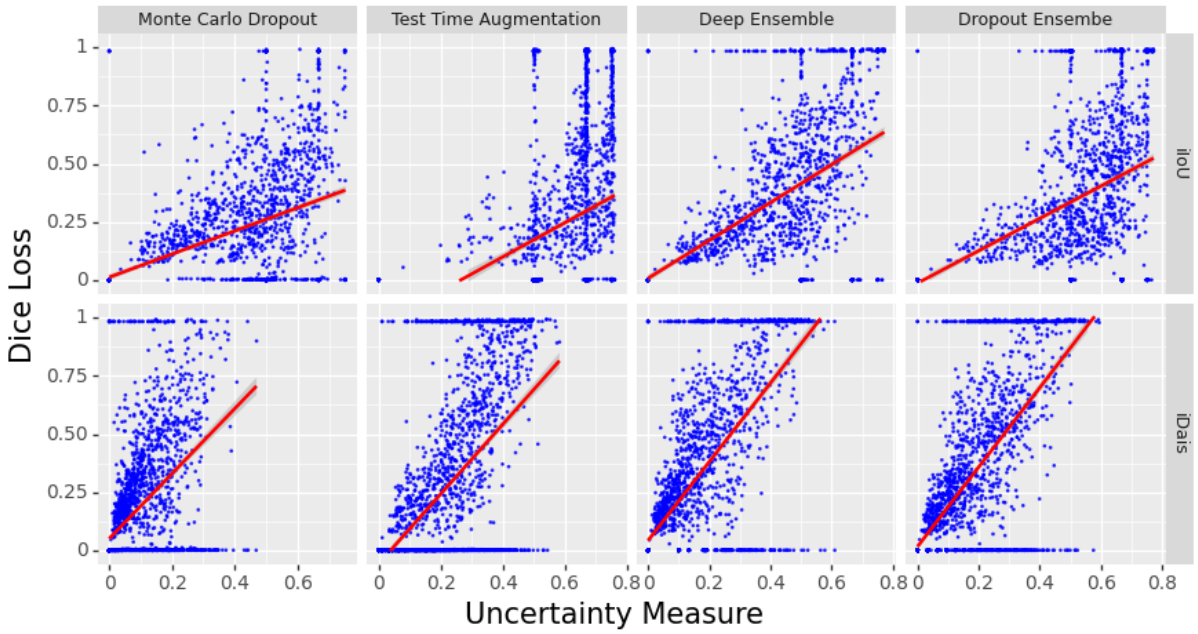

Figure 7: Performance of iDais and iIoU as a measure of imagewise uncertainty for all sample based methods. The horizontal and vertical clusters discussed in the text are clearly visible.

the score. The horizontal stripes reflect then cases where 1, 2 or 3 fluid types are missing from some samples, so the IoU for these categories will be 0, affecting the overall score. Similarly, iDais has some issues, as we can notice two horizontal artifacts at Dice loss values close to 0 and 1. These correspond to empty images that are predicted correctly in the first case and images with a small fluid patch present in either the ground truth or the prediction in the second. In both cases, if some samples predict fluid patches, but some do not, we will assign a medium uncertainty value, leading to these clusters in the graph. These cases are a inherent feature of our uncertainty prediction (sometimes, the model will be somewhat unsure even though it is completely correct or completely wrong), but they make it hard to compare the quality of uncertainty estimation from these plots alone.
However, an objective comparison can be made via *Confidence-Accuracy Curves*[10] which have been introduced in [17]. For these, we set different thresholds for the iDais and iIoU at $0.05, 0.1, 0.15, ..., 1.0$ (which is the highest possible value for both metrics) and calculate the average Dice loss for all images with an iDais lower or equal to the threshold. This is similar to a receiver operating characteristic (ROC) curve used to evaluate classification quality and mirrors what might happen in practice, where scans with a high estimated uncertainty might be redirected to a human expert for further judgment. However, unlike for a ROC curve, a random uncertainty classifier in this case would correspond to a straight line, as its provided uncertainty values would not carry any information in regard to image quality. Using these curves, we compare the four sample based methods in Figure 8, where we find that Test Time Augmentation and the two Ensemble methods show similar behaviour, while Monte Carlo Dropout shows less signs of a good discrimination, i.e. referring uncertain images to human epxerts does not improve performance in the same way it does for the other methods. We hypothesize that this reflects the fact that our data set is uniform[11], so aleatoric uncertainty plays a bigger role, and as Monte Carlo Dropout is designed to measure epistemic uncertainty, this explains the difference in behaviour.

4.5 Quality of Pixelwise Uncertainty

In a second part of our discussion, we are focusing on the performance of the uncertainty estimation methods when considering pixelwise predictions. For this, we follow a similar approach as in the imagewise discussion in Section 4.4, however, we show correlation plots on a pixelwise prediction scale. We recall the different measures we introduced in Section 3: For sample-based methods, we can calculate the *variance* among samples, the *average entropy*, *entropy of averages* or the *residual probability*. For the other three methods, we always can calculate the entropy[12] and residual entropy as well, but in case of Learned Loss Attenuation and Direct Error Prediction also have the predicted uncertainty the networks use during training available as an uncertainty measure at test time.

[10] We neither use confidence, nor accuracy, but rather iDais/iIoU and the Dice loss, but we still choose to keep the original name.

[11] While we do have three different devices, all of them are well represented in the data set, so none can be considered to be out-of-distribution.

[12] This corresponds to the entropy of averages in the sample-based case.

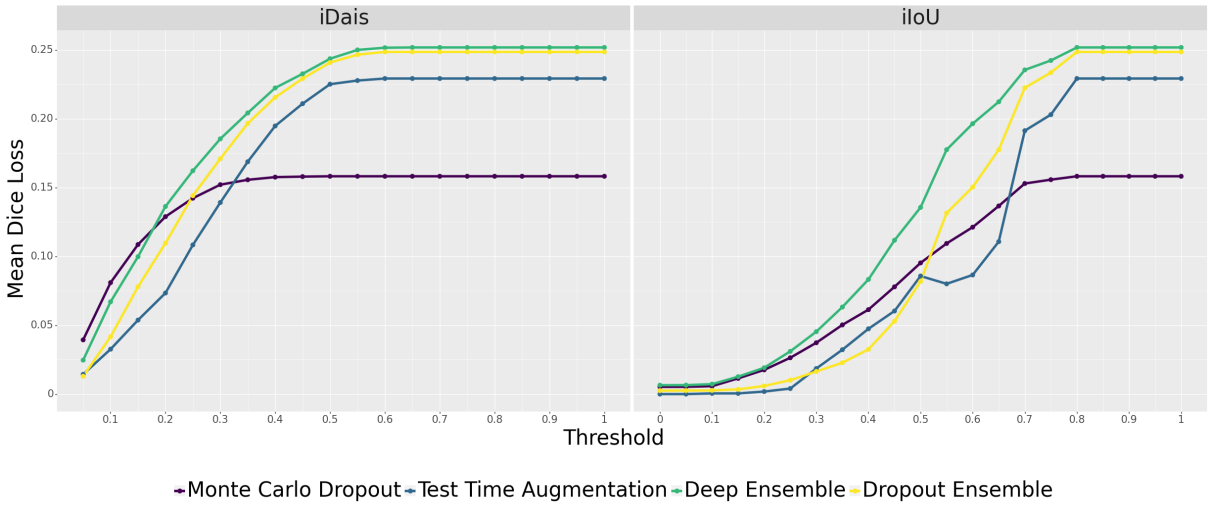

Figure 8: Confidence-Accuracy Curves for iDais/iIoU and all four sample based methods

4.5.1 Sample-based Methods

We first analyze the sample-based methods as they share the same set of uncertainty measures. Firstly, we show correlation plots for the sampled-based methods on the respective metrics in Figure 9. For a sound uncertainity estimation method we expect a strong correlation between the uncertainty and the performance of the model. Thus, we can directly compare the methods based on correlation which is shown in Table 3. Test Time Augmentation in general outperforms the other methods on this benchmark.

Table 3: Pearson's correlation coefficient of the respective plots in Figure 9

Correlation	MCD	Ensemble	Dropout Ensemble	TTA
Variance	0.888	0.693	0.515	**0.944**
Average Entropy	**0.755**	0.694	0.642	0.709
Entropy of Averages	0.713	0.498	0.725	**0.840**
Residual Probability	0.843	0.697	0.760	**0.918**

4.5.2 Baseline, Learned Loss Attenuation and Direct Error Prediction

For the three methods where no samples are generated, the variance and entropy across samples cannot be computed. However, they all are able to use the entropy and residual probability of the output distribution as uncertainty measure. Additionally, for Learned Loss Attenuation and Direct Error Prediction we can also use the uncertainty that is learned during training as an estimate at test time. Comparing the performance of these measures in Figure 10, we observe that the uncertainty predictions of the baseline are not very useful. While very low values indeed imply high accuracy[13] and a lot of errors are made for very high uncertainty values, a medium uncertainty holds almost no information

[13]This is very easy to achieve, as any method can just assign low uncertainty to background pixels, which will be correctly predicted most of the time

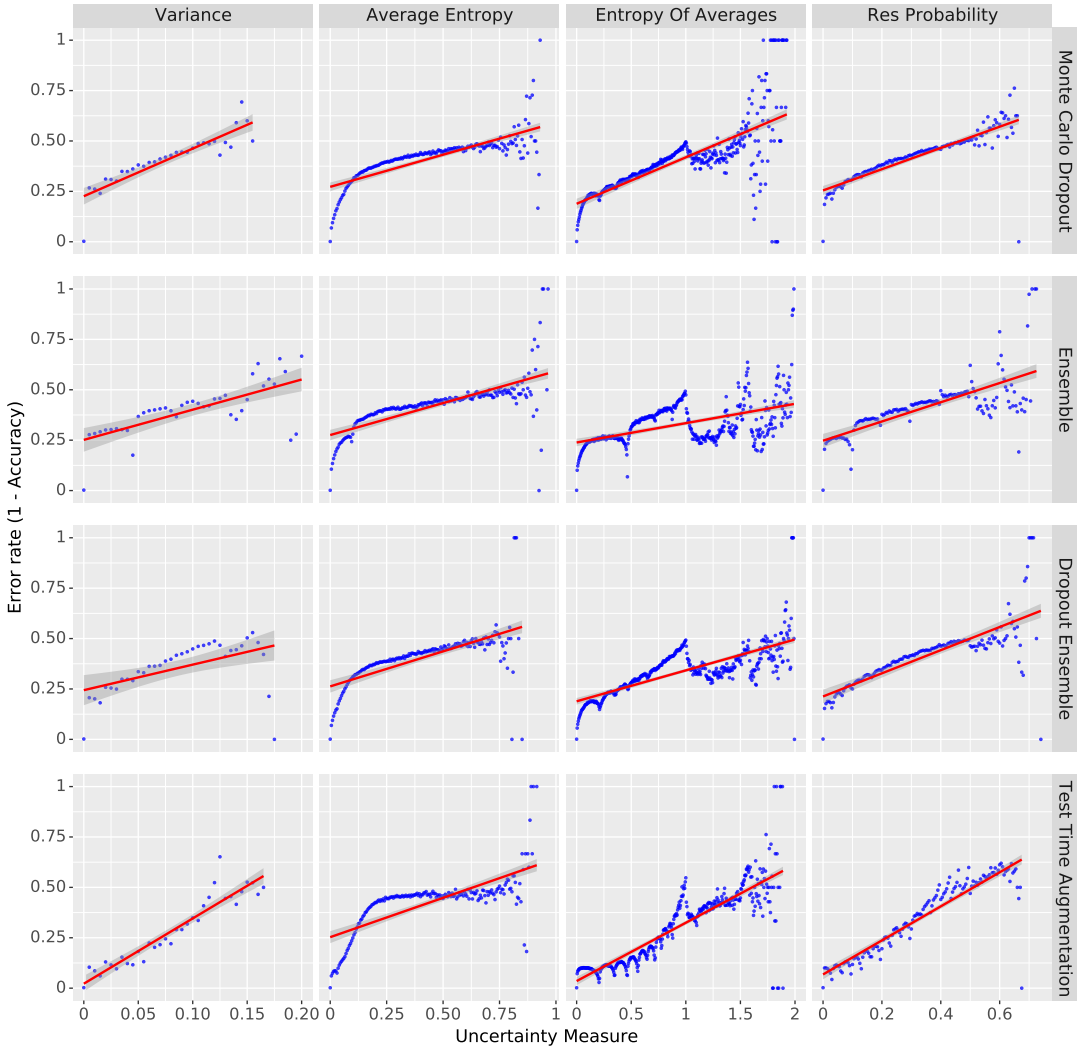

Figure 9: Performance of pixelwise uncertainty for all sample based methods. Each data point in the image corresponds to one uncertainty value and the average error rate among pixels with that value.

at all, which is reflected in the almost horizontal slope for each of the uncertainty measures. This does not change significantly when considering uncertainty in training, as can be seen from the minimal change in the behaviour of these curves for the other two methods. However, the predicted uncertainties in both cases show more useful shape, as these have more discriminatory value. They are also qualitatively very similar between these two methods, but on rather different absolute scales.

4.5.3 Calibration

Finally, we report the calibration of the output softmax distribution of each method. Intuitively, calibration measures that if a model makes a prediction with a certain probability, it should be correct in a corresponding percentage of cases, e.g., if it outputs 0.5 it should be correct in half of the cases. In other words, calibration is a measure of how well a model reports its own uncertainty. This relationship is usually captured in a calibration

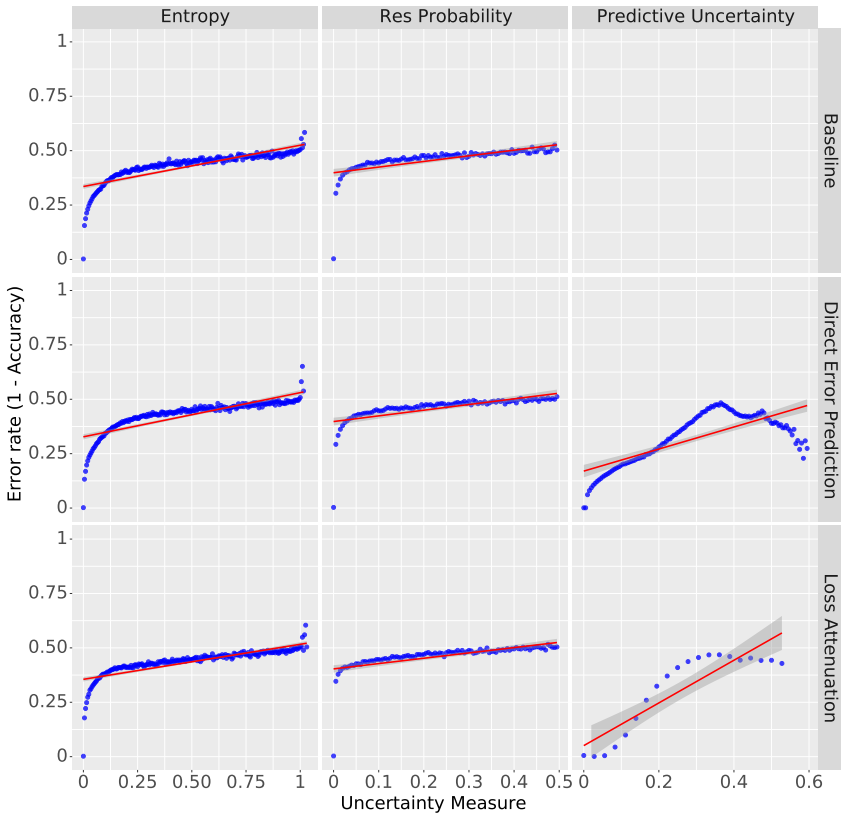

Figure 10: Performance of all non sample-based methods. Each data point in the image corresponds to one uncertainty value and the average error rate among pixels with that value. There is no predictive uncertainty for the baseline, as this is not part of its architecture.

plot where one plots output probability versus the frequency of the model being correct for that particular output probability. We construct this plot by binning the output probabilities for each fluid class and pixel and then computing the frequency of the model being correct in each bin. The resulting plot is shown in Figure 11 where the black line $x = y$ corresponds to a perfectly calibrated model. Based on this, we further report the calibration error computing the absolute difference between the probability and frequency of each bin where lower values correspond to better calibration in the caption. We observe that TTA outperforms all other methods by a large margin, while the other sample-based methods, i.e., both Ensemble methods and Monte Carlo Dropout, perform comparably, with Dropout Ensembles having an edge since it obtains additional samples beyond the 10 samples of the standard Deep Ensemble. Both considered non-sample based methods show similar calibration to the Baseline method aligning with the results in Figure 10 on the pixelwise entropy and residual probability, however they still benefit of their additional methods to obtain uncertainty estimates through their model-specific prediction heads whose training does not seem to influence calibration of the output distribution at all.

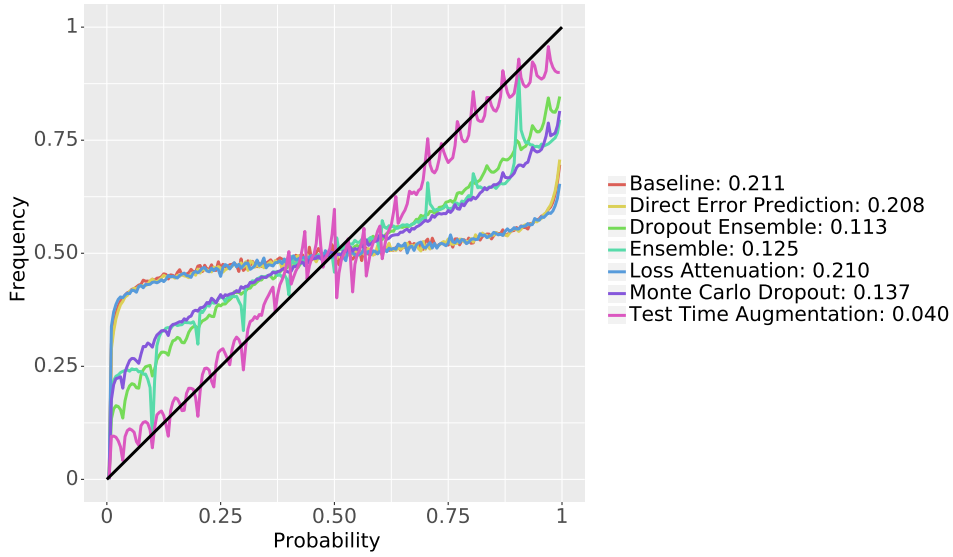

Figure 11: Calibration curve for all considered methods. Perfect calibration corresponds to the black line $x = y$. Calibration errors for each method are reported in the legend.

5 Conclusion

In this project, we explored methods for uncertainty estimation in the context of medical image segmentation of pathological fluids within the retina based on the ReTouch dataset. We conducted experiments and hyperparameter tuning to obtain a baseline segmentation model based on the U-Net architecture and to make the consecutive evaluation of the uncertainty estimation approaches as comparable as possible. We implemented several baseline methods for the task of epistemic as well as aleatoric uncertainty quantification and added more recent approaches and our own proposal afterwards. While we have seen that sample based methods such as Monte Carlo Dropout, Test Time Augmentation and (Dropout) Ensembles outperform the more direct approaches in our experiments, we still see a lot of potential for these, e.g. by deriving loss functions that are better suited to the problem at hand. Altogether, we have provided a extensive study of the problem of uncertainty estimation for optical coherence tomography and considered a wide range of common methods. As with any field as vast as this, there is a large number of further research to be done: The literature still hold a number of methods we have not examined, not to mention all the ways in which the methods we have described may be improved. Also, considering the special setting of OCR scans, one might think about additions to our distinction of imagewise and pixelwise uncertainty: One could e.g. calculate a single number for every connected fluid region, or, on the other hand, try to predict the uncertainty for each class of fluid type separately. However, we hope to have provided the suitable foundation and framework for these and other questions and are eager to see them tackled.

References

[1] Jennifer J Arnold et al. "The role of sub-retinal fluid in determining treatment outcomes in patients with neovascular age-related macular degeneration-a phase IV randomised clinical trial with ranibizumab: the FLUID study". In: *BMC ophthalmology* 16.1 (2016), pp. 1–9.

[2] Murat Seckin Ayhan and Philipp Berens. "Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks". In: (2018).

[3] Simon Bachstein. "Uncertainty Quantification in Deep Learning". MA thesis. 2019.

[4] Charles Blundell et al. "Weight uncertainty in neural networks". In: *arXiv preprint arXiv:1505.05424* (2015).

[5] Hrvoje Bogunović et al. "RETOUCH: The retinal OCT fluid detection and segmentation benchmark and challenge". In: *IEEE transactions on medical imaging* 38.8 (2019), pp. 1858–1874.

[6] Glenn W Brier. "Verification of forecasts expressed in terms of probability". In: *Monthly weather review* 78.1 (1950), pp. 1–3.

[7] Errol W Chan et al. "Quantitative changes in pigment epithelial detachment area and volume predict retreatment in polypoidal choroidal vasculopathy". In: *American Journal of Ophthalmology* 177 (2017), pp. 195–205.

[8] Thomas G Dietterich. "Ensemble methods in machine learning". In: *International workshop on multiple classifier systems*. Springer. 2000, pp. 1–15.

[9] Yarin Gal and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In: *international conference on machine learning*. 2016, pp. 1050–1059.

[10] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[11] David Huang et al. "Optical coherence tomography". In: *science* 254.5035 (1991), pp. 1178–1181.

[12] Huimin Huang et al. "UNet 3+: A Full-Scale Connected UNet for Medical Image Segmentation". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 1055–1059.

[13] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, pp. 448–456.

[14] Saumya Jetley et al. "Learn to pay attention". In: *arXiv preprint arXiv:1804.02391* (2018).

[15] Alex Kendall and Yarin Gal. "What uncertainties do we need in bayesian deep learning for computer vision?" In: *Advances in neural information processing systems*. 2017, pp. 5574–5584.

[16] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[17] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles". In: *Advances in neural information processing systems*. 2017, pp. 6402–6413.

[18] Xiaomeng Li et al. "H-DenseUNet: hybrid densely connected UNet for liver and tumor segmentation from CT volumes". In: *IEEE transactions on medical imaging* 37.12 (2018), pp. 2663–2674.

[19] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[20] Donghuan Lu et al. "Retinal fluid segmentation and detection in optical coherence tomography images using fully convolutional neural network". In: *arXiv preprint arXiv:1710.04778* (2017).

[21] Yaniv Ovadia et al. "Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift". In: *Advances in Neural Information Processing Systems*. 2019, pp. 13991–14002.

[22] Fernando M Penha et al. "Quantitative changes in retinal pigment epithelial detachments as a predictor for retreatment with anti-VEGF therapy". In: *Retina* 33.3 (2013), pp. 459–466.

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[24] Abhijit Guha Roy et al. "Bayesian QuickNAT: model uncertainty in deep whole-brain segmentation for structure-wise quality control". In: *NeuroImage* 195 (2019), pp. 11–22.

[25] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 1929–1958. ISSN: 1532-4435.

[26] Guotai Wang et al. "Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks". In: *Neurocomputing* 338 (2019), pp. 34–45.

Appendix

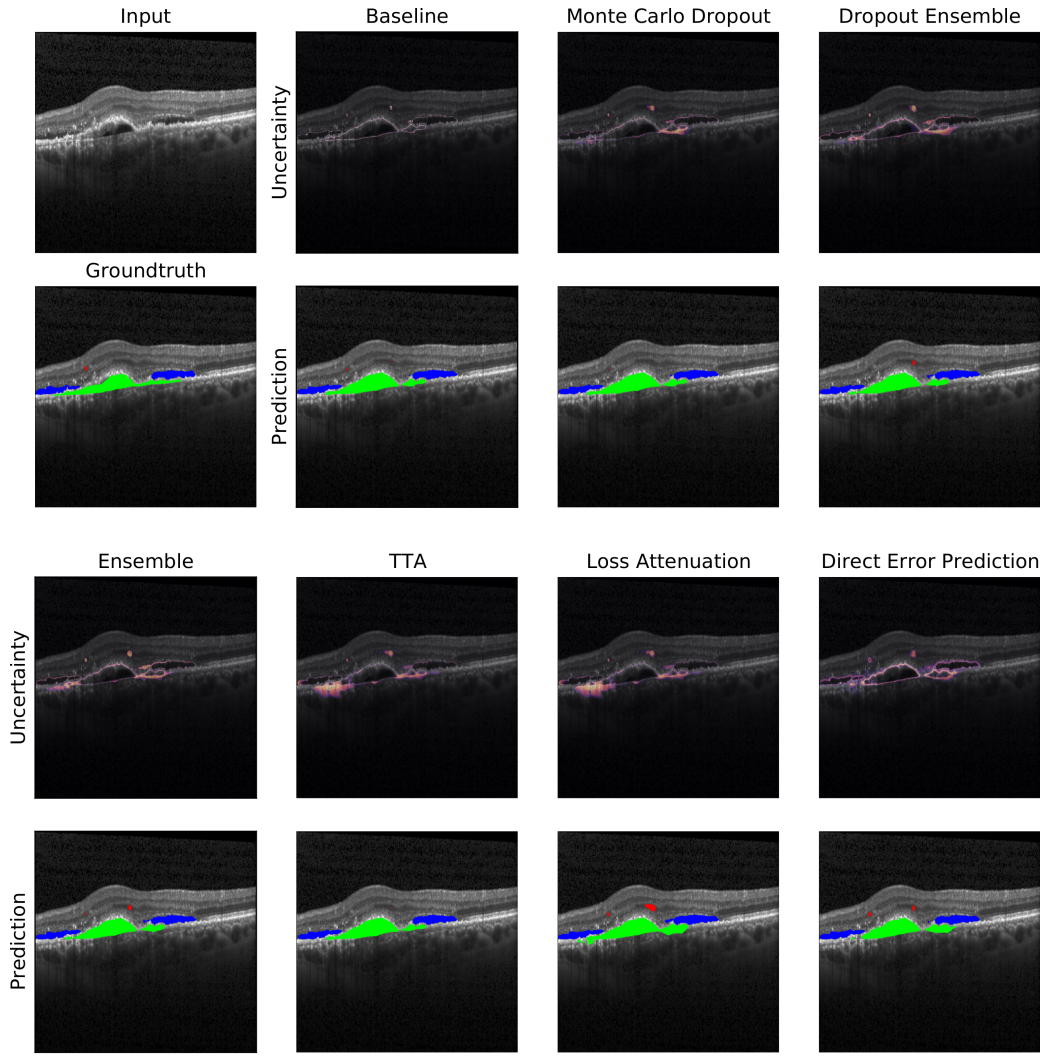

Figure 12: Visual comparison of all proposed methods. The uncertainty estimates are in the first row and the fluid prediction in the second row. The first column shows the input B-scan to the network and the respective groundtruth fluid annotation. We superimpose the input B-scan below uncertainty estimates and predictions for visual orientation. We note in particular that the baseline approach simply traces the outlines of the prediction as its uncertainty estimation, which is only of limited use in practice, justifying the need for the more sophisticated methods.

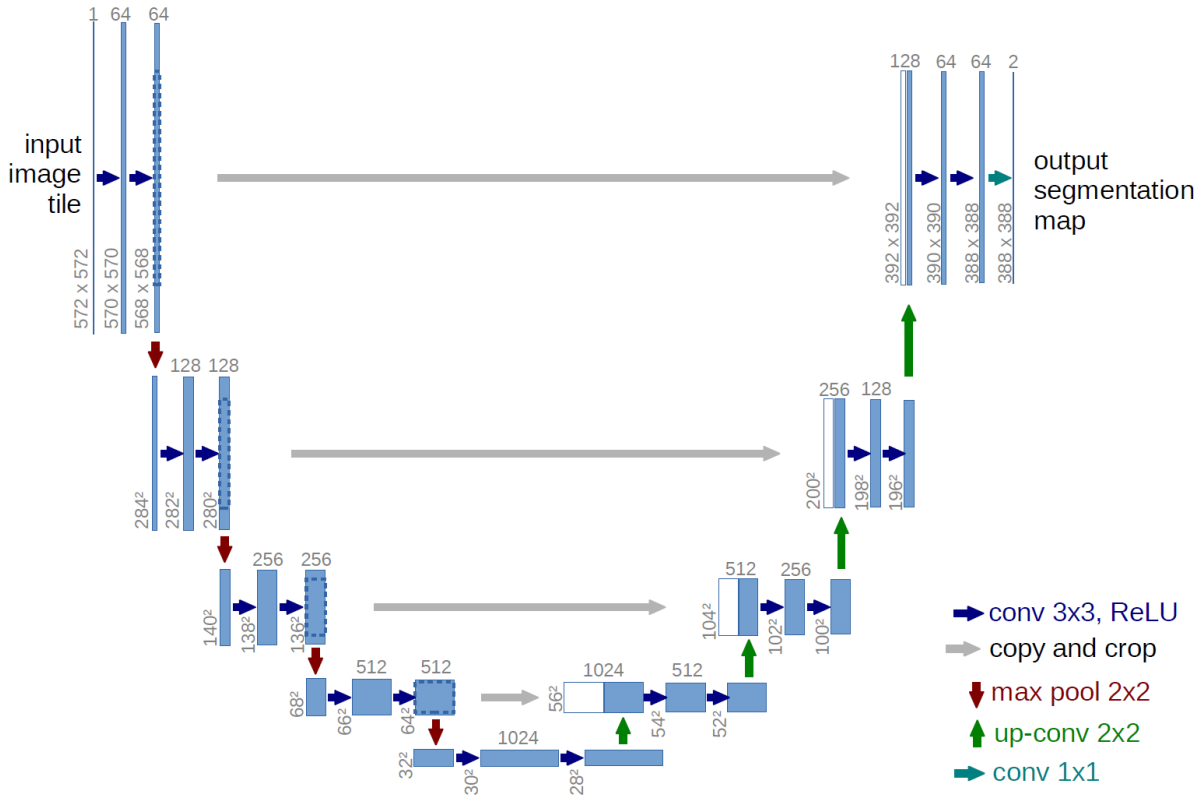

Figure 13: U-Net Architecture (Source: Ronneberger, Fischer, and Brox [23])