2020 IEEE International Conference on
Autonomous Robot Systems and Competitions (ICARSC)
April 15-16, Ponta Delgada, Portugal

Performance Analysis on Deep Learning Semantic Segmentation with multivariate Training Procedures

Bernardo Lourenço
UA, DEM
Aveiro, Portugal
bernardo.lourenco@ua.pt

Vitor Santos
UA, DEM, IEETA
Aveiro, Portugal
vitor@ua.pt

Miguel Oliveira
UA, DEM, IEETA
Aveiro, Portugal
mriem@ua.pt

Tiago Almeida
UA, DEM, IEETA
Aveiro, Portugal
tm.almeida@ua.pt

***Abstract*—Deep Learning approaches are becoming ubiquitous in many fields of computation, especially for tasks of object detection and classification in images. However, the diversity of architectures, the number and range of training parameters, and the dimension and representativeness of datasets make it very complex to define a unified method or set of techniques to address diverse problems, and it is quite common for the literature to skip valuable implementation details. One of these fields is the Semantic Segmentation with many applications like autonomous driving. In that line, this paper carried out a study using an ENet architecture for semantic segmentation on road images for intelligent vehicles and multiple experiments of deep learning training with a wealth of combinations of training procedures, including multiple loss functions and datasets for out-of-domain performance assessment. Besides some expected outcomes, the results allow identifying a set of combinations with performances that stand out and are advisable for researchers to tackle more efficiently problems of a similar nature.**

***Index Terms*—Deep Learning, Semantic Segmentation, Convolutional Neural Networks**

I. Introduction

Semantic segmentation is the task of assigning a label to each pixel in an image. This is a fundamental task in computer vision and can be applied in various fields such as augmented reality [1], robotics [2]–[4], and autonomous driving [5], [6].

Due to the nature of this task, several challenges are currently driving the design of new models, such as:

1) A small output feature resolution: The repeated combinations of pooling and strided convolutions performed consecutively by convolutional neural networks are essential for a large receptive field. A large receptive field allows for the network to capture enough context, however, a low-resolution feature mapping requires more upsampling for the final prediction on the fully-connected neural network [7].
2) Recover spatial information: The successive convolutions designed to capture context require to be invariant to spatial transformations [8]. Therefore, the final feature mappings have limited information about location, which in turn is required to reconstruct the fine boundaries of objects in the final output. Often, the models use the mappings from the previous layers, called "hypercolumn features" for aggregation in the final prediction, to recover the lost information [9].
3) Existence of objects at multiple scales: Images for semantic segmentation have multiple objects. For example, in road segmentation, the size of the objects (sky, cars, building, pedestrians) have a wide range. To recognize objects, models compute features at multiple scales, with techniques such as cascading features [10], pyramid pooling [11] or branching [12].

The acknowledgments of these challenges and newer model architectures have set a trend of model accuracy improvement. For example, the mean Intersection-over-Union (mIoU) of the popular Cityscapes dataset has been consistently rising in recent years from $58\,\%$ in [9] to about $80\,\%$ [13], [14] for models for real-time segmentation.

However, these advancements did not come solely from improvements in model architectures. Training procedures, such as a choice of loss functions, data processing, and optimization methods also played a major role. Unfortunately, these training procedures are rarely detailed in the literature, which hampers the possibility of reproducing the results claimed by such works.

Also, semantic segmentation datasets are sparse and contain a small number of images, because pixel-wise annotations are time-intensive and costly [15]. This makes it hard to generalize from such datasets, which causes a gap between the accuracy assessed in data from the datasets and measured in real-world data. However, datasets created with new annotation and acquisition techniques have increasingly more data, which in turn could allow for a better generalization.

In this work, a baseline training technique will be set up for training and present several improvements or changes to this technique, to evaluate if a better performance and better metrics arise from these changes. Finally, the models trained using different datasets will be compared in real-world examples.

Both models and training methods are implemented using the *PyTorch* framework and are publically available online[1].

