Balancing Calibration and Performance: Stochastic Depth in Segmentation BNNs – Supplementary Material

Linghong Yao
leon.yao.18@alumni.ucl.ac.uk
Denis Hadjivelichkov
dennis.hadjivelichkov@ucl.ac.uk
Andromachi Maria Delfaki
a.delfaki@ucl.ac.uk
Yuanchang Liu
yuanchang.liu@ucl.ac.uk
Brooks Paige
b.paige@ucl.ac.uk
Dimitrios Kanoulas
d.kanoulas@ucl.ac.uk

Department of Computer Science &
Mechanical Engineering
University College London
Gower Street, WC1E 6BT
London, UK

2 Background and Related Work

2.1 Sources of Uncertainty and Calibration

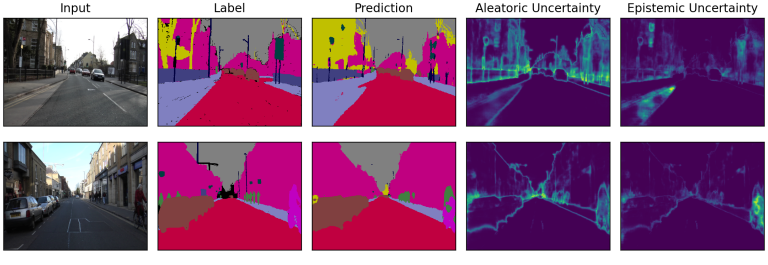


Figure 1: A Bayesian neural network predicts that aleatoric uncertainty is generally greater at object edges and boundaries due to limited pixel resolution, while epistemic uncertainty is more prevalent within objects, relating to uncertainties in classifying the object itself.

3 Methodology

3.1 Efficient Network Design

Methods	mIOU	Params (d) ↓	Params (t) ↓	GFLOPs ↓	T (ms) ↓	Mem
Regular	63.91	1.52	3.34	2.26	8.03	103
Depthwise	64.60	**0.19**	**2.00**	**1.38**	**7.88**	**124**
Inv-Res-0.75	65.69	1.04	**2.06**	1.56	10.54	167
Inv-Res-Bl	**66.61**	1.86	3.67	2.24	11.47	178

Table 1: Different decoder designs. Params (d): number of parameters for decoders in millions, Params (t): total number of parameters of the network in millions, T: inference time measured in ms on GPU, Mem: memory usage measured in MB on GPU.

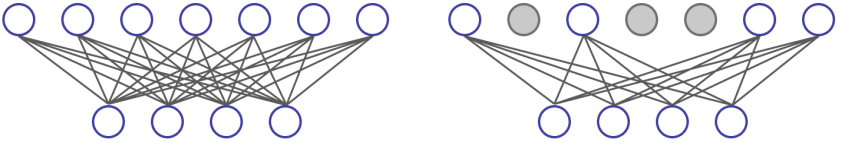
We evaluate the performance of the network using mIOU, and we quantify the efficiency of the network using the number of parameters, FLOPS, runtime, and memory requirement. Training details follow the same recipe that is described in Section **??**.

Table 1 shows the results of the preliminary experiment. First, we see that the depthwise decoder does in fact reduce the number of parameters in the decoder by about 8-fold, and achieves the fastest inference speed out of all three experiments. Furthermore, it does not hurt performance compared to regular decoders. Our results show that the inverted residual design achieves the highest mIOU of 66.61, but uses most parameters out of all three experiments.

For fair comparison to the depthwise decoder, we reduce the number of total parameters of *inv-Res* via width and depth scaling by a factor of 0.75 (*Inv-Res-0.75*) on both the encoder and the decoder to obtain around 2M parameters. The scaled-down version of the network has an encoder that is strictly worse than the regular encoder, so one would expect performance to decrease. However, we show that even with the same number of parameters, the inverted residual design still outperforms the depthwise decoder. This suggests that the inverted residual decoder is able to recover a significant portion of the information lost in the decoder, and the performance-to-parameter trade-off is better.

3.2 Bayesian Inference with Stochastic Depth

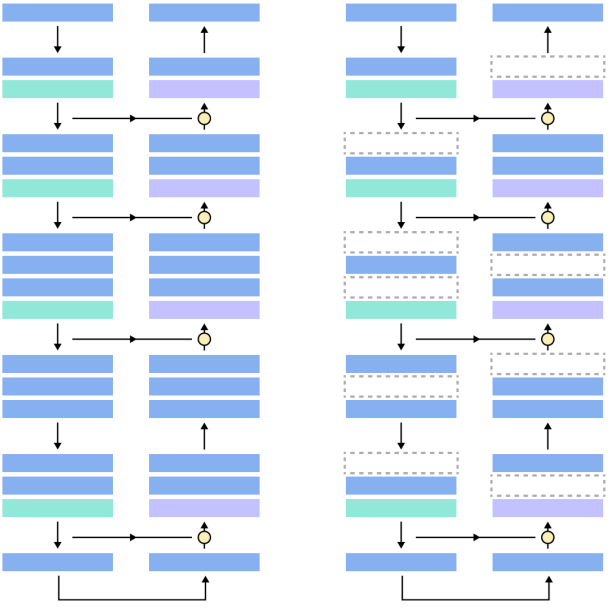
3.2.1 MC Dropout



(a) (b)

Figure 2: Example illustration of how dropout layers work with two layers of units. a) shows a fully connected layer with all units activated, b) shows the same layers with only 50% of the units activated.

3.2.2 Stochastic Depth



(a) Baseline. (b) Baseline with stochastic depth.

Figure 3: Example illustration of how stochastic depth could be applied to the baseline network. a) shows the network with all blocks activated, and b) shows an instance of a shallower network with stochastic depth, where blocks with dashed lines indicate blocks that are turned off (i.e., bypassed with identity functions).

3.3 Metrics: Network Performance

We evaluate the network's performance mainly by mIOU, but we also monitor global and average accuracy. We define these metrics below.

3.3.1 Accuracy (global)

Given predicted pixel $\hat{y}_i$ and corresponding ground truth label y_i, we define global accuracy as

$$\text{Accuracy(g)} = \frac{1}{N}\sum_{i}^{N} I[\hat{y}_i = y_i] \tag{1}$$

where N defines the total number of pixels. This is usually denoted as *Acc(g)* in our tables.

