Computer vision - Project 2

Segmentation
Camvid

Brice Tayart

July 21, 2020

1 Introduction

The purpose of the project is identify anomalous, out-of-distribution pixels when doing semantic segmentation of images with deep learning. For this purpose, the CamVid dataset is used, with the labels of some classes removed in order to get unlabeled pixels that are known to be out-of-distribution.

Two segmentation architectures are used in the project: ENet[3] and BiSeNetv2[5].

2 Set-up

2.1 Dataset

The project uses the ENet implementation available at `https://github.com/davidtvs/PyTorch-ENet`. The dataset is a 11 class version of the CamVid dataset used in the SegNet[1] paper, with the *road* and *road marking* classes merged. The dataset is available at `https://github.com/alexgkendall/SegNet-Tutorial/tree/master/CamVid`. It is split into a train, validation and test subsets of 367, 101 and 233 images respectively. The images are downsampled from their original resolution of 720×960 to 360×480.

For tests about prediction confidence, the *car*, *pedestrian* and *bicyclist* classes were removed and merged into the *unlabeled* class.

A first training was done on the dataset with all classes, subsequent ones on the dataset with fewer classes.

2.2 Architecture

There are two major types of CNN architectures for segmentation. One type uses *dilation* (or, in frenglish, *à trous* convolutions). A typical CNN for classification is made of several convolution cells, within which the size of the image is constant, separated by downsampling operators. For segmentation, downsampling is replaced by the use of dilated convolutions so that the image size remains constant. Pixels that were removed by the downsampling are instead skipped by the dilation. CNN pre-trained for classification can be easily adapted for segmentation. Examples include the Deeplab family such as Deeplabv3[2].

Another option is an encoder-decoder scheme. An encoder transforms the image into a wide, low-resolution feature map. The feature map is then decoded into a label map. Several tricks are used to let details from the high resolution image be used in the decoder for a finer segmentation. Examples include UNet[4], SegNet[1], PSPNet[7].

The two architectures used for this project are ENet[3] and BiSeNetv2[5].

ENet has a mixed architecture, with an encoder to downsample the image by a factor 4, followed by a dilation type CNN and a decoder to upsample back to the orignial size. It is extremely lightweight, with only about 350 000 parameters.

BiSeNetv2 uses a CNN with two branches. The *details branch* is a wide and shallow CNN made of plain convolution layers, with a feature map downsampled by a factor 8 with respect to the original image. The *semantic branch* is a narrower and deeper CNN akin to a wide-ResNet[6]. Both are combined by a custom *guided aggregation layer* and the final result is simply upsampled 8-fold, back to the original resolution. Although much heavier than ENet (1.95M parameters), it runs faster on a GPU as it uses well optimized 3×3 convolutions instead of dilated convolutions.

The BiSeNet model used here slightly differs from the network presented in the paper, as it has 8 channels instead of 16 at the root of the semantic branch. This saves some memory usage and it only slightly degraded performance in the article. Also, auxiliary losses in the semantic branch to hasten training were not used, once again to limit memory consumption.

2.3 Training

All models were trained over 300 epochs on the train subset, using an Adam optimizer with learning rate 5.10^{-4} with 0.1 decay every 100 epochs, momentum $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The weight decay was set to 2.10^{-3} for ENet and 5.10^{-3} for the BiSeNetv2 model. Batch size was 6 for ENet and 16 for BiSeNet (or, as much as what is possible with 4GB GPU memory).

The classes are weighted using the weights described in the ENet paper: $w_{class} = 1/\log(1.02 + p_{class}))$ where p_{class} is the proportion of pixels belonging to the class. The weight of the *unlabeled* class is set to 0.

The following ENet model were trained: *ENet_allclasses_wd2* was trained with all 11 classes, *ENet_8classes*, *ENet_8classes1*, ...,*ENet_8classes4* were trained with 8 classes, *ENet_dropout* was trained wit 8 classes and 0.15 dropout.

The following BiSeNet model were trained: *BiSeNet_allclasses_wd3* was trained with all 11 classes, *BiSeNet_8classes* was trained with 8 classes, *BiSeNet_dropout* was trained with 8 classes and 0.15 dropout.

All models are available in the *save* directory, and the list of model trained with their exact parameters are in the *experiments.py* file.

2.4 Validation

Sample predictions from model *ENet_allclass_wd2* (with all 11 classes) and *ENet_8classes* (with only 8 classes left) are shown in figures 1 and 2. What we see is that the prediction confidence is almost always close to 1 in the middle of the segments, and that it only drops near the borders between segments. There is a small interpolation footprint, with a regular grid of squares with a slightly lower confidence. After training the network with the pixels labeled car, bicyclist or pedestrian weigthed to 0 for the computation of the loss function, pixels from these classes are necessarily mis-labeled by the model. However, this bad prediction is sometime done with a high confidence (on the top image of figure 2, cars are labeled mostly as road with a high confidence).