II. Related Work

The advancement of training techniques is one of the keys to the success of deep neural networks (DNN). One of the beliefs in deep learning is the learning capacity and performance of

[1]Available at github.com/bernardomig/torch_semantic_segmentation

DNNs is only limited by its number of parameters and depth. However, the training is often the limiting factor, because large networks suffer from overfitting, are slower and require more memory, and the convergence is not guaranteed. On one hand, small and shallow networks are easier to train but have limited performance. One the other hand, larger and deeper networks have more potential but training is more difficult, which in turn hampers the performance of these DNNs.

The current research tackles this problem in two strands: 1) optimizing the design of DNNs, and 2) improving the current training techniques.

The first approach, present in works such as [12], [16], [17], modify the topology and the layers of neural networks, to achieve high performance with a fraction of the parameters. The main advantage of these models is that they do not suffer from overfitting and have low computational cost.

The second approach focuses on techniques that are present during training, but not during inference. These techniques can be grouped in several classes: regularization, optimization and efficiency methods. Regularization techniques add constraints during training, to reduce overfitting, or guide the learning through a manifold of the possible parameter space. For example, apart from the weight decay and dropout, new regularization techniques include deep supervision [18], knowledge distillation [19], Mixup [20], MixCut [21] or Stochastic Depth [22]. Optimization techniques study how to update the parameters of the network. Currently, a high plethora of optimizers, such as the AdamW and RMSprop, and methods to change the learning rate during training, such as the Cosine Annealing [23] or the One-Step Learning [24] are studied, which can train the models in fewer steps. Finally, new methods study how to train the models efficiently with current hardware architectures. For example, mixed-precision training [25] is a technique for training with half-precision and [26] that presents a method to train DNNs with large batch sizes, for distributed training.

However, most of the methods are often applied only to the problem of classification, and it is not trivial how these advancements can be integrated into other problems, such as semantic segmentation.

This work is inspired by the methodology and unprecedented results presented in [27], which included tweaks to both the original ResNet [28] and the training methods. Using these tweaks improved the top-1 classification accuracy of the ResNet50 by 5%, surpassing the original ResNet101, which is a larger and deeper model, in comparison. Furthermore, the training techniques presented in [27] reduced the training times of the network, mostly by increasing the batch size and by using new features available in present hardware.

III. Training Procedures

The training procedures presented in this paper have the objective of improving the training, for example, reducing the time required to perform a forward pass and increasing the batch size, as well as the final model, by reducing the overfitting and improving generalization. The training techniques proposed are data augmentation, mixup, progressive scaling, and mixed precision.

A. One Cycle Learning

One of the challenges of Deep Learning is the careful choice of training hyperparameters, such as the learning rate or momentum. In the area of semantic segmentation, most works choose adaptative optimizers, such as Adam of AdaGrad, with a poly learning rate scheduler [17]. However, this method is known to perform poorly [29]. On the other hand, the standard SGD achieves better results with careful tuning of parameters and parameter scheduling [23]. In [24], a method is proposed to determine the learning rate and anneal the learning rate and momentum during training. This method can achieve improved results with a fraction of the training iterations.

B. Data Augmentation

The main enabler for Deep Learning is undoubtedly the scale of datasets available in recent years. However, data augmentation produces new examples in the vicinity of the original examples in an online fashion. It also helps generalization, so that models perform equally well on new data. During training, for this paper, the following steps are performed:

- Randomly sample an example (x_i, y_i) from the training dataset;
- Scale the image by a factor s randomly sampled from a uniform distribution $\mathrm{Uniform}\,(s_{min}, s_{max})$;
- Crop a patch from the scaled image using a fixed crop size $W \times H$;
- Randomly flip the image horizontally;
- Finally, normalize the RGB channels by subtracting the mean and the standard deviation[2].

During validation, the images are only normalized similarly to the last step on the training augmentation.

Some hyperparameters are crucial to this process. The crop size and the scale factor must be carefully chosen as they play a major role in the model generalization and training times. The principles for crop size and scale factor choices are as follows: 1) A large crop size contains more context, so the performance of a model trained on large scales should, in theory, perform better than one trained on scaler crop sizes. 2) Unfortunately, training on large crop sizes is significantly more memory and computational demanding. Thus models train slower and with smaller batch sizes. This could have a negative impact, as Batch Normalization is especially sensitive to small batch sizes, and class imbalance has a stronger effect. 3) Also, as noted in [31], the input size has a small impact on the learning rate and the choice of the optimizer. Therefore, small input sizes are optimal for quick experimentation of these hyperparameters. 4) Finally, the scale limits should be carefully chosen. They should have a high range so that objects appear at multiple scales. Also, the minimum scale should be the minimum possible for maximum context. This

[2]The mean and standard deviation used for image tasks is calculated from the ImageNet dataset [30].

could be especially important for models specially designed for context aggregation, for example, models with pyramid pooling modules.

