Multiclass Semantic Segmentation of Urban Scenery: Can CBAM or Attention U-Net Unmask What Classical U-Net Cannot?

Ali Magzari
EECE Department
The University of British Columbia
Vancouver, BC, Canada
amagzari@student.ubc.ca

Abstract

Understanding urban scenery is critical in a number of areas, the most common of which is autonomous driving. Semantic segmentation specifically is the appropriate task in achieving this purpose as it would allow the conversion of urban scenes into specific regions, each belonging to a particular class (pedestrian, vehicle, traffic sign, etc.). U-Net is a convolutional network that was proposed for biomedical image segmentation. The initial idea behind this paper was to conduct semantic segmentation on the Cityscapes dataset using the classical U-Net, and attempt to ameliorate its performance either by incorporating a Convolutional Block Attention Module (CBAM) or implementing the Attention U-Net architecture, proposed in *Learning Where to Look for the Pancreas*. The experiments show that the three aforementioned network variations do not result in a significant performance improvement. This paper details the methodology and results supporting this statement.

1 Introduction and Background

Autonomous driving has been subject to extensive research due its numerous benefits such as mobility improvement and energy minimization [1]. The initial tasks of perception systems involved in self-driving vehicles are object recognition, localization, and more particularly, image segmentation [2]. Image segmentation is an operation that is highly computationally expensive as it requires assigning each pixel in the studied image to a specific class.

Cityscapes [3], a large-scale dataset consisting of 5000 pixel-level annotated frames portraying street scenes consisting of 30 visual classes from 50 different cities is the chosen dataset to evaluate the performance of the U-Net architecture [4] when extended to conduct multiclass semantic segmentation as its basic version targets binary segmentation.

The Convolutional Block Attention Module [5] is incorporated after every intermediate U-net feature map to seek improvement. The Attention U-Net [6], a U-net architecture enhanced with attention gates is also evaluated in the hope of ameliorating network performance. Although the dataset is prepared for instance-level labeling, the scope of this paper limits itself to semantic segmentation.

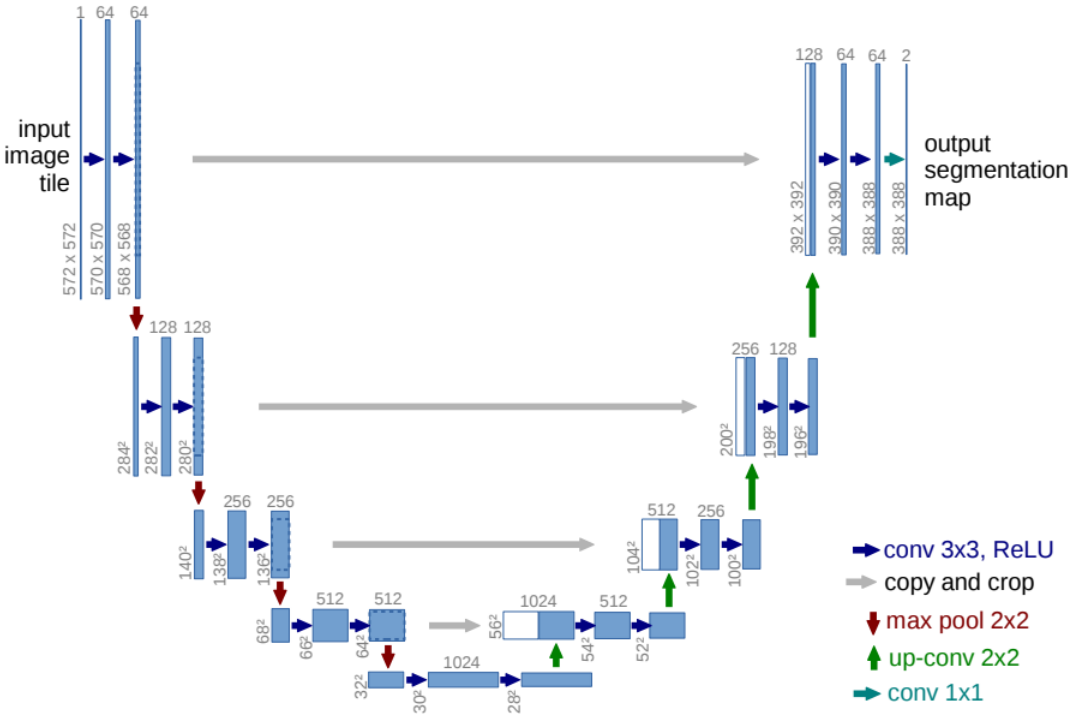
EECE 570 (2022-2023), UBC.

2 Related Work

2.1 U-Net

U-Net is a convolutional network that was initially proposed for biomedical image segmentation [4]. It consists of two parts: a contracting path that captures context, and an expanding path that enables precise localization, thus resembling a U-shape to which it owes its name (Figure 1). The primary block of the contracting path is composed of a 3x3 double un-padded convolution followed by a rectified linear unit (ReLU), and a 2x2 max-pooling operation with a stride of 2. The former assists in increasing the depth of the input image while the latter focuses on down-sampling its size by a factor of 2. At the end of the contracting phase, the input is transformed into 512 32x32 feature maps, upon which they pass through a bottleneck where they undergo a double convolution to see their number double to 1024. They are then inputted into the symmetrical expanding path, which primary building block consists of a 2x2 up-convolution that doubles the size and halves the number of feature maps, a concatenation with the cropped maps resulting from the corresponding contracting step, and a double convolution. The final output of the network is a segmentation map with two channels, as is needed for a binary segmentation task. This architecture generated the best Intersection-over-Union (IoU) score on the ISBI cell tracking challenge in 2015: 0.9203 for *PhC-U373*, and 0.7756 for *DIC-HeLa*.

Figure 1: U-Net architecture [4]

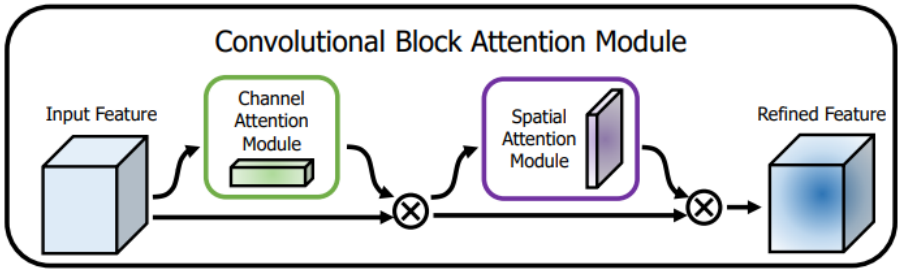


2.2 Convolutional Block Attention Module (CBAM)

The Convolutional Block Attention Module (CBAM) is a lightweight and effective attention module that was proposed for feed-forward convolutional neural networks [5]. It is applied to intermediate feature maps to infer both channel and spatial attention. As shown in Figure 2, the input feature is first subjected to a Channel Attention Module (CAM) that produces a mask with which the input is multiplied, thus generating channel-wise refined features. These features constitute a feature map volume upon which a Spatial Attention Module (SAM) is applied, producing a mask that is multiplied with the input volume to generate refined feature maps.

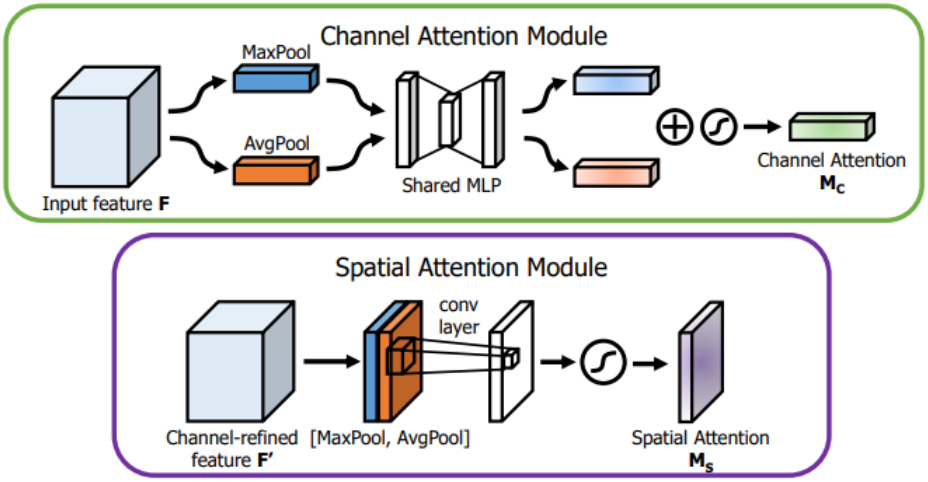
The CAM (Figure 3) utilizes both average and max-pooling to aggregate spatial information, which generates two spatial context descriptors that are forwarded to a shared multi-layer perceptron (MLP).

Figure 2: CBAM Architecture [5]



The resulting descriptors are then added and undergo a Sigmoid function that generates channel attention map. The SAM (Figure 3), on the other hand, applies average and max-pooling along the channel dimension to highlight informative regions. The resulting maps are concatenated and convolved to generate a spatial attention map.

Figure 3: CAM and SAM Architectures [5]



Woo et al. [5] conducted a thorough ablation study to determine the best order of the sub-modules (CAM and SAM) and their characteristics. They enhanced ResNET by integrating a CBAM within the ResBlock, and the results suggest that channel attention followed by spatial attention module (average and max-pooling, k=7) yielded the lowest Top-1 and Top-2 error percentages on the ImageNet-1K dataset, 22.80% and 6.52% respectively.

2.3 Attention U-Net

Attention U-net [6] leverages the contracting and expanding paths of the classical U-net where input images are progressively convoluted and downsampled while the encoder feature maps are concatenated with their corresponding decoder maps (Figures 4 and 5). The major difference is that the encoder feature maps, along the decoder maps (gating signal from a coarser scale) are forwarded to an attention gate (AG) before concatenation. This serves to identify salient image regions, compute a corresponding attention coefficient that multiplies the input feature maps (element-wise) to only preserve relevant activations. Experiments on the CT-150 dataset show that Att U-net yields a statistically significant perfomance improvement relative to the *Pancreas* Dice Score (DSC), Recall and Surface-to-Surface Distances (S2S) in the 120/30 train-test split: 0.840 ± 0.087, 0.841 ± 0.092 and 1.920 ± 1.284 compared to Standard U-Net results: 0.814 ± 0.116, 0.806 ± 0.126 and 2.358 ± 1.464, respectively. However, slight or stastiscally insignificant improvements were noticed regarding *Spleen* and *Kidney* scores.

Figure 4: Attention U-Net Architecture [6]

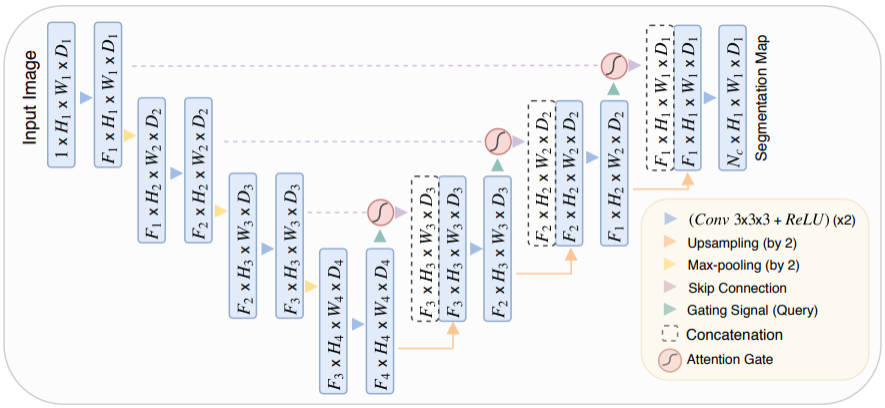
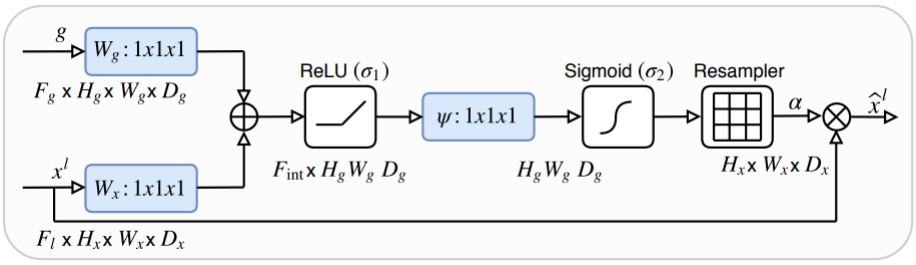


Figure 5: Attention Gate in Attention U-Net [6]



3 Methodology

This section details three variations of the original U-Net architecture. The first variation (Figure 6) contains minor changes. The convolutions are performed with padding and stride of 1 to preserve the image size (height and width), allowing the concatenation of the encoder feature maps with their corresponding decoder maps without need for cropping. Finally, the outputted segmentation map contained 20 channels to take into account all of the 20 classes of interest.

Figure 6: U-Net for Multiclass segmentation

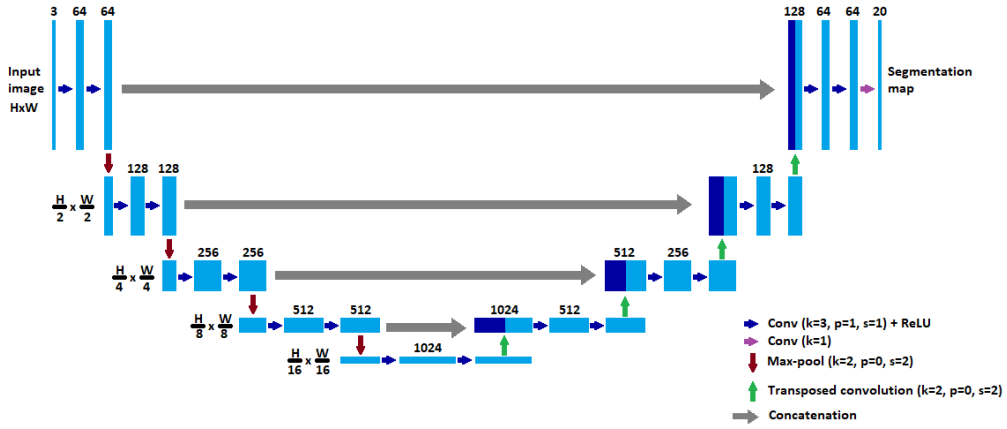


Figure 7: UNet-CBAM

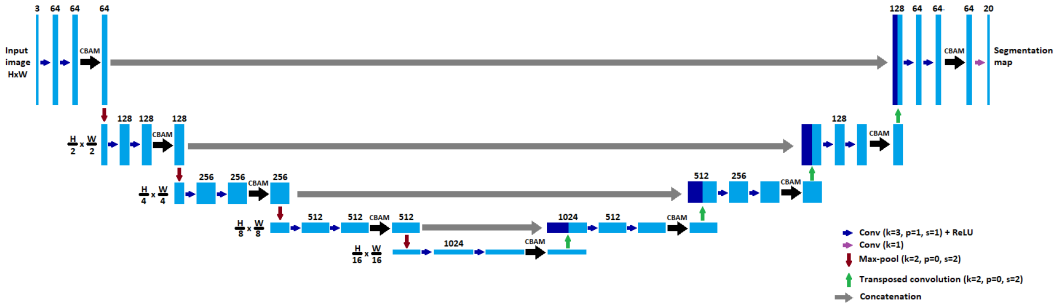
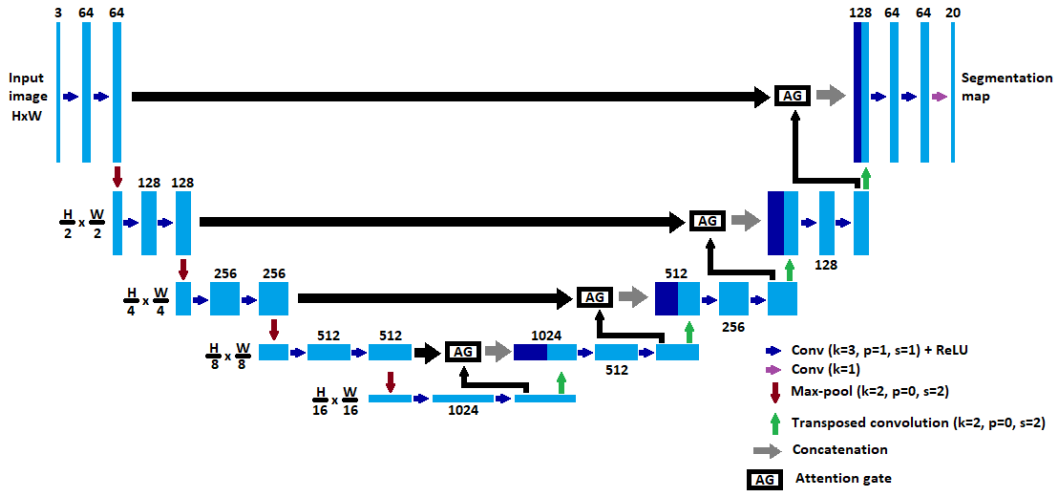


Figure 8: Attention U-Net



The second variation, UNet-CBAM applies a Convolutional Block Attention Module (CBAM) after every double convolution present in the encoder, bottleneck and decoder (Figure 7), while the third implementation is a direct integration of Attention Gates (Figure 5) in our first variation of the U-Net, as shown in Figure 8.

4 Experiments and Results

This section covers in detail the experimental setup, training hyperparameters, numerical and visual results.

4.1 Dataset

Cityscapes is an urban scenery segmentation dataset [3] containing three different folds: 2975, 500 and 1525 1024x2048 pixel-level annotated training frames for train, validation and test set respectively. The dataset contains 35 classes, 16 of which are ignored in the evaluation process, as it is stated in the labeling script [7] of Cityscapes official Github page [8]. In this project, those labels are encoded as a train ID of 0. The test annotations are private, and 1024x2048 predicted masks have to be upload to the Cityscapes server for evaluation. To reduce computational expenses, our models are trained on 256x512 resized images, they naturally output 256x512 masks. We noticed that up-sampling the results to a higher resolution render low-quality images, and therefore decided to not participate in the on-line server evaluation. We therefore re-arranged the data split as follow:

- Training: 2705 image and mask instances
- Validation: 500

- 270 (from the cities of Aachen and Bochum)

We also do not ignore any of the classes during evaluation, and use the training ID (Figure 4 in Appendix), thus amounting the number of semantic classes to 20 (0-19).

4.2 Setup

The following hyper-parameters are used for training:

- Loss function: Cross-Entropy (*torch.nn.CrossEntropyLoss*)
- Evaluation metric: Weighted Jaccard Index, also known as weighted Intersection-over-Union (wIoU) using *torchmetrics.JaccardIndex*
- Number of epochs: 50
- Optimizer: AdamW (*torch.optim.AdamW*)
- Starting learning rate: 0.01
- Learning rate reduction (*torch.optim.lr_scheduler.ReduceLROnPlateau*) by half (*factor* of 0.5) when the validation score stops improving for 3 consecutive epochs (*patience* value).
- Batch size: 16
- Transforms: Image and mask resizing to 256x512 to speed training

4.3 Results and Discussion

4.3.1 Train and Validation Results

Starting around the 15th epoch onward, the The Attention U-Net training loss decreases further than the loss registered by the U-net and Unet-CBAM (Figure 9). However, Attention U-Net does not seem to generalize well as its validation loss is higher than its counterparts. The training weighted Jaccard index slightly improves with Attention U-Net while it degrades using CBAM (Figure 9). However, after the 20th epoch mark, all of the three models yield a similar validation Jaccard index. As a matter of fact, at the end of training, the validation Jaccard indices are **0.7684**, **0.7694** and **0.76915** for U-net, UNet-CBAM and Attention U-Net respectively. It is deplorable to report that neither CBAM nor Attention U-Net yield significantly better results in comparison with classical U-Net. Training and validation results are further detailed in Table 1.

Figure 9: **(a):** Crosss-Entropy Loss Curves, **(b):** Weighted Jaccard Index

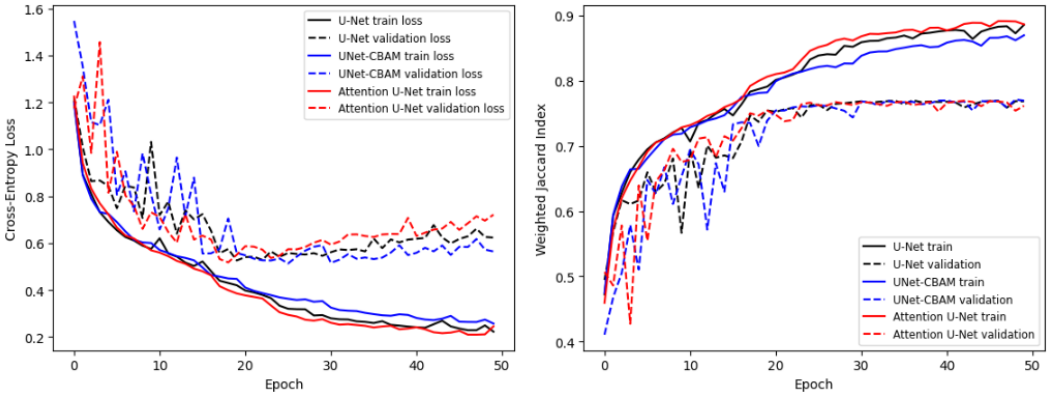


Table 1: Train and Validation Results

Model	Train Loss	Validation Loss	Train Jaccard Index	Validation Jaccard Index
U-Net	**0.2226**	0.6234	0.8857	0.7684
UNet-CBAM	0.2570	**0.5645**	0.8697	**0.7694**
Attention U-Net	0.2458	0.7215	**0.8863**	0.7615

Figure 10 illustrates the IoU score corresponding to each class, according to the train IDs listed in Table 4. It is clear that some classes (such as 0, 1, 2, 3, 9, 11 and 14) score considerably higher that others (7, 13, etc.). The direct reason one could assume is the class imbalance present in the training set, and nature in general. Road (0), vegetation (9) and sky (11) are intrinsically larger in size compared to traffic lights (7) for instance. This size difference translates naturally in the number of pixels occupied by each class, which in turn impacts the model ability to segment individual classes. To test this hypothesis, the class pixel-density distribution present in the training set is depicted in Figure 11 It is clear that classes of larger pixel attributions benefit from a higher Jaccard index. To provide a deeper look, Tables 5, 6, 7, 8, 9 and 10 in the Appendix detail the information contained in Figures 10 and 11.

Figure 10: **(a):** Train Jaccard Index, **(b):** Validation Jaccard Index

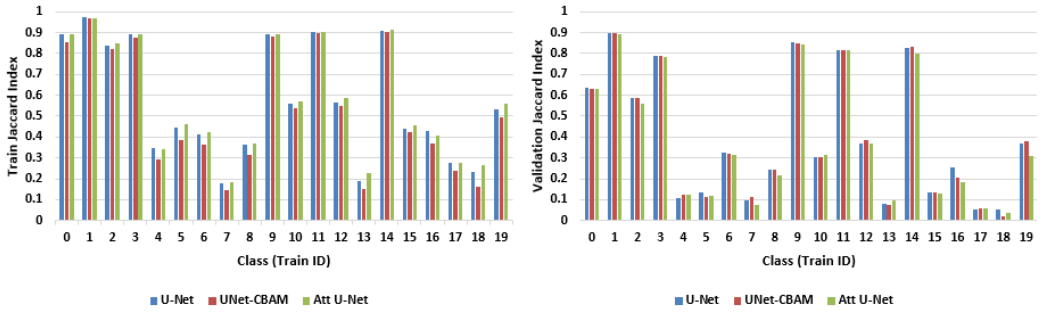
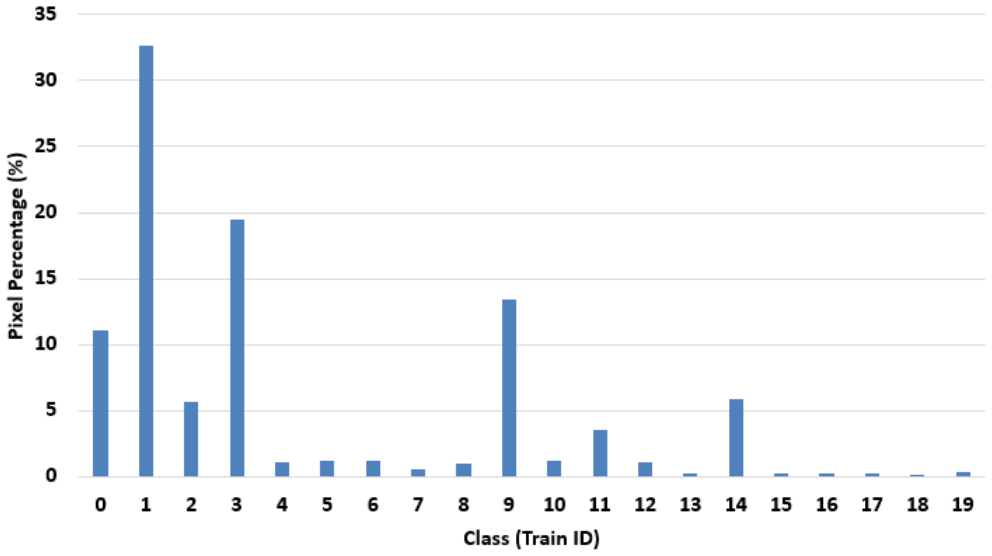


Figure 11: Train Set Class Pixel-Distribution



The trained models are loaded with their best weights (yielding highest validation score), and infered on a sample of the validation set. The visual results are displayed in Figures 12, 13 and 14. As the aforementioned numerical and graphical results suggest, U-Net, UNet-CBAM and Attention U-Net render predicted masks without notable differences. Note that pixel-dense classes such as roads (1), sidewalks (2), vegetation (9) and sky (11) benefit from a nearly-perfect segmentation, while pixel-frail and under-represented classes such as traffic signs (8), busses (16), rider (13) and bicycle (19) do not.

Figure 12: Sample of U-Net Visual Results on Validation Set

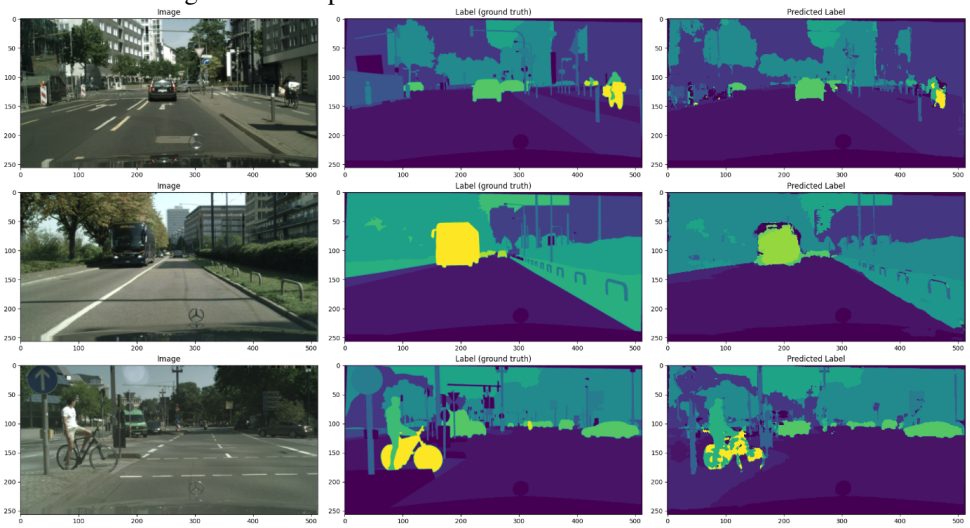


Figure 13: Sample of UNet-CBAM Visual Results on Validation Set

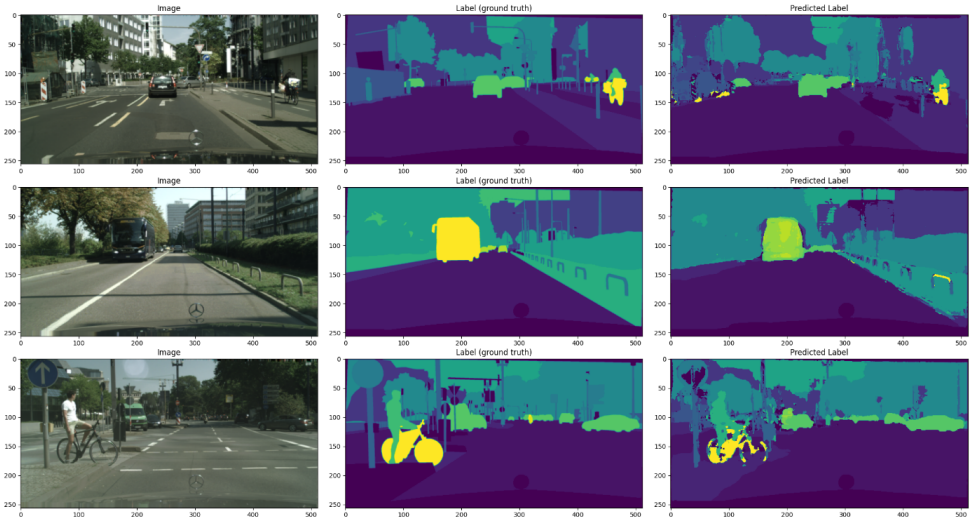
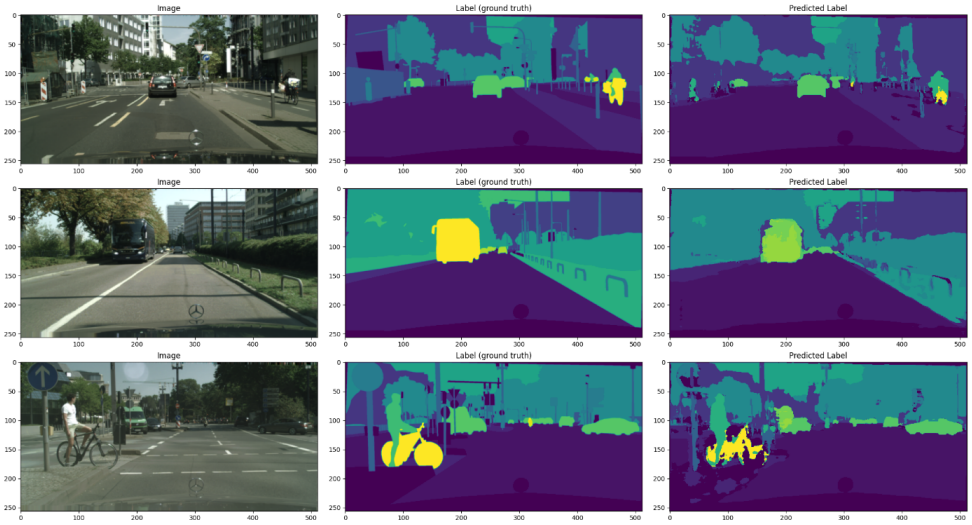


Figure 14: Sample of Attention U-Net Visual Results on Validation Set



4.3.2 Test Results

The UNet-CBAM is loaded with its best weights, and chosen to infer on the test set. The model generates a weighted Jaccard Index of **0.7897**, the distribution of which is listed in Tables 2 and 3. Visual results are shown in Figure 15. Both numerical and visual findings corroborate the class pixel-distribution phenomenon discussed in the previous section.

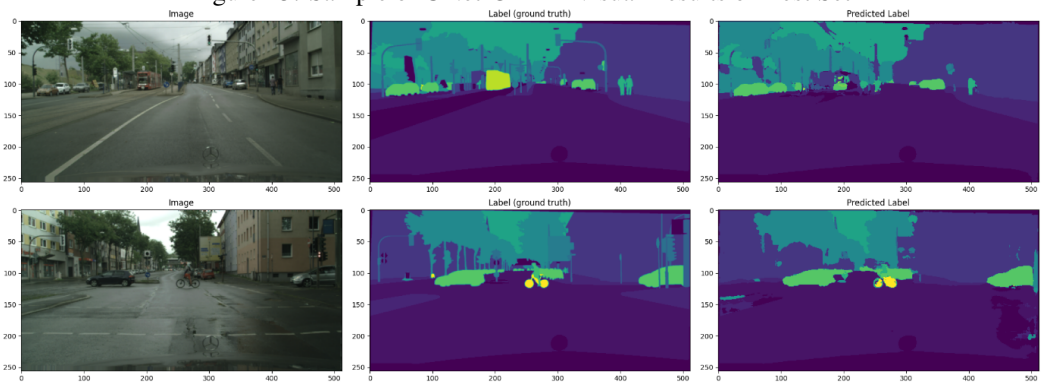
Table 2: UNet-CBAM Test Jaccard Index per Class (1/2)

0	1	2	3	4	5	6	7	8	9
0.6677	0.9264	0.6324	0.7933	0.0869	0.0764	0.2469	0.0796	0.1998	0.8291

Table 3: UNet-CBAM Test Jaccard Index per Class (2/2)

10	11	12	13	14	15	16	17	18	19
0.4099	0.8406	0.3256	0.0057	0.8586	0.0080	0.1180	0.0000	0.0120	0.1945

Figure 15: Sample of UNet-CBAM Visual Results on Test Set



4.4 Conclusion and Future Work

As the numerical, graphical and illustrative findings provided in this paper, it is deplorable to report that neither CBAM-enhancement nor Attention U-Net ameliorate the performance of the standard U-Net, which itself yielded satisfactory semantic segmentation results on the Cityscapes dataset. We also wish to re-iterate the negative impact of class imbalance on under-represented and pixel-frail classes.

Having said that, the following list comprises possible improvements to be carried in the future:

- Conduct experiments beyond 50 epochs
- Implement classical augmentation techniques, such as random cropping and horizontal flipping
- Implement class-specific augmentation techniques to fight class imbalance
- Propose a custom loss function that further penalizes incorrect segmentation of pixel-frail classes

References

[1] Kareem Othman. Exploring the implications of autonomous vehicles: a comprehensive review. 2022.

[2] Michael Treml, Jose Arjona-Medina, Thomas Unterthiner, Rupesh Durgesh, Felix Friedmann, Peter Schuberth, Andreas Mayr, Martin Heusel, Markus Hofmarcher, Michael Widrich, Bernhard Nessler, and Sepp Hochreiter. Speeding up semantic segmentation for autonomous driving. 12 2016.

[3] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016. URL `https://arxiv.org/abs/1604.01685`.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[5] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module, 2018.

[6] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention u-net: Learning where to look for the pancreas, 2018.

[7] Marius Cordts and Mohamed Omran, 2022. URL `https://github.com/mcordts/cityscapesScripts/blob/master/cityscapesscripts/helpers/labels.py`.

[8] Marius Cordts and Mohamed Omran, 2022. URL `https://github.com/mcordts/cityscapesScripts`.

Appendix

Table 4: Label Encoding

Name	ID	Category	Ignored in Evaluation	Train ID
unlabeled	0	void	True	0
ego vehicle	1	void	True	0
rectification border	2	void	True	0
out of roi	3	void	True	0
static	4	void	True	0
dynamic	5	void	True	0
ground	6	void	True	0
road	7	flat	False	1
sidewalk	8	flat	False	2
parking	9	flat	True	0
rail track	10	flat	True	0
building	11	construction	False	3
wall	12	construction	False	4
fence	13	construction	False	5
guard rail	14	construction	True	0
bridge	15	construction	True	0
tunnel	16	construction	True	0
pole	17	object	False	6
polegroup	17	object	True	0
traffic light	19	object	False	7
traffic sign	20	object	False	8
vegetation	21	nature	False	9
terrain	22	nature	False	10
sky	23	sky	False	11
person	24	human	False	12
rider	25	human	False	13
car	26	vehicle	False	14
truck	27	vehicle	False	15
bus	28	vehicle	False	16
caravan	29	vehicle	True	0
trailer	30	vehicle	True	0
train	31	vehicle	False	17
motorcycle	32	vehicle	False	18
bicycle	33	vehicle	False	19
license plate	-1	vehicle	True	0

Table 5: Train Jaccard Index per Class (1/2)

Model	0	1	2	3	4	5	6	7	8	9
U-Net	0.8919	0.9731	0.8389	0.8938	0.3462	0.4452	0.4104	0.1793	0.3627	0.8906
UNet-CBAM	0.8528	0.9696	0.8199	0.8744	0.2934	0.3874	0.3660	0.1460	0.3163	0.8804
Attention U-Net	0.8931	0.9698	0.8480	0.8924	0.3419	0.4630	0.4234	0.1867	0.3680	0.8896

Table 6: Train Jaccard Index per Class (2/2)

Model	10	11	12	13	14	15	16	17	18	19
U-Net	0.5621	0.9025	0.5660	0.1886	0.9107	0.4404	0.4268	0.2748	0.2356	0.5321
UNet-CBAM	0.5393	0.8952	0.5469	0.1496	0.9038	0.4231	0.3666	0.2400	0.1649	0.4959
Attention U-Net	0.5693	0.9052	0.5887	0.2281	0.9133	0.4581	0.4055	0.2787	0.2644	0.5610

Table 7: Validation Jaccard Index per Class (1/2)

Model	0	1	2	3	4	5	6	7	8	9
U-Net	0.6364	0.8950	0.5875	0.7864	0.1087	0.1368	0.3263	0.0981	0.2445	0.8524
UNet-CBAM	0.6326	0.8950	0.5892	0.7898	0.1257	0.1141	0.3199	0.1114	0.2441	0.8470
Attention U-Net	0.6329	0.8913	0.5625	0.7847	0.1216	0.1206	0.3141	0.0759	0.2179	0.8405

Table 8: Validation Jaccard Index per Class (2/2)

Model	10	11	12	13	14	15	16	17	18	19
U-Net	0.3031	0.8148	0.3686	0.0811	0.8205	0.1373	0.2572	0.0539	0.0540	0.3675
UNet-CBAM	0.3043	0.8176	0.3880	0.0749	0.8305	0.1354	0.2073	0.0604	0.0217	0.3789
Attention U-Net	0.3150	0.8168	0.3715	0.0963	0.7970	0.1315	0.1852	0.0572	0.0395	0.3085

Table 9: Pixel Distribution in Training Set (1/2)

Class	0	1	2	3	4	5	6	7	8	9
Number of pixels	39070107	115627145	20153466	68886285	3901309	4063928	4082219	1849116	3250569	47520393
Pixel percentage	11.0196	32.6123	5.6842	19.4292	1.1003	1.1462	1.1513	0.5215	0.9168	13.4030

Table 10: Pixel Distribution in Training Set (2/2)

Class	10	11	12	13	14	15	16	17	18	19
Number of pixels	4092022	12667511	3699097	767910	20930487	924148	777710	828890	337714	1119734
Pixel percentage	1.1541	3.5728	1.0433	0.2165	5.9033	0.2606	0.2193	0.2337	0.0952	0.3158