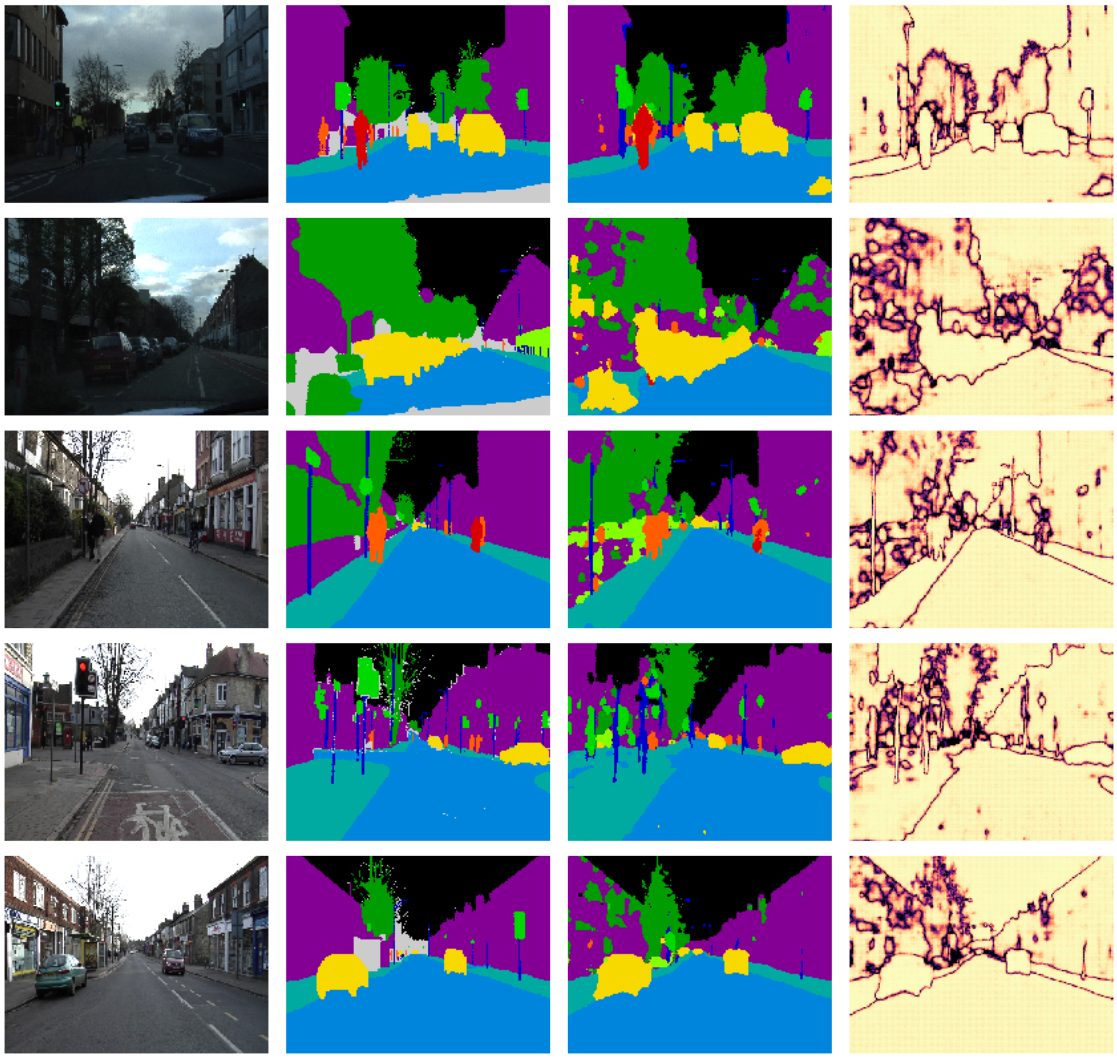

Figure 1: *Ground truth, prediction and confidence with* ENet_allclass_wd2 *model*

3 Predicting uncertainty

Three methods were used to attempt to measure the prediction uncertainty. Each method provide a "confidence" estimate that may be used to assert whether a pixel is anomalous, based on a threshold.

- Use of the Maximum Softmax Value, including variations in the softmax temperature
- Model aggregation
- Monte-Carlo dropout

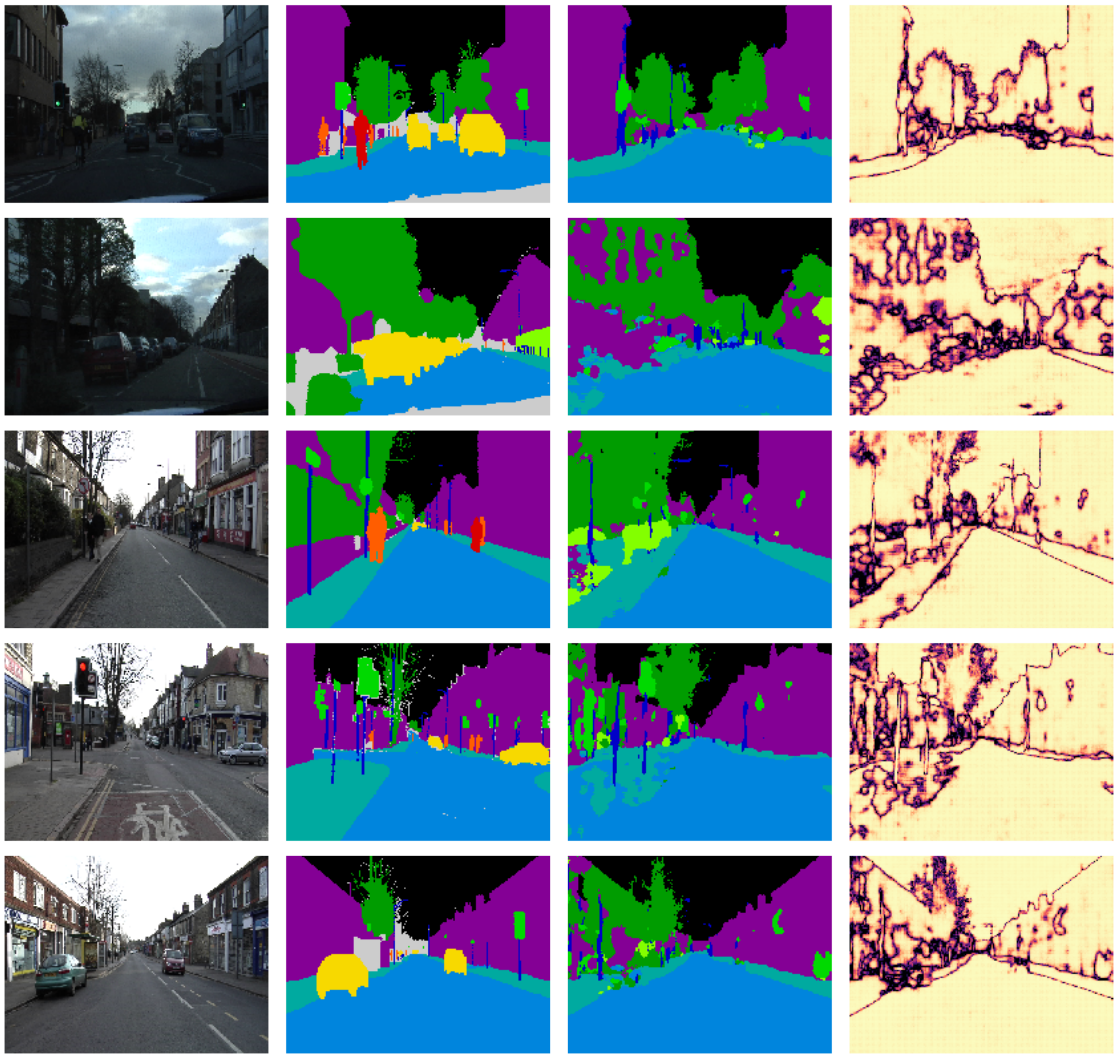

Figure 2: *Ground truth, prediction and confidence with* ENet_8class *model*

3.1 Maximum Softmax Value

The maximum softmax value (MSV) is a simple measure of confidence. Because the softmax provides for each pixel a vector of values normalized to 1, an interpretation is to consider these as probabilities for the pixel to belong to each class. One parameter of the softmax is the temperature T. The i-th component s_i of the softmax of a vector x of length l is :

$$s_i = \frac{e^{x_i/T}}{\sum_{j=1}^{l} e^{x_j/T}} \tag{1}$$

Softmax temperature	ROC AUC
1.3	0.8090
1.5	0.8102
1.8	0.8103
2.0	0.8101
2.5	0.8093
3.0	0.8084

Table 1: *ROC AUC for different values of T*

It is traditionally set to 1 as it affects neither the predicted class nor the model training (temperature merely scales the loss and gradient). Temperature does matter however if we seek to interpret the softmax value as a probability.

One may attempt to seek anomalous (out of distribution) pixels based on a threshold on the MSV. Using the 11 labels as ground truth (exculding unlabeled pixels) and the 8 classes predicted by the model, an histogram of prediction confidence is done for each (ground truth, prediction) pair using all the pixels of all the images of the test subset. Histograms are further aggregared to look at well classified pixels, misclassified pixels, pixels with a ground truth label in one of the 3 removed classes. A ROC curve is also computed.

With a softmax temperature T equal to 1, the mis-classified pixels tend to has a high MSV, i.e. be misclassified with a high confidence as shown in figure 3. A higher temperature alleviates this, and misclassified pixels have a lower confidence. The same alo happens for well classified pixels, though, and there is no clear confidence threshold between well and misclassified pixels. Neither is there a threshold that would identify anomalous pixels but not well classified ones. The Receiver Operating Curves are quite similar for all temperatures. In order to determine the usefulness of the MSV for classifying anomalies, the ROC AUC is computed for temperatures [1.3, 1.5, 1.8, 2.0, 2.5, 3.0] as shown in figures 4. The ROC AUC, in table1, barely changes, so it is clear that fine tuning the softmax temperature does not help for anomaly detection.

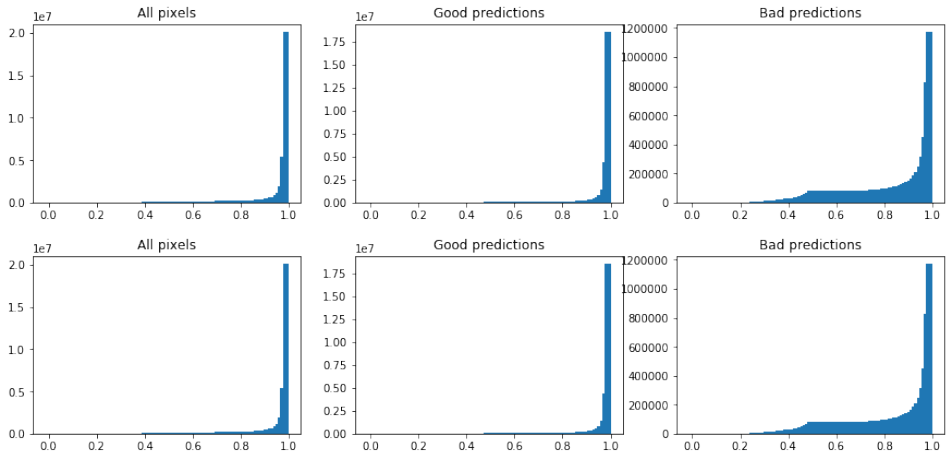

Figure 3: *Confidence histograms for $T = 1$*

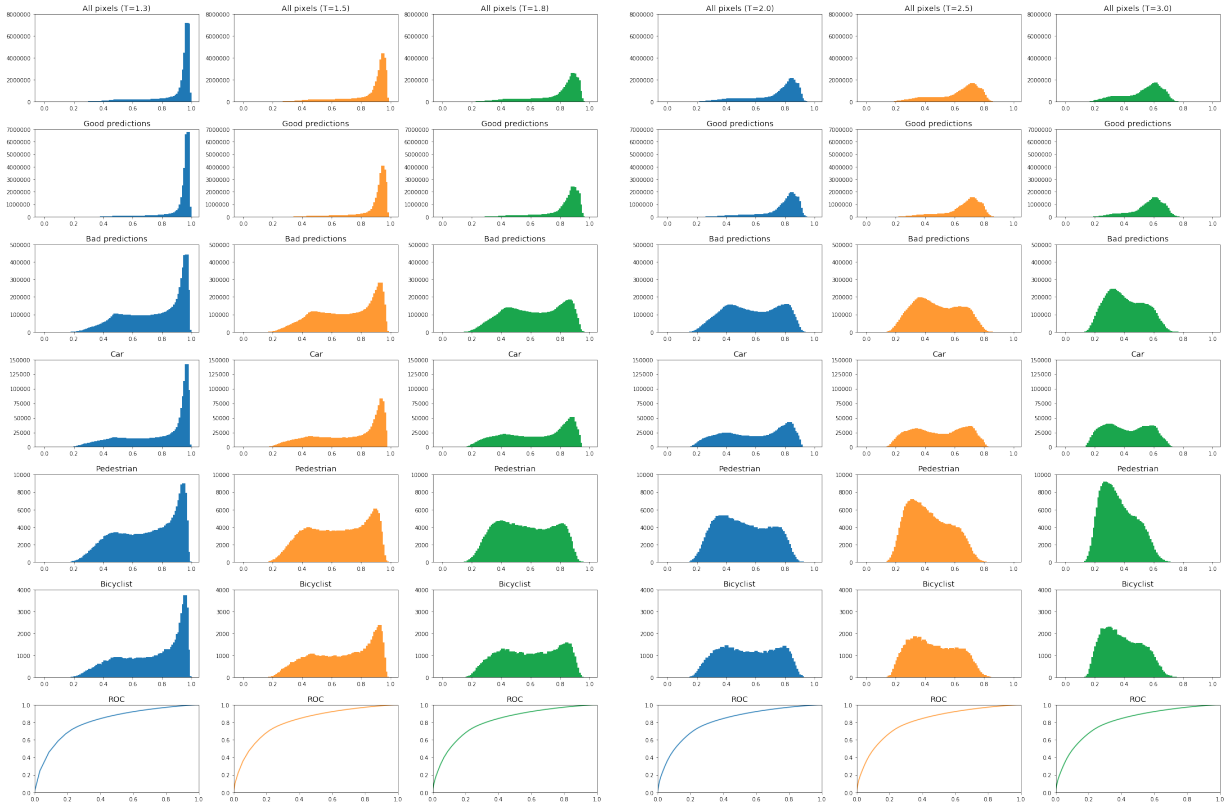

Figure 4: *Confidence histograms and ROC for* $T \in \{1.3, 1.5, 1.8, 2.0, 2.5, 3.0\}$
Histograms from top to bottom: all pixels, well classified, misclassified, class car, *class* pedestrian, *class* bicyclist
Bottom-most curve is ROC AUC

3.2 MonteCarlo Dropout

Dropout is often used during training, for regularization. MC dropout also uses the dropout for inference. The result of the prediction is thus stochastic. Running multiple inferences for the same image gives access to the distribution of the predictions. Predictions will have a high confidence if the maximum softmax value is consistently high.

An ENet model with 0.15 dropout is used. For each image, inference is run $n_{MC} = 24$ times. For each pixel and each class, the average μ and the variance var of the softmax across all runs is computed. The predicted class is the one with the highest average softmax value. For the confidence, a large but consitent softmax value is sought. The confidence c is thus a composite term: $c = \mu - \kappa \times \sqrt{var}$, where κ is an hyper-parameter set to 1. The confidence histograms are shown in figure 5. An example of the predictions and confidence is shown in figure 6 The ROC AUC is 0.8189.

Another option was explored, where the predicted class is the one most often predicted class across all runs. The confidence is then simply chosen as the number of times that class was predicted, over the total number of predictions; it is discrete rahter than continuous (it varies by increments of $1/n_{MC}$). The result

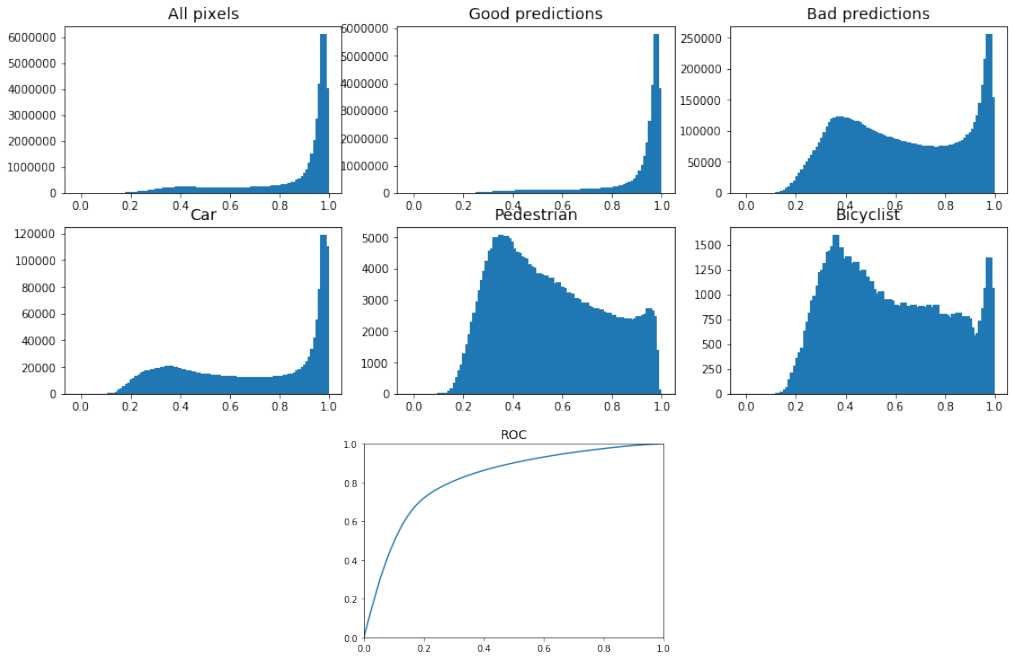

Figure 5: *Confidence histograms and ROC ENet with MonteCarlo dropout, using average of softmax values*

are shown in figure 7. The ROC AUC is 0.7928.

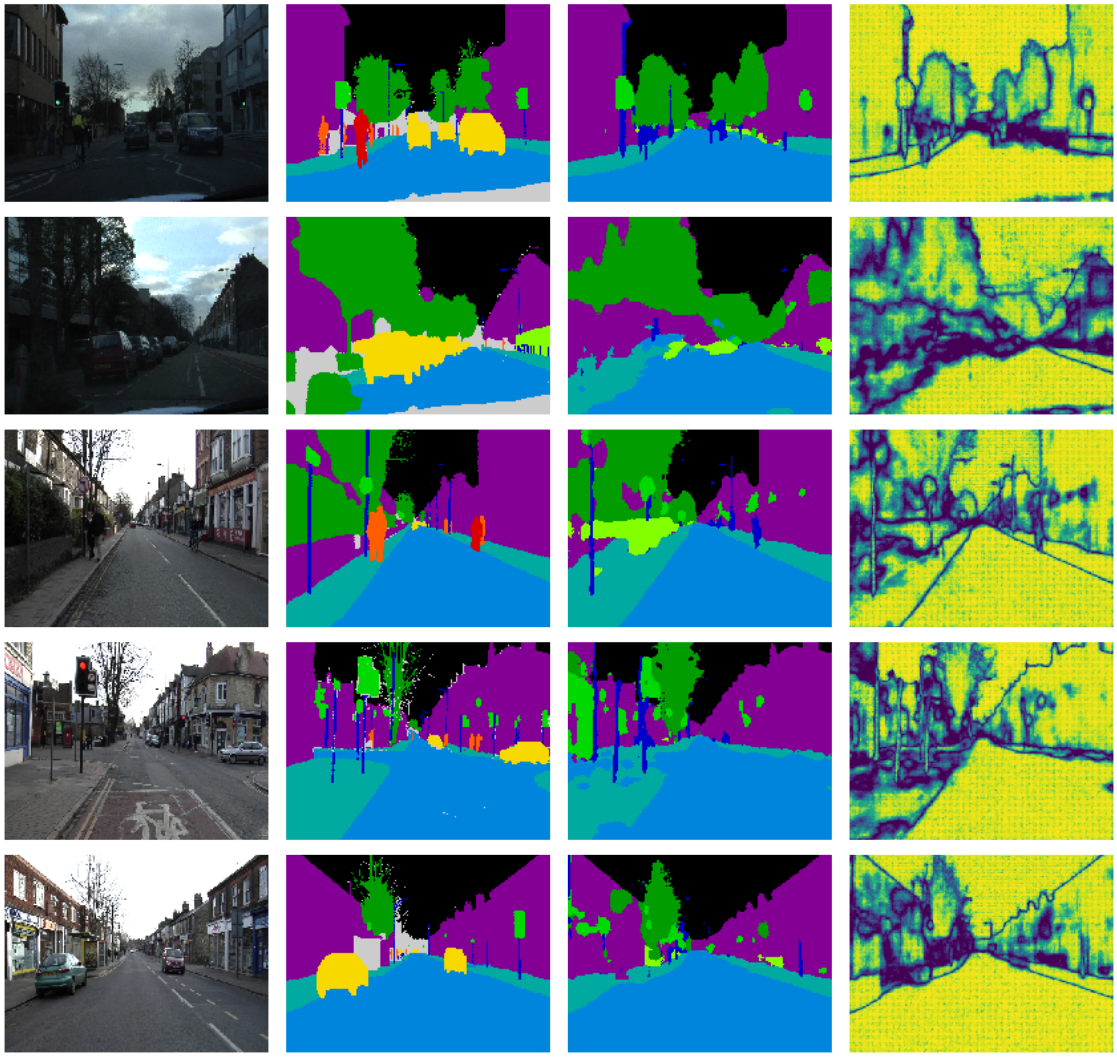

Figure 6: *Ground truth, prediction and confidence using the MC dropout method. Some pedestrians and cyclists can be seen as low confidence spots, but cars are not well identified as anomalous*

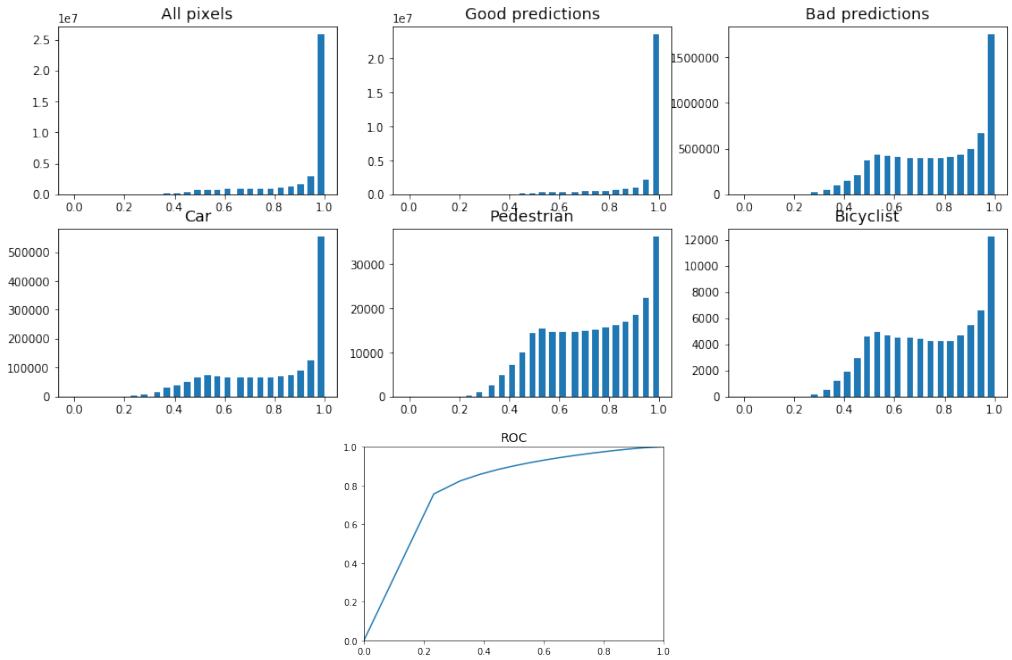

Figure 7: *Confidence histograms and ROC ENet with MonteCarlo dropout, using count of predictions*

3.3 Model aggregation Dropout

Rather than using the stochastic nature of a random dropout from a single model, several models trained from different (and stochastic) initial conditions are used. This involved more calculations for training and more memory (as each instance has its own parameters) but may bring more diversity in the models than a mere dropout. The results are shown in figure 8. The distribution of the confidences is multimodal with several secondary peaks. This is because the aggregation is done after softmax: outliers are likely to be close to 0 or 1, and this gives these distinct peaks after aggregating the small number of predictions. These were not seen on the MC dropout as there were 24 runs. The ROC AUC is 0.8215.

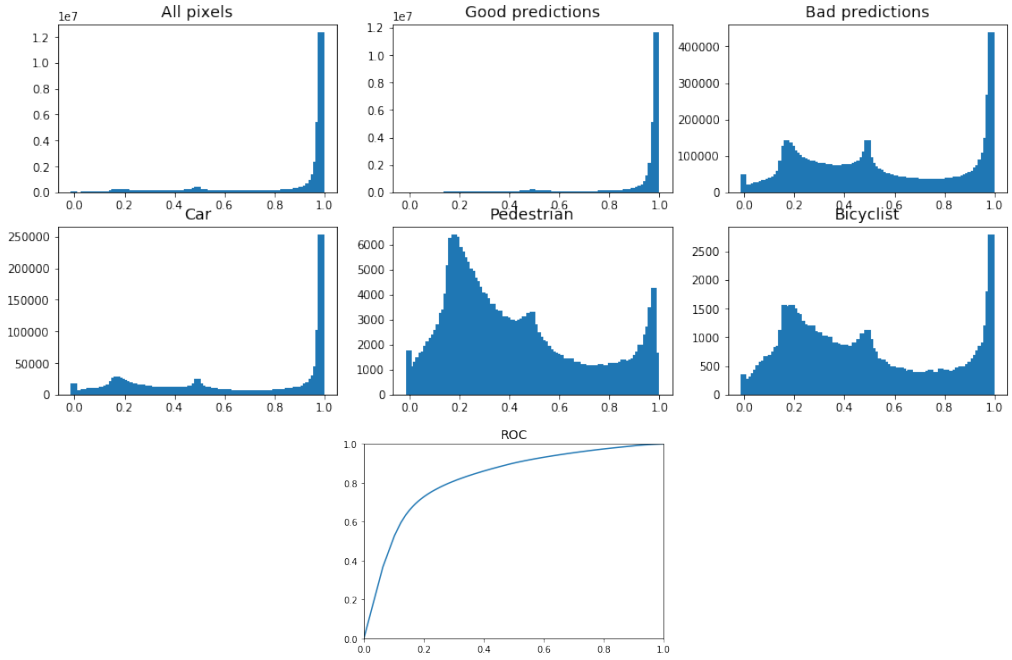

Figure 8: *Confidence histograms and ROC ENet with MonteCarlo dropout, using count of predictions*

3.4 BiSeNet model

The BiSeNet model trained with 8 classes with 0.15 dropout rate is used: *BiSeNet_dropout*. Preduction is done with MC dropout, with softmax values averaged over 24 runs and a confidence computed with $\kappa = 1$. It is to be noted that the introduction of dropout adds regularization and that, even without the MC dropout. This models has a 0.706 average IoU metric on the validation subset, while the same model without dropout only has 0.673 IoU (despite fine tuning the L_2 regularization parameter).

The inference results are shown in figure 9 and the confidence histograms in figure 10. We can see that most of the pedestrians, bicyclists and cars are associated with a drop in confidence with this model. Predicting classification success with the confidence is done with a slight improvement in ROC AUC compared to ENet, with AUC = 0.8384.

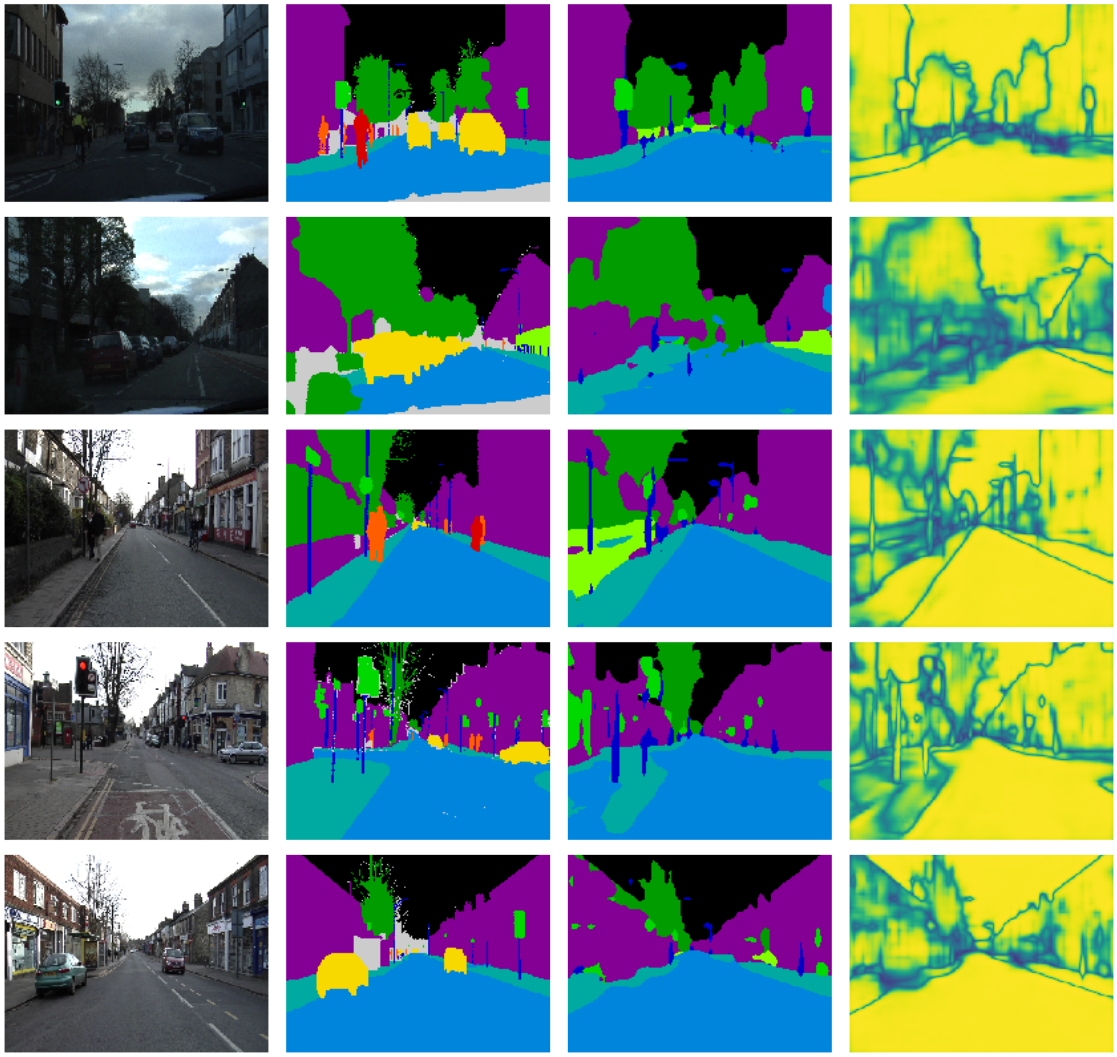

Figure 9: *Ground truth, prediction and confidence; MC dropout method with BiSeNetv2 model with 0.15 dropout rate. Pedestrians, cyclists and cars are associated with some drop in prediction confidence*

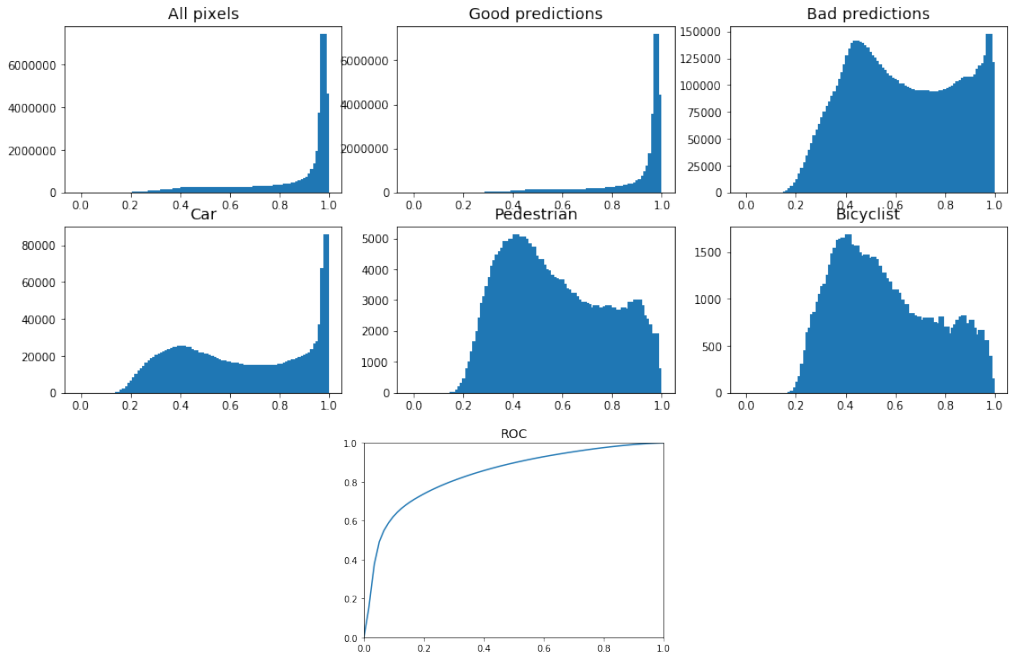

Figure 10: *Confidence histograms and ROC, BiSeNetv2 with MonteCarlo dropout, using mean of softmax value*

Discussion and Conclusion

Pixels from the three classes are not identified as anomalous with the same ease. I particular, *car* is much more difficult to identify as *bicyclist* and *pedestrian* and, in all histograms, it has a large proportion of pixels classified with a high confidence.

The use of serveral models in place of MC dropout slightly increased the AUC, i.e. provided a confidence estimate that better discriminated between well- and mis-classified pixels. Training several models probably produces more diversity in the predictions than a mere dropout.

Other methods (such as predicting directly the confidence or the loss with a CNN) could also be tested.

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *arXiv e-prints*, arXiv:1511.00561 (Nov. 2015), arXiv:1511.00561. arXiv: 1511.00561 [cs.CV].

[2] Liang-Chieh Chen et al. "Rethinking Atrous Convolution for Semantic Image Segmentation". In: *arXiv e-prints*, arXiv:1706.05587 (June 2017), arXiv:1706.05587. arXiv: 1706.05587 [cs.CV].

[3] Adam Paszke et al. "ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation". In: *arXiv e-prints*, arXiv:1606.02147 (June 2016), arXiv:1606.02147. arXiv: 1606.02147 [cs.CV].

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *arXiv e-prints*, arXiv:1505.04597 (May 2015), arXiv:1505.04597. arXiv: 1505.04597 [cs.CV].

[5] Changqian Yu et al. "BiSeNet V2: Bilateral Network with Guided Aggregation for Real-time Semantic Segmentation". In: *arXiv e-prints*, arXiv:2004.02147 (Apr. 2020), arXiv:2004.02147. arXiv: 2004.02147 [cs.CV].

[6] Sergey Zagoruyko and Nikos Komodakis. "Wide Residual Networks". In: *arXiv e-prints*, arXiv:1605.07146 (May 2016), arXiv:1605.07146. arXiv: 1605.07146 [cs.CV].

[7] Hengshuang Zhao et al. "Pyramid Scene Parsing Network". In: *arXiv e-prints*, arXiv:1612.01105 (Dec. 2016), arXiv:1612.01105. arXiv: 1612.01105 [cs.CV].