C. Mixup training

Mixup [20] is a training technique that was successfully employed for improving the accuracy of classification models. It is known that it has a regularization effect on training by forcing the neural network to learn linear behavior discriminating between different classes. This technique can be trivially applied for semantic segmentation.

In mixup training, new training examples $(\tilde{x}, \tilde{y})$ are generated as

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1-\lambda) x_j, \\ \tilde{y} &= \lambda y_i + (1-\lambda) y_j, \end{aligned} \tag{1}$$

where (x_i, y_i) and (x_j, y_j) are random image-label samples taken from the training set and λ is sampled from a $\mathrm{Beta}(\alpha, \alpha)$ distribution.

D. Progressive Scaling

As noted in Sec. III-B, the input size is an important parameter training fully-convolutional Neural Networks. This final accuracy and the IoU metrics are very dependent on this parameter and, at the same time, have an impact on the memory and computational cost.

Progressive scaling is a training technique that processively increases the input size and, accordingly, the batch size and learning rate, to train models faster and with stable convergence. In the early stages of training, with low size input, and large batch sizes and learning rate, the network stabilizes training and the model learns to capture context and detect larger objects, while in the end, with larger size input and small batch sizes and low learning rate, the network learns finer details and smaller objects.

E. Mixed Precision Training

Usually, the training of neural networks is done using 32bit floating-point operations. In recent years, advances in GPU architectures have developed the half-precision (16bit floating-point) operations on this special hardware. Because half-precision has more throughput and is less memory intensive, training times are reduced as well as memory usage.

Training in half-precision can, therefore be faster and allow for larger batch sizes. However, as studied in [25], training solely in half-precision is not numerically stable so a mix of both 32 and 16bit is required, depending on the operations. This training technique is called Mixed Precision Training.

Moreover, previous work has measured a regularization effect with this technique, due to the noise introduced by the loss of precision, during numerical operations and casts.

IV. Loss functions for semantic segmentation

In supervised learning, the loss function is a distance that measures the difference between a prediction and the label (or ground truth). The loss function plays an essential role in training, as it has to reflect the problem that is going to be optimized, in a differentiable and convex scalar value. In the case of semantic segmentation, the loss function should measure the prediction error for objects at multiple scales, account for the class invariance, and force the model to learn the shapes and boundaries of the objects.

In this work, a survey of the loss functions used for semantic segmentation is reviewed and applied in the training of a model, to compare the different approaches.

A. Review of the loss functions for Semantic Segmentation

In this section, a systematic review of the loss function applied to semantic segmentation is done. For this review, y is the final prediction of the model, after the sigmoid activation, and $\hat{y}$ is the one-hot encoded label or ground truth.

In all the presented equations, i and N represents the pixels in each image and its number, and c and C represents each class and the total number of classes, respectively.

1) Cross-Entropy Loss: The Cross-Entropy loss (CE) is the most commonly used function in classification problems. This function evaluates each pixel individualy and evaluates the loss as

$$\mathrm{CE}(y, \hat{y}) = -\frac{1}{N} \sum_{i}^{N} \sum_{c}^{C} \hat{y}_{i,c} \log y_{i,c}. \tag{2}$$

2) Balanced CE Loss: Because each pixel has an equal contribution to the loss, the CE loss is highly susceptible to class imbalance, as it is the case in semantic segmentation problems. Balanced Cross-Entropy loss (BCE) tries to cope with this problem by adding a weight factor w to the CE loss, based on the class frequency on the dataset, as

$$\mathrm{BCE}(y, \hat{y}) = -\frac{1}{N} \sum_{i}^{N} \sum_{c}^{C} w_c \ \hat{y}_{i,c} \ \log y_{i,c}. \tag{3}$$

This way, objects with low frequency contribute more to the loss. The weight factor can be computed by multiple methods. For this work, the one presented in [17] was used:

$$w_c = \frac{1}{\log\left(1.02 + f_c\right)}, \tag{4}$$

where f_c is the class frequency for class c.