3.3.2 Accuracy (average)

A downside of using global accuracy is that it doesn't take the frequency of each class into account, which means that the metric is biased towards classes with larger areas. We further

define per-class or average accuracy, which is not influenced by the frequency of each class:

$$\text{Accuracy(c)} = \frac{1}{C}\sum_{c}\frac{1}{\sum_{i}^{N} I[y_i = c]}\sum_{i}^{N} I[\hat{y}_i = y_i] \tag{2}$$

where C is the total number of classes. This is usually denoted as *Acc(c)* in our tables.

3.3.3 Intersect-over-union (IOU)

Let C be the total number of classes, we further define per-class true positives N_c^{TP}, false positives N_c^{FP}, and false negatives N_c^{FN} as:

$$N_c^{TP} = \sum_{i,c} I[\hat{y}_i = y_i | y_i = c]$$
$$N_c^{FP} = \sum_{i,c} I[\hat{y}_i \neq y_i | \hat{y}_i = c]$$
$$N_c^{FN} = \sum_{i,c} I[\hat{y}_i \neq y_i | y_i = c]$$

We define per-class IOU as the ratio between the intersect $(y \wedge \hat{y})$ and the union $(y \vee \hat{y})$, which in practice would be computed as:

$$\text{IOU}_c = \frac{N_c^{TP}}{N_c^{TP} + N_c^{FP} + N_c^{FN}} \tag{3}$$

and mean IOU as:

$$\text{mIOU} = \frac{1}{C}\sum_{c}\frac{N_c^{TP}}{N_c^{TP} + N_c^{FP} + N_c^{FN}} \tag{4}$$

The mIOU metric is a suitable and commonly used metric in semantic segmentation, it measures the degree of "similarity" between the ground truth and prediction and puts the same weight on each class even if classes are imbalanced in the dataset.

3.3.4 Other metrics

We also sometimes monitor the speed and cost of the networks wherever we need to compare efficiency between networks. In particular, we measure the number of parameters of the network and the number of FLOPs to perform one inference. Furthermore, we track the GPU inference time and memory time averaged over 5 forward passes. These metrics help us better evaluate the efficiency of the network.

3.4 Metrics: Uncertainty Calibration

3.4.1 Calibration Error

Two natural metrics arise from the definition of calibration. The first is Expected Calibration Error (ECE) [12]:

$$E\left[\left|P\left(\hat{Y} = Y | \hat{P} = P\right)\right|\right]$$

It's clear that an ECE value of 0 indicates perfect calibration. Another metric commonly used is the Maximum Calibration Error (MCE) [12], defined as the maximum difference between the confidence and the true accuracy:

$$\max |P\left(\hat{Y} = Y|\hat{P} = P\right) - p|$$

In practice, we can compute the ECE and MCE using discrete bins described in Eq. 2 and 3.

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)| \tag{5}$$

$$\text{MCE} = \max_{m} |\text{acc}(B_m) - \text{conf}(B_m)| \tag{6}$$

3.4.2 PAvPU

In the context of computer vision tasks such as segmentation, global statistics such as ECE and MCE are not able to capture the local quality of the segmentation. Mukhoti and Gal [11] introduced a way of evaluating uncertainty quality for computer vision and has been adopted by several other studies [9, 10]. The central idea of the metric is based on the statement that a well-calibrated model should be accurate (a) when it's confident (c), and unconfident (u) when it's inaccurate (i).

Let the normalized confidence value $I \in [0,1]$ be the threshold of confident/unconfident. We define n_{ac} as the number of pixels that are predicted correctly and confidently, n_{ic} be the number of pixels that are predicted inaccurately but confidently, n_{au} be the number of pixels that are predicted accurately but not confidently, and n_{iu} be the number of pixels that are predicted inaccurately and unconfidently. Mukhoti and Gal [11] further proposes that accuracy and certainty be performed over patches of pixels by using sliding windows. We define the accurate-certain ratio as:

$$p(\text{accurate}|\text{certain}) = \frac{n_{ac}}{n_{ac} + n_{ic}} \tag{7}$$

Similarly, we define the inaccurate-uncertain ratio as:

$$p(\text{uncertain}|\text{inaccurate}) = \frac{n_{iu}}{n_{iu} + n_{ic}} \tag{8}$$

Lastly, we define the uncertainty accuracy (UA), to the patch accuracy vs. patch uncertainty (PAvPU) as:

$$\text{PAvPU} = \frac{n_{ac} + n_{iu}}{n_{ac} + n_{iu} + n_{au} + n_{ic}} \tag{9}$$

Clearly, networks with larger values of the above metrics are better. These metrics can be evaluated with a particular uncertainty threshold I [11], but can also be evaluated by sliding the uncertainty threshold from $[0,1]$ and computing the area under the curve [9, 10]. In this paper, we compute the area under the curve as it's a more robust way that doesn't require manual tuning of the uncertainty threshold. The normalized uncertainty value is computed using $I_{\text{norm}} = \frac{I}{I_{\max} - I_{\min}}$, with the minimum and maximum values computed over a validation set.

4 Evaluation

4.0 Dataset

The Cambridge-driving labelled Video Database (CamVid) is a roadscene dataset captured from the perspective of a driving automobile [2]. All images are semantically labelled with 11 classes such as cars, pedestrians, bicyclists, and poles. Images are captured at a resolution of 960×720, with 367, 101, and 233 images in training, validation, and testing sets respectively. Following [1], we downsize the images to 480×360 and drop ambiguous pixels labeled as "void" for training and evaluation.

Figure 4: Example labelled image from the CamVid Dataset (test set)

4.0.1 Image Augementation

We follow a similar image augmentation procedure as [5, 8], with random horizontal flipping, followed by random color jittering, and random cropping from scale $[0.7, 1.3]$. Finally, the images are resized to 360×480 and normalized. Figure 5 demonstrates two examples of images going through the augmentation pipeline. The purpose of these image augmentation steps is to help reduce the overfitting of the network on the training set. We also note that all "void" pixels are ignored in both loss functions and metrics, therefore we use the same pixel value for filling in the blank space when images are scaled down (e.g. second row in the figure).

Figure 5: Example of two images going through the augmentation process step by step, the images are randomly flipped, jittered, and cropped. Note that the label and the image go through the same augmentation but with random parameters each time. Image augmentation is computed in parallel to the main training loop

4.0.2 Hyperparameters

All networks are trained and tested on Google Colab with an Nvidia T4 GPU with 15GB of RAM, and the networks are built and trained using PyTorch. Unless explicitly stated, all experiments in this paper utilize the same hyperparameters in the training process. This is so that we can make fair comparisons across different runs. We utilize a batch size of 10 images, which was empirically determined to have relatively stable gradients but not using too much memory during training. We train all experiments for 200 epochs, or equivalent to around 7400 steps. Similar to [3], we use RMSProp with an initial learning rate of $1e-3$ with a polynomial learning rate scheduler that updates the learning rate using $\left(1-\frac{iter}{max\_iter}\right)^{power}$ with power 0.9. We use a weight decay of $1e-4$ to help with regularization, and unless stated otherwise, we use dropout layers after each stage, with dropout probability linearly decaying from 0.1 to 0 from the deepest to the shallowest layers on both encoders and decoders. Finally, we use cross-entropy loss with no class balancing.

It's important to note that all networks trained with 200 epochs on a polynomial learning rate schedule are *underfitted*. An example of training history is shown in Figure 6, where it's clear that the network can still improve further. We make this choice simply due to limited time and computational resources, and we argue that the performance of the network at the same cut-off epoch is indicative of how well it will perform with more epochs. This also implies that the performance of the network at the end of training is typically lower than what it should be if more epochs are trained. We supply further experiments that use exponential learning rate and patience 100 in Section 4.1.2 of Main Paper. Additionally, due to different network sizes, larger networks will be more underfitted than smaller networks.

Finally, we also point out that many papers utilize three additional "tricks" to improve results: a) pre-training on cityscape and/or ImageNet, b) fine-tuning with full-resolution images, and c) multi-scale evaluation. In this paper, we do not perform any of these techniques to further improve results, all of our experiments are trained and tested on half-resolution images only from the CamVid dataset.

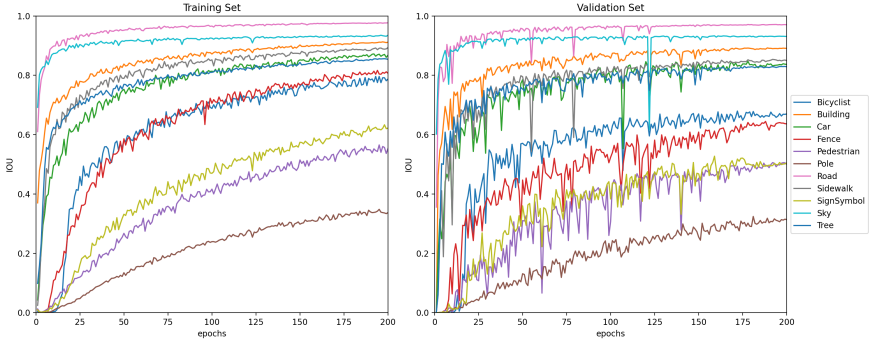


Figure 6: The performance history of the baseline network on the training and validation set. There tends to be a larger gap between the training and validation set on classes with high variabilities in appearances, such as cyclists, sign symbols, and fences. Note that performance on both training and validation sets has not plateaued for most classes at the end of 200 epochs.

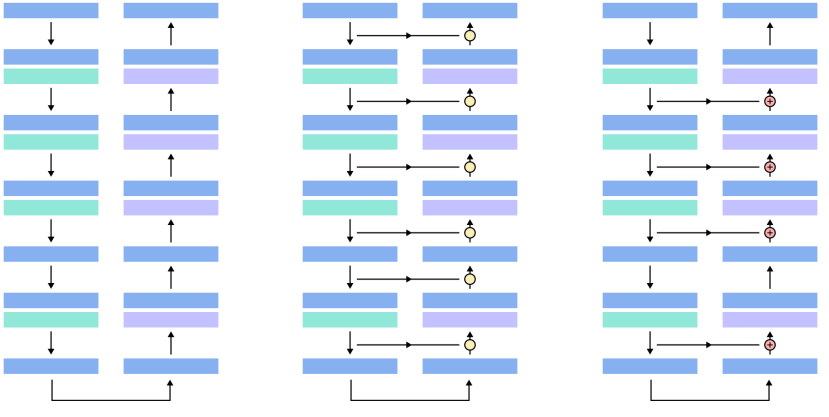
4.1 Efficient Designs

4.1.1 Modifications

No skip connections This is a simple ablation test to see how effective skip connections are. As it is commonly known, skip connections help transfer high-resolution features and usually help with thinner classes. By removing them, we expect the network to underperform in classes with fine boundaries or small areas, such as fences and posts. In the tables below we refer to this experiment as *no-skip*, and an illustration of the experiment setup is shown in Figure 7a.

Dense connections With similar motivation as no skip connections, we want to see how much can extra skip connections help with performance, and how much slower inference becomes with the presence of more skip connections. Note that this experiment only differs a little bit from the baseline experiment, with additional skip connections on the first and fifth stages. We refer to this experiment as *dense-skip*, and it's illustrated in Figure 7b.

Residual connections In this variation, we add the features from the encoder to the decoder rather than concatenating it. This is motivated by the residual block [4], where skip connections in the form of addition can help with training networks across large depths. Since our network is symmetrical, the output feature of each encoder block shares the exact same shape as a corresponding output feature of a decoder block, making feature addition possible. We refer to this experiment as *add-skip*, and it's illustrated in Fig. 7c.



(a) no-skip (b) dense-skip (c) add-skip

Figure 7: Modifications to skip connections between the encoder and corresponding decoder blocks. (a) shows the baseline network with all skip connections turned off, (b) with all skip connections turned on, and (c) uses addition rather than concatenation to transfer features

Squeeze and Excitation We adopt the squeeze and excitation block [6] in similar fashion to the MBConv block used in EfficientNet [15]. The squeeze and excitation block is inserted after the inverted residual expansion block. The idea of squeeze and excitation block is

that features can be used more effectively through attention mechanisms, and that we don't require a significantly higher parameter count to improve performance.

Shallow decoder In this variation, we use depth = 1 in the decoder of the network. The idea of this variation is motivated by high performing segmentation architectures that use relatively light-weight decoders [3, 8], and the intuition is that the encoder architecture that reduces feature resolution whilst increases feature depth is an effective enough structure on its own. In other words, this experiment hypothesizes that decoder are closer to an implementation detail that helps to transform the encoder features into desired shapes. We abbreviate this experiment as *shallow-dec* in tables.

Method	Params ↓	GFLOPs ↓	Time(ms) ↓	Acc(g) ↑	Acc(c) ↑	mIOU ↑
baseline	3.67M	2.24	11.47	92.09	75.22	66.61
shallow-dec	**2.59M**	**1.68**	**8.99**	92.01	74.65	66.47
no-skip	3.63M	2.21	11.38	90.28	66.96	58.51
add-skip	3.63M	2.21	11.44	91.97	74.18	65.95
dense-skip	3.77M	2.27	11.48	92.03	**75.97**	67.17
squeeze-exc	4.56M	2.27	13.97	**92.39**	75.46	**67.70**

Table 2: Qualitative performance of network modifications.

Method	Bicyclist	Building	Car	Fence	Pedestrian	Pole	Road	Sidewalk	SignSymbol	Sky	Tree	Acc(g)	Acc(c)	mIOU
baseline	53.7	84.8	77.6	56.8	41.8	**28.4**	95.5	81.7	40.8	93.2	78.3	92.1	75.2	66.6
shallow	55.2	84.4	78.4	55.4	41.7	26.8	95.6	**81.9**	40.9	93.1	77.7	92.0	74.7	66.5
no-skip	47.6	82.7	75.0	50.6	27.2	1.8	94.1	77.2	21.9	90.7	74.9	90.3	67.0	58.5
add-skip	54.2	84.6	78.0	55.9	41.3	24.8	95.5	81.4	38.7	93.2	77.9	92.0	74.2	66.0
dense-skip	56.2	84.9	77.6	57.0	**45.8**	28.1	95.3	80.9	41.9	93.2	77.8	92.0	**76.0**	67.2
s-e	**57.8**	**85.5**	**79.1**	**58.3**	44.4	27.9	**95.6**	81.9	**42.4**	**93.3**	**78.6**	**92.4**	75.5	**67.7**

Table 3: Per-class metric breakdowns for experiments with modifications to skip connections.

4.1.2 Network Scaling

Width	Depth	Params ↓	GFLOPs ↓	Time(ms) ↓	Acc(g) ↑	Acc(c) ↑	mIOU ↑
0.5	0.5	0.67M	0.49	4.13	90.28	67.82	59.14
0.5	1.0	0.97M	0.73	6.98	90.70	69.51	60.89
1.0	0.5	2.53M	1.54	4.60	91.87	74.01	65.58
1.0	1.0	3.67M	2.24	11.47	92.09	75.22	66.61
1.0	1.5	6.98M	3.74	17.58	92.25	75.39	67.14
1.5	1.0	8.13M	4.90	14.80	92.60	78.29	69.59

Table 4: Quantitative performance of network scaling.

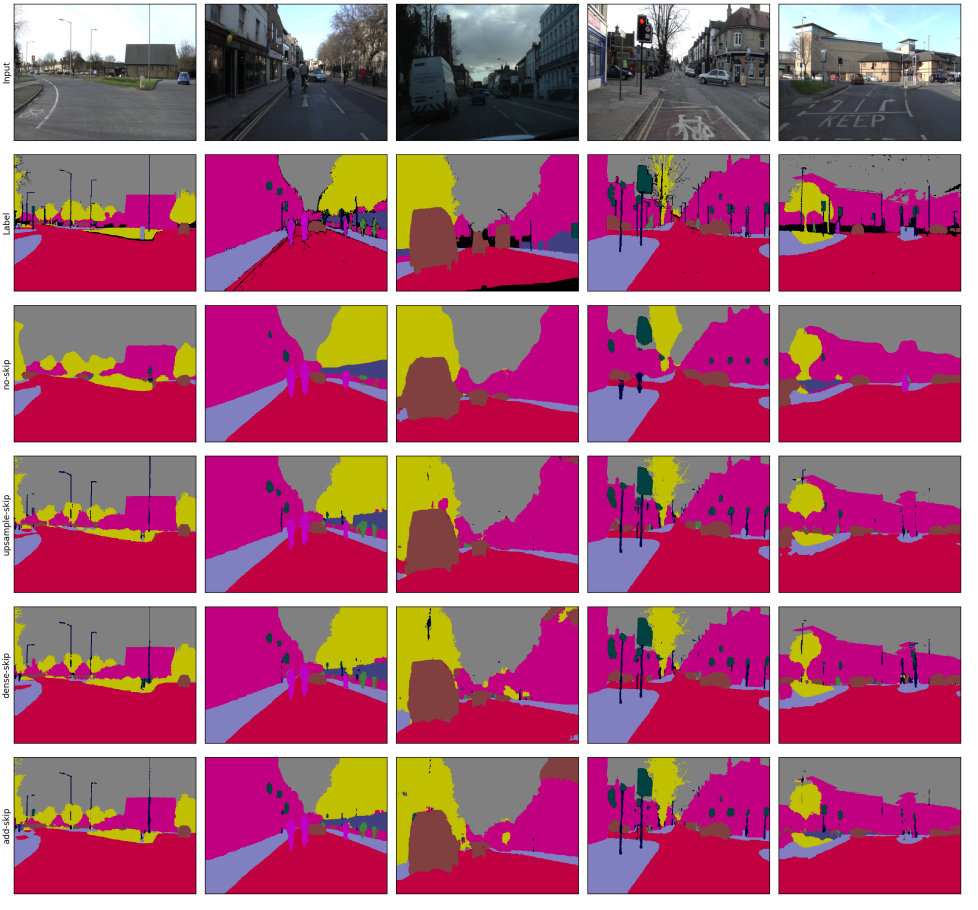


Figure 8: Qualitative results of modifications to skip connections. We highlight that *no-skip* produces very coarse results with smooth boundaries, as it's unable to recover the spatial resolution without the skip connections. On the opposite spectrum, *dense-skip* is able to better predict small and thin classes such as posts and signs.

4.1.3 Comparison to State-of-the-Art

Name	Params	GFLOPs	Time(ms)	Acc(g)	Acc(c)	mIOU
Lite	0.56	0.44	3.88	90.83	69.38	61.13
Lite(+)	0.56	0.44	3.35	91.97	74.59	66.40
Medium	2.70	1.70	6.72	91.86	75.39	66.66
Medium(+)	2.70	1.70	6.72	93.16	79.78	71.63
Large	10.52	6.11	10.27	93.00	78.99	70.84
Large(+)	10.52	6.11	10.27	93.49	81.53	73.61

Table 5: Quantitative performance of Lite, Medium, and Large Networks, (+) indicates experiments that have been trained for more epochs.

Method	Resolution	Pretrain	Encoder	Params (M)	mIOU
FCN8 [8]	1/2	ImageNet	VGG	134.5	57.0
DeconvNet [13]	1/2	ImageNet	VGG	252	48.9
SegNet [1]	1/2	-	-	29.5	46.4
ENet [14]	1/2	-	-	0.37	51.3
Lite [ours]	1/2	-	MobileNetV2	0.56	66.4
Medium [ours]	1/2	-	MobileNetV2	2.70	71.6
Large [ours]	1/2	-	MobileNetV2	10.52	73.9

Table 6: Comparison to state-of-the-arts that are trained and tested on half resolution.

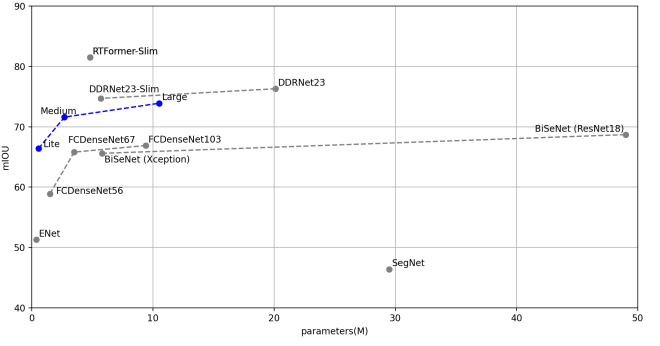


Figure 9: Comparison of our approach to state-of-the-art approaches, methods connected with a dashed line belong to the same paper.

Method	Resolution	Pretrain	Encoder	Params (M)	mIOU
FCDenseNet56 [7]	1/2 + F	-	DenseNet	1.5	58.9
FCDenseNet67 [7]	1/2 + F	-	DenseNet	3.5	65.8
FCDenseNet103 [7]	1/2 + F	-	DenseNet	9.4	66.9
BiSeNet [17]	F	ImageNet	Xception	5.8	65.6
BiSeNet [17]	F	ImageNet	ResNet18	49.0	68.7
DDRNet23-Slim [5]	F	ImageNet	-	5.7	74.7
DDRNet23 [5]	F	ImageNet	-	20.1	76.3
RTFormer-Slim [16]	F	ImageNet	ViT	4.8	81.5
RTFormer [16]	F	ImageNet	ViT	16.8	82.5
Lite [ours]	1/2	-	MobileNetV2	0.56	66.4
Medium [ours]	1/2	-	MobileNetV2	2.70	71.6
Large [ours]	1/2	-	MobileNetV2	10.52	73.9

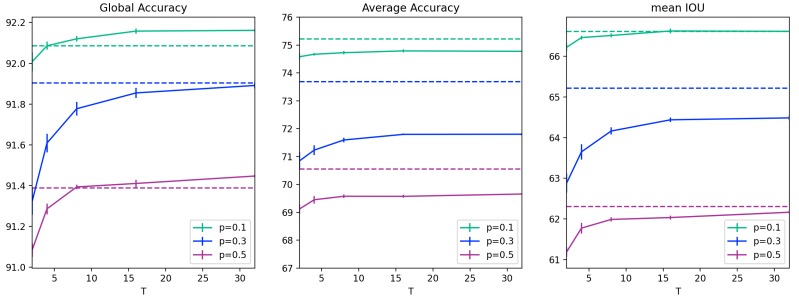
Table 7: Comparison to state-of-the-arts that are trained and tested on using full resolution. 1/2 + F: trained on half resolution followed by fine-tuning on full resolution.

4.2 Bayesian Inference

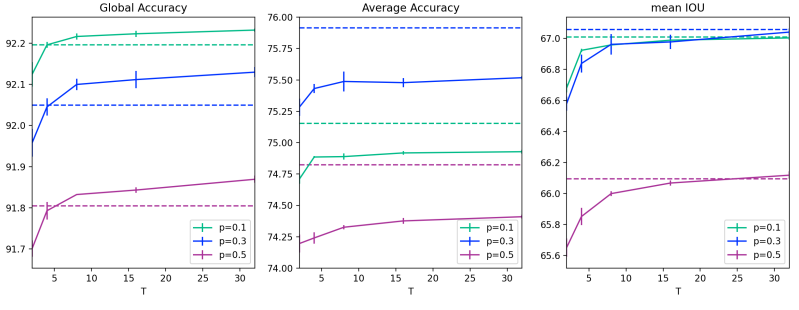
4.2.1 Network Performance and Calibration

Dropout	Sd	Acc(g)	Acc(g) [b]	Acc(c)	Acc(c) [b]	mIOU	mIOU [b]
-	-	92.08	-	75.42	-	66.90	-
0.1	-	92.09	92.14	75.22	74.81	66.61	66.59
0.3	-	91.90	91.87	73.68	71.74	65.21	64.34
0.5	-	91.39	91.40	70.55	69.63	62.31	62.07
-	0.1	**92.20**	**92.22**	75.15	74.92	67.01	66.97
-	0.3	92.05	92.08	**75.91**	**75.48**	**67.06**	**66.97**
-	0.5	91.80	91.83	74.82	74.37	66.09	66.00
0.1	0.1	92.08	92.10	75.05	74.40	66.70	66.50
0.3	0.3	91.78	91.73	71.97	70.91	63.92	63.30
0.5	0.5	91.28	91.28	68.38	67.13	60.62	60.14

Table 8: Accuracy and IOU with Bayesian Approximation, [b] indicates values ran with Bayesian approximation with $T = 10$, without [b] indicates deterministic inference.



(a) Performance of *dropout* variants with varying T



(b) Performance of *sd* variants with varying T

Figure 10: Performance of networks of Bayesian variants, dashed lines indicate performance of deterministic test-time inference, error bars show the variance of the values from 5 repeated trials. Note that global accuracy can usually be improved with Bayesian inference, but average accuracy and mIOU tend to be deteriorate.

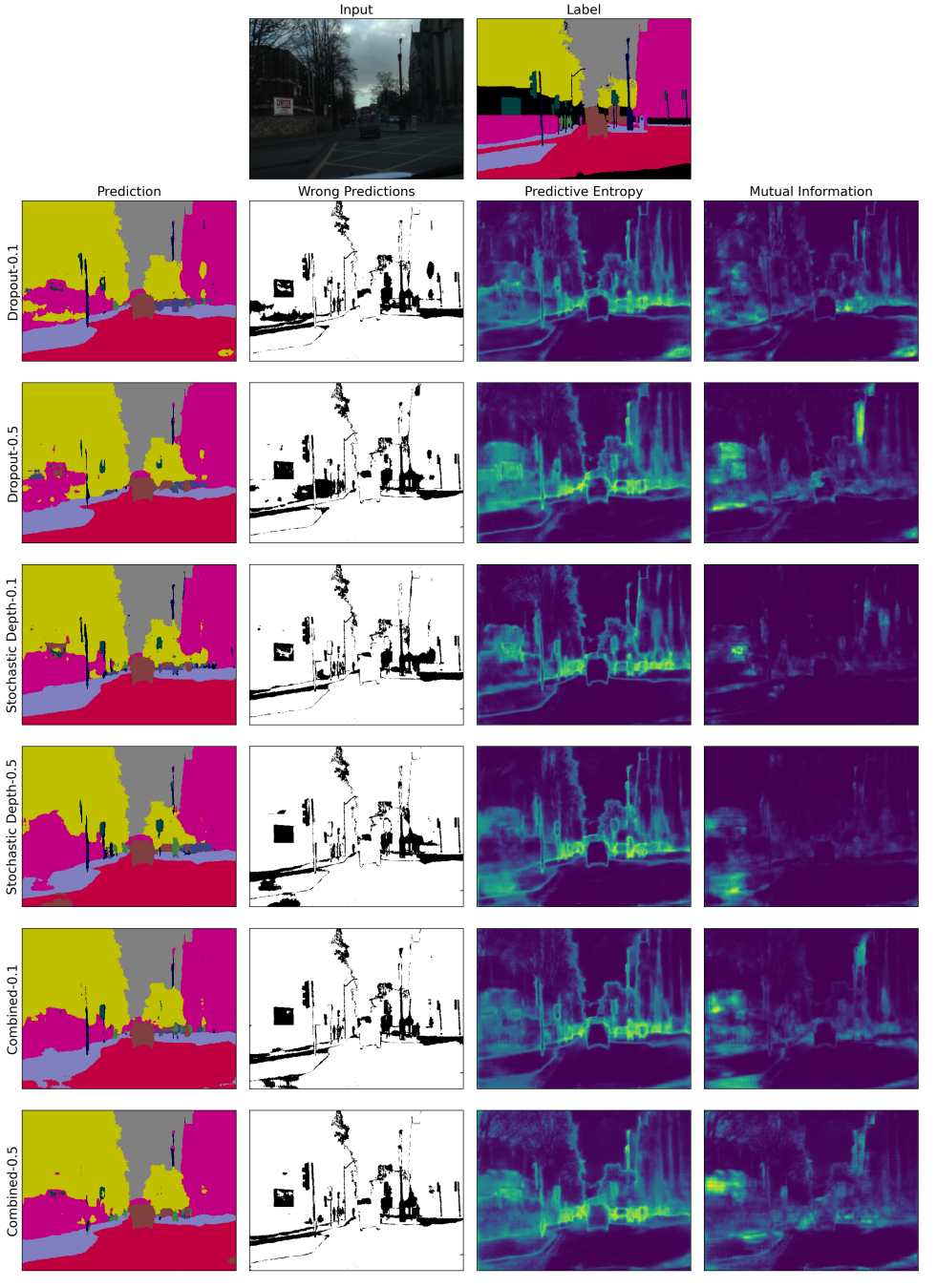


Figure 11: Qualitative results of Bayesian variants on a test set image, note that variants with higher probabilities (i.e. with probability 0.5) produce qualitatively more uncertainties.

Dropout	Sd	mIOU ↑	ECE ↓	ECE [b]	MCE ↓	MCE [b]	PAvPU ↑
-	-	66.90	2.71	-	1.61	-	-
0.1	-	66.61	2.46	1.60	1.40	0.84	90.14
0.3	-	65.21	2.19	0.70	1.25	0.16	88.35
0.5	-	62.31	2.20	**0.17**	1.09	**0.05**	88.63
-	0.1	67.01	2.36	1.87	1.26	0.92	**90.33**
-	0.3	**67.06**	2.14	1.23	1.11	0.58	89.79
-	0.5	66.09	2.20	1.17	1.16	0.55	89.61
0.1	0.1	66.70	2.30	1.07	1.31	0.48	89.83
0.3	0.3	63.92	1.92	0.27	1.00	0.12	88.53
0.5	0.5	60.62	**1.72**	1.10	**0.76**	0.41	87.66

Table 9: MCE, ECE, and PAvPU with Bayesian Approximation, [b] indicates values ran with Bayesian approximation with $T = 10$, without [b] indicates deterministic inference.

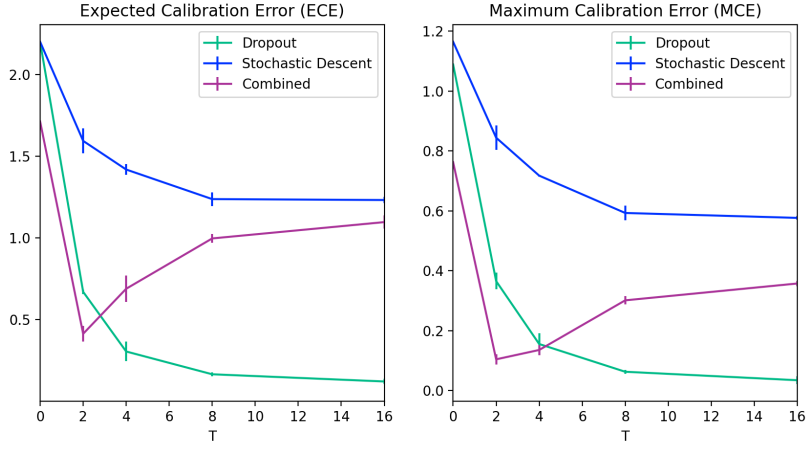


Figure 12: Effect of T on calibration errors for the different Bayesian methods with probability 0.5, error bars show variance obtained over 5 repeated trials over the test set.

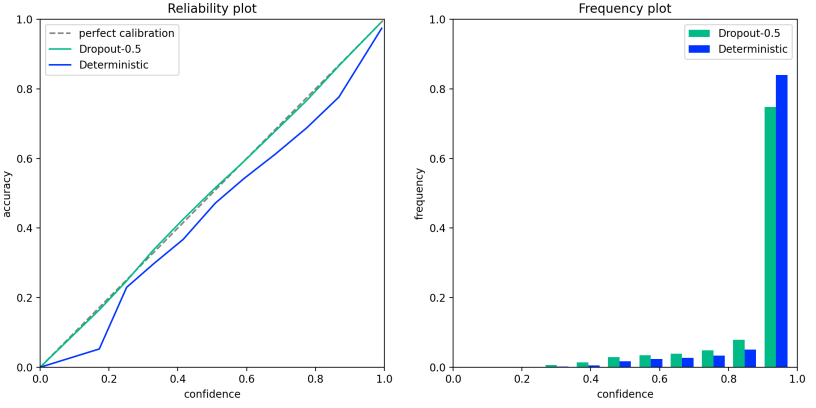


Figure 13: The reliability and frequency plot of stochastic vs. deterministic networks, networks with dropout show much more calibrated uncertainties whilst deterministic networks are overconfident.

Inference	Time(ms) ↓	FPS ↑	Memory ↓	mIOU ↑	ECE ↓	MCE ↓
dropout-0.1						
Det.	11.4±0.0	88	189 Mb	66.61	2.46	1.40
T=2	21.9±0.0	46	358 Mb	65.94	1.97	1.11
T=4	36.1±4.9	28	711 Mb	66.36	1.72	0.94
T=8	61.5±11.6	16	1.39 Gb	66.50	1.66	0.87
T=16	107.4±1.9	9	2.77 Gb	66.58	1.55	0.80
T=32	215.5±5.4	5	5.54 Gb	66.65	1.52	0.80
sd-0.3						
Det.	11.4±0.0	88	190 Mb	**67.06**	2.14	1.11
T=2	19.8±2.3	51	361 Mb	66.45	1.68	0.89
T=4	33.7±4.1	30	712 Mb	66.90	1.37	0.69
T=8	52.4±0.7	19	1.39 Gb	66.94	1.30	0.65
T=16	108.7±0.2	9	2.77 Gb	67.01	1.25	0.60
T=32	220.4±0.5	5	5.54 Gb	67.03	**1.22**	**0.58**

Table 10: Computational requirements for different Bayesian inference, using the dropout and sd variant that obtained highest mIOU performance. Stochastic depth obtains overall lower ECE and higher mIOU compared to dropout.

4.2.2 Out of distribution data

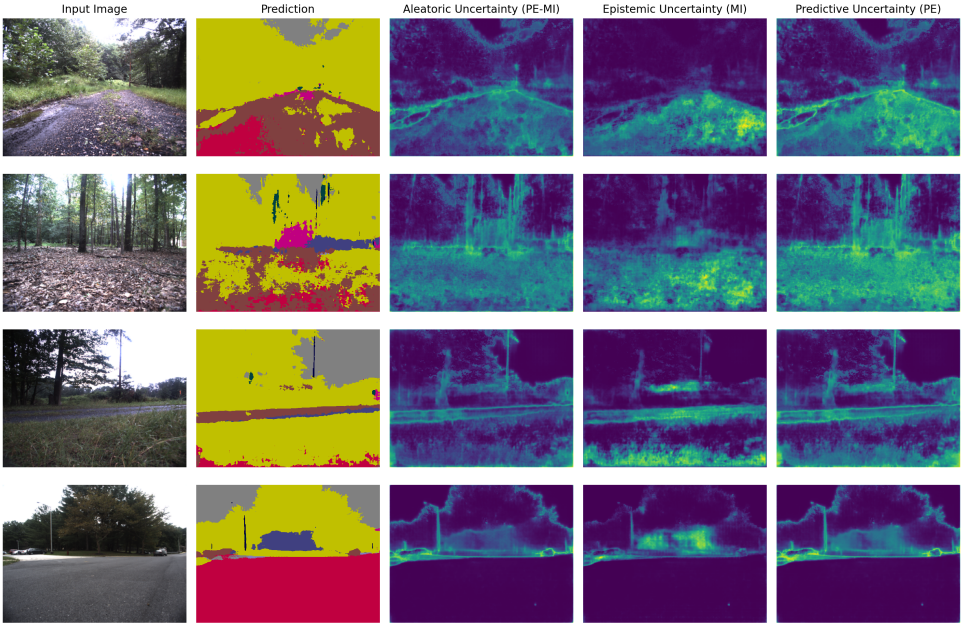


Figure 14: The predictive outputs of a Bayesian network with dropout, we observe qualitatively very high epistemic uncertainty values on images that are different to our training set, e.g. the first three row, and much lower uncertainty on the last image, which is similar to the CamVid dataset.

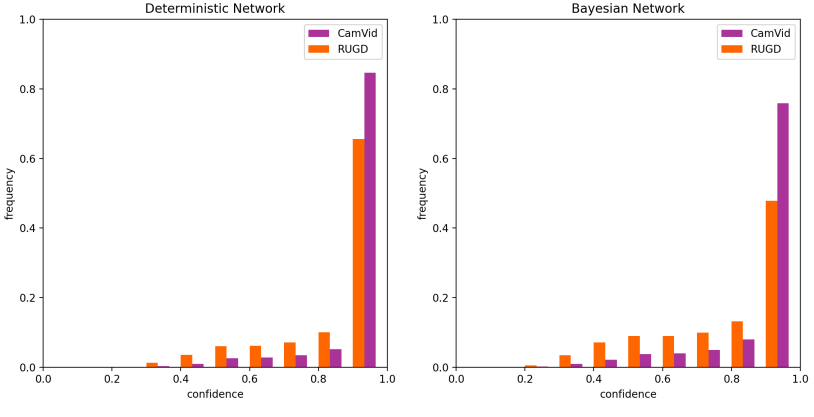


Figure 15: The distribution of softmax probabilities of the deterministic and Bayesian network on the CamVid dataset and the RUGD dataset. The Bayesian network produces much lower confidence on the out-of-distribution dataset compared to the deterministic network.

5 Discussion

5.1 Efficient Designs Discussions

Skip connections are a low-cost way to improve the detection of small classes. Our experiments with modifications to skip connections showed that concatenating features from the encoder to the decoder significantly helps the detection of small obstacles. In addition, we highlight that this operation is relatively low-cost, adding only a small fraction of parameter and inference time. Note that this may not hold in other encoder-decoder networks that do not use inverted residual blocks. The expansion steps in inverted residual blocks are the dominant computational cost of our network and hence transferring high-resolution features comes at a relatively low cost.

Scaling coefficients can be improved with a small grid search. Our network scaling experiments use width and multipliers in $[0.5, 1.0, 1.5]$ for convenience of quickly exploring the parameter space. A much better but perhaps more costly method would be to adopt a small grid search similar to [15], whereby we find the best scaling coefficient for the network depth and width based on a few constraints. For example, we can perform a random grid search of width and depth multipliers to determine the most efficient way to improve performance. The scaling coefficients obtained this way are more likely going to better optimize the scaling efficiency of the network.

Wider features are desired in segmentation networks. This finding is validated by our experiments involving shallow decoders and network scaling. We conjecture that network width is extremely crucial in encoder-decoder networks, allowing networks to improve their detection for small classes. We provide conjecture on why this is the case, although more rigorous experiments are required to validate this hypothesis:

The nature of network width for encoder-decoder architectures is different to classification networks. Consider a standard classification network such as MobileNet, it outputs last features with shape $(7 \times 7 \times 1024)$ to predict 1000 classes on ImageNet. This suggests that the information contained in the last feature map sits at a much lower (about 50-fold) dimensional manifold than the feature dimension itself. Now consider building an encoder-decoder architecture with MobileNet, the final feature map of the network from the decoder will have 32 channels with half the original resolution. Compared with the final prediction at full resolution ($4\times$ more pixels) with 11 channels, we actually have less information contained in the feature map than is required from the final prediction.

Therefore, the loss in spatial information from lossy downsampling and upsampling steps can be reduced by increasing the network width, which increases the sparsity at which information exists in the final feature output. Modern CNNs are typically designed and fine-tuned to the ImageNet dataset with 1000 classes, so its intermediate feature outputs are not designed for semantic segmentation. This points to a future direction of neural architecture search (NAS) for finding scalable and efficient encoder-decoder architectures, which we will further discuss in the conclusion.

5.2 Bayesian Inference Discussion

Trade-off between network performance and calibration error An obvious artifact of testing with Bayesian inference is that they produce lower mIOU and per-class accuracy.

This presents a trade-off between improving the calibration error and achieving high performance. Our experiment show that using dropout probabilities as low as 0.1 combined with T as low as 2 to 5, is enough to reduce the calibration error by a large margin. Whilst using several stochastic forward passes for Bayesian approximation achieves significantly improved calibration errors, it also reduces the performance measured in mIOU compare to deterministic test-time inference. Furthermore, this gap is usually non-reducible even at very high T, as shown in Figure 10. In other words, Bayesian inference using test-time stochastic regularization trades off performance for improved calibration.

Trade-off between computation cost and calibration error Our results in Table 9 show that the best calibration error is obtained when Bayesian inference is used with T stochastic forward passes. However, a clear downside of this approach is that the computational costs grows with T. Table 10 show the computational trade-off between T, mIOU, and calibration errors. The first observation we make is that inference time does not grow linearly with T, but memory requirement roughly grows linearly with T. This is intuitive since GPU can compute inference in parallel to save compute time, but still has to produce the entire intermediate tensors. We see that with our GPU environment, we lose real-time inference at around $T = 4$, where the framerate drops below 30.

Dropout versus stochastic depth Our results show that stochastic depth is a valid way of obtaining more calibrated uncertainties. We further show that Bayesian variants with stochastic depth with appropriate probabilities have the desired property of achieving higher mIOU whilst reducing ECE. We demonstrate the trade-off with *dropout-0.1* and *sd-0.3* in Table 10, which are the best network in terms of mIOU for the dropout and sd variants. We see that networks trained with stochastic depth obtain higher accuracy and lower calibration errors compared to dropout methods. This is true for both deterministic and non-deterministic test inferences. Therefore in practical cases, stochastic depth is a good alternative over dropout methods to achieve lower calibration errors without losing performance. On the other hand, if one is only interested in achieving the best calibration error, dropout methods with high probability (e.g. 0.5) with test-time Bayesian inference is the best choice.

PAvPU is a biased metric for measuring uncertainty Our results in Table 8 and Table 9 show that PAvPU is proportional to the global accuracy of the data. We see similar observations with papers that use this metric or its variations [9, 10, 11], where networks that perform better tend to achieve higher area under the curve. This phenomenon is due to the fact that the curve is not normalized to its performance, in the limit with maximum uncertainty threshold 1, the formula 9 equates to exactly the global accuracy.

We argue that PAvPU is not a good metric for measuring the quality of uncertainties due to the following reasons. 1) The quality of uncertainty should be mutually exclusive with respect to the quality of performance, in other words, the network should be allowed to predict poorly, and "knows" that it's performing poorly. 2) The PAvPU value is computed with based on a sliding window of uncertainty threshold that's derived from normalized entropy values across a validation set, which makes the metric dependent on the dataset (i.e. if it's in-domain, the entropy values will be lower, and vice versa if the validation is out-of-distribution from the training set, the entropy values will be higher), as well as noisy artifacts from the stochastic nature of Bayesian approximation. We argue that a good metric should

not be dependent on the dataset itself and that such quality are defects that make the metric unsuitable for evaluating uncertainty.

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[2] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2): 88–97, 2009.

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] Yuanduo Hong, Huihui Pan, Weichao Sun, and Yisong Jia. Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes. *arXiv preprint arXiv:2101.06085*, 2021.

[6] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[7] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 11–19, 2017.

[8] 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*, pages 3431–3440, 2015.

[9] Patrick McClure, Nao Rho, John A Lee, Jakub R Kaczmarzyk, Charles Y Zheng, Satrajit S Ghosh, Dylan M Nielson, Adam G Thomas, Peter Bandettini, and Francisco Pereira. Knowing what you know in brain segmentation using bayesian deep neural networks. *Frontiers in neuroinformatics*, 13:67, 2019.

[10] Aryan Mobiny, Pengyu Yuan, Supratik K Moulik, Naveen Garg, Carol C Wu, and Hien Van Nguyen. Dropconnect is effective in modeling uncertainty of bayesian deep networks. *Scientific reports*, 11(1):5458, 2021.

[11] Jishnu Mukhoti and Yarin Gal. Evaluating bayesian deep learning methods for semantic segmentation. *arXiv preprint arXiv:1811.12709*, 2018.

[12] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[13] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[14] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[15] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[16] Jian Wang, Chenhui Gou, Qiman Wu, Haocheng Feng, Junyu Han, Errui Ding, and Jingdong Wang. Rtformer: Efficient design for real-time semantic segmentation with transformer. *Advances in Neural Information Processing Systems*, 35:7423–7436, 2022.

[17] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341, 2018.