3) Focal Loss: Similarly to the BCE loss, the Focal Loss (FL) introduces a factor $(1-y)^\gamma$ to the standard CE, to mitigate the problem of class imbalance. For $\gamma > 0$, the contribution to the loss of miss-classified examples is larger, so the training puts more focus on them. The loss equation is as follows:

$$\mathrm{FL}(y, \hat{y}) = -\frac{1}{N} \sum_{i}^{N} \sum_{c}^{C} (1 - y_{i,c})^\gamma \ \hat{y}_{i,c} \ \log y_{i,c}. \tag{5}$$

4) Online Hard Example Mining: OHEM loss uses the partial cross-entropy losses $l_i \in \mathcal{L}$ sorted in descending order, so that $l_i >= l_j$, for $i < j$. This loss is composed by the mean of the subset

$$\mathcal{L}_p = \{l_i \mid l_i \in \mathcal{L},\ l_i > t\} \cup \{l_i \mid l_i \in \mathcal{L},\ i < N_p\}, \tag{6}$$

where t and N_p are two hyperparameters that define the minimum threshold and the minimum number of pixels that

contribute for the loss. Usualy N_p is defined by the factor $p = N_p/N$, where N is the total number of points of the batch.

This loss has the property to force the network to train on hard examples and ignores easy contributions to the loss.

5) Soft Dice Loss: The Soft Dice Loss (Dice) is based on the Dice coefficient, which measures the overlap between two binary samples. The Dice loss, however, directly uses the prediction and therefore is differentiable. The equation for this loss is expressed as

$$DICE(y, \hat{y}) = 1 - \sum_{c}^{C} \left[\frac{2 \sum_{i}^{N} y_{i,c} \, \hat{y}_{i,c}}{\sum_{i}^{N} y_{i,c} + \sum_{i}^{N} \hat{y}_{i,c}} \right] \quad (7)$$

One of the advantages of the Dice loss is that each class has an equal contribution to the loss, so class imbalance is mitigated.

V. Datasets and Out-of-domain Performance

Datasets contain the data for supervised training. In the case of semantic segmentation for autonomous driving, data acquisition is quite easy to perform and plentiful. However the labeling process is slow and expensive, so the available datasets have a limited number of examples and are often impoverished in terms of the variety of visual features and contextual information.

One of the problems with datasets with a reduced number of examples and visual content is that models perform well in the domain of the dataset, but the out-of-domain (images from outside the dataset) performance is poor. For example, if the dataset has a few examples of images with rain, it is expected that the model would not perform well for raining conditions. One further problem is that, with a few examples, achieving generalization is significantly difficult.

With the advent of new and better tools for data annotation [32], more modern datasets have more labeled images and have a diverse set of geography, weather conditions and environmental scenarios. These new datasets can reduce the gap between the domains.

In this work, we will evaluate if the training with modern datasets has a noticeable impact on the performance of the models in images taken from local roads in Aveiro, Portugal.

A. The Datasets for road semantic segmentation

The datasets used in this work were the Camvid [33], the CityScapes [34], and the Berkeley DeepDrive [32] dataset. Each one of the datasets will be presented next.

1) CamVid: CamVid is a small road segmentation dataset [33] that consists of only 600 images for training and testing. There are eleven different classes such as building, car, sky or road. The resolution of the images is 960×720. This dataset has the disadvantage to be very small and the images are of low quality so it is not appropriate for training a model from scratch.

2) CityScapes: This dataset [34] consists of 5000 fine-annotated images, acquired in multiple cities in Germany. CityScapes has 19 classes for training that are expected most in road scenarios. The high resolution of the images (2048×1024), high quality of the images, and high variety of the scenes made this dataset one of the most important for road segmentation.

3) Berkeley DeepDrive: DBB100k [32] is one of the largest public driving datasets with 100K $720p$ videos captured from diverse locations in the USA, with different equipment, mainly cellphones. This dataset features multiple scenes, weather conditions and time of day. For the task of segmentation, a subset of 10K images is fine-annotated with labels compatible with the CityScapes dataset.

VI. Results

In this section, the results from the different training techniques, loss functions, and datasets are shown.

A. Baseline

To compare the influence of each training technique, the ENet model [17] was trained with the CityScapes dataset. Unless otherwise noted, the model was trained with 512×512 crops with scaling from 0.5 up to 1.5 times, and no further data augmentation. Also, the training was done for 300 epochs with an SGD optimizer and a one-cycle learning rate scheduler. The batch size was set to 16, the learning rate was 10^{-2} and the weight decay was set to 10^{-4}. The default loss function was the BCE.

B. Influence of the crop size

For this study, the model was trained with 4 crop sizes: 256×256, 384×384, 512×512 and 768×768. The batch size was adjusted to maximize the memory usage of the GPU. It was found that the learning rate should stay constant, and the intuition is that the number of pixels in each batch is approximately constant. Also, the number of epochs was adjusted so that the convergence was possible.

The results of this study are condensed in Table I. Unsurprisingly, both mean-IoU and accuracy are higher for larger crop sizes. However, it is important to notice that the training times are much different. The reason is that the number of pixels in each batch is constant, and the time per iteration is similar for all crop sizes. However, the number of iterations is proportional to $epochs/batchsize$, so a smaller crop size takes only a portion of time in comparison to a larger one.

Table I
THE INFLUENCE OF THE CROP SIZE ON THE FINAL PERFORMANCE OF THE ENET MODEL.

Crop Size	Batch	LR	Epochs	mIoU	Acc
256×256	48	0.01	400	47.1	87.9
384×384	32	0.01	350	49.0	88.9
512×512	16	0.01	300	53.1	89.9
768×768	8	0.01	400	56.8	90.7

C. MixUp training

The MixUp technique was added to the baseline configuration, with $\alpha = 0.1$. Also, according to [20], the number of epochs was changed to 500 because, usually, mixup training requires more iterations to converge. Unfortunately, this technique did not improve performance or reduced overfitting (a result of 50.1 mIoU was achieved), so it is possible that is not adequate for the task of semantic segmentation.

D. Training procedures for training optimization

In this paper, several techniques for training optimization are proposed: Mixed Precision[3] (MP) and Progressive Scaling (PS). The objective of these techniques is to reduce training time, without loss of performance or changing the hardware. To test this proposition, the ENet model was trained for 300 epochs with a maximum crop size of 768×768 on a single *Nvidia RTX2080TI*. With mixed-precision, the number of images per batch could be increased to 16. Finally, the progressive scaling technique was implemented by increasing the batch size every 100 epochs and, at the same time, reducing the batch size. The results are shown in Table II.

Table II
CONFIGURATIONS AND RESULTS OF THE TRAINING PROCEDURES: BASELINE, MIXED PRECISION AND PROGRESSIVE SCALING.

Procedure	Crop Size	Batch Size	mIoU	Time
BL	768	8	56.8	12h30
BL + MP	768	16	52.4	8h30
BL + MP + PS	384, 512, 768	64, 32, 16	56.9	5h30

The results show that the preferred method is by using the combination of Mixed Precision and Progressive scaling. This configuration leads to a reduction of about 50% on the training time without a reduction of performance. The configuration with only Mixed Precision performed poorly because the number of training iteration was insufficient for learning with a large crop-size.

E. Influence of the loss functions

For this study, the ENet model was trained according to the baseline configuration using the loss functions presented in Section IV. Also, the learning rate has to be adjusted for each loss. Finally, the IoU was determined in the validation set for each class, as well as the mean-IoU. The results are presented in Table III. The class frequency (percentage of pixels per class in the dataset) was included for context.

A broad view of the results shows that both the dice loss and OHEM loss were the losses that achieved the best results, while both the cross-entropy and focal loss are demonstrably unusable for this problem. Intuitively these results can be attributed to two factors: the class frequency of each class in the dataset, and the presence of instances of each class in the dataset. For example, classes such as road, building, or sky are present in most images and have large representability. These classes are well classified by all models. On the contrary, classes that appear sparingly or have low frequency are the ones where most models performed poorly, in comparison.

[3]The mixed precision technique was implemented using the *Nvidia Apex* library

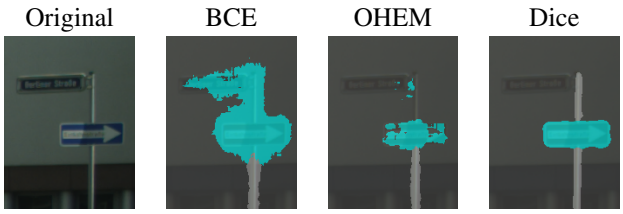

Figure 1. The effect of the loss function in the prediction of a small object.

Surprisingly, the OHEM loss excels or performs well in most classes, because it adjusts automatically so that the model trains in classes that have a higher loss. This loss function has several advantages. First, it can be used to fine-tune models that were trained using other cross-entropy losses. Also, it does not require high customization and adjusts automatically prediction of the model and training examples. In contrast, the balanced CE performance is highly dependent on the choice of weights used for training. Despite having a higher mIoU, the OHEM surpassed the BCE in all classes except for two.

The Dice loss was the loss function that outperformed the other losses, with a mIoU of 53.8. This result is explained by the fact that the Dice loss optimized for all classes equally, in contrast to the other losses, where the contribution is pixel-wise. This reasoning explains why the Dice loss achieves the best results for the most unrepresented classes and why, conversely, it does not perform best for classes with a large representation. Curiously, the model trained with the Dice loss was unable to detect any truck or train classes in the entire dataset. This can be due to insufficient activation of the neurons responsible for this class, which were then suppressed by the weight decay present in training. Also, a disadvantage of the dice loss is its compatibility with the other losses, so a fine-tuning of existent models using this loss is not possible.

Figure 1 shows the detection of a small road sign object, for a comparison of the loss functions BCE, OHEM and Dice.

F. Datasets and Out-of-domain performance

One of the major challenges in deep learning is the generalization from the datasets to the real-world. This is commonly referred as out-of-domain performance. To evaluate the out-of-domain performance for the three datasets presented in Section V, the ENet model was trained using the training technique with each dataset. For the evaluation, the models trained on the three datasets were applied to images acquired from local roads in Aveiro. Results are shown in Figure 2.

As can be observed, the results are increasingly better from the CamVid dataset to the CityScapes, and then the DeepDrive, which was the one that performed best by visual inspection.

The main observation regarding these results is described next. First, the model trained with the CamVid dataset has a large number of artifacts, for example, the gridding and

Table III
PERFORMANCE OF THE ENET MODEL USING DIFFERENT LOSS FUNCTIONS FOR THE CITYSCAPES VALIDATION SET. ALL NUMBERS ARE PERCENTAGES. HIGHEST SCORES FOR EACH CATEGORY ARE HIGHLIGHTED.

Loss Function	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic Light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motocycle	Bicycle	mIoU
Cross Entropy	**96**	70	85	23	29	43	6	44	88	48	88	54	<1	85	2	30	15	<1	50	45.0
Balanced CE	94	68	82	33	35	42	35	48	85	45	87	59	37	86	**28**	44	15	**27**	57	53.1
OHEM	96	**73**	**86**	**37**	**38**	48	36	53	**89**	**51**	88	62	14	**88**	9	**46**	**20**	9	61	52.8
Dice	94	68	84	29	38	**49**	**49**	**58**	87	46	**90**	**65**	**46**	84	0	44	0	26	**65**	**53.8**
Focal Loss	96	72	85	31	33	44	29	49	89	49	89	57	6	86	12	40	19	4	57	49.7
Class Frequency	37	6	23	1	1	1	<1	1	16	1	4	1	<1	7	<1	<1	<1	<1	<1	

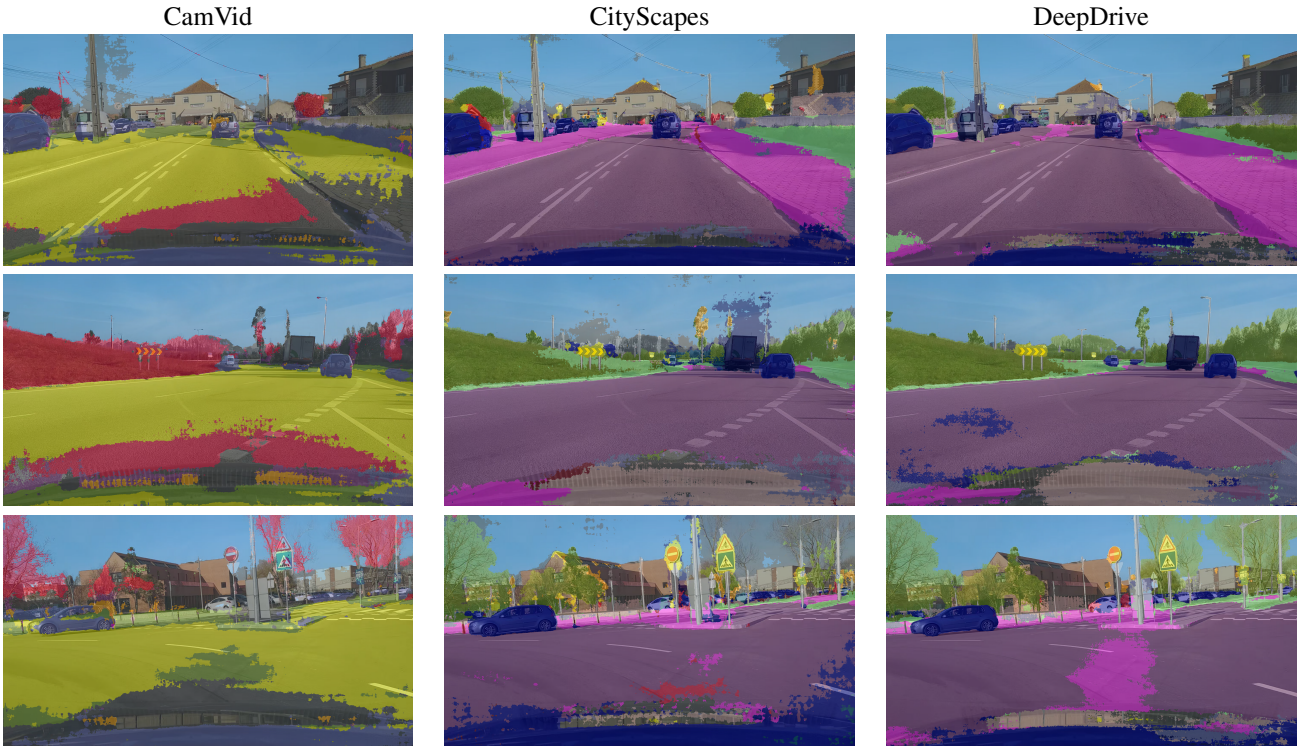

Figure 2. The visual results for the out-of-domain segmentation, on images not present in any dataset. Each row contains a frame that was classified by each one of the models trained. Best viewed with color.

checkerboard pattern. These artifacts are originated by the dilated convolutions present in the ENet. Both the models trained with the other datasets present these artifacts, but sparingly. Also, the confusion between classes is present in all models. For example, the model trained with CityScapes often confuses the sky with other classes. Likewise, the model trained with DeepDrive often confuses the sidewalk with the road. Further on, the boundaries between each object are significantly better in the model trained with Deep Drive. For example, the boundary between traffic signs or cars is noticeably better than with other models. This analysis shows that the performance of the model, for real-world scenarios, can be highly dependent, not only on the model but also on the dataset used for training. The choice of the dataset can have a determinant impact on the performance of the models.

VII. CONCLUSIONS

In this paper, a series of techniques were surveyed to train deep learning semantic segmentation models, for road segmentation. These techniques were entirely focused on the training, and introduced changes in the loss function, datasets, optimizer and learning rate schedulers, data augmentation and other techniques that focused on reducing training times. It is shown that these techniques are an important factor for model accuracy and out-of-domain performance, and they are advisable for researchers training models for semantic segmentation tasks.

ACKNOWLEDGEMENTS

This work was supported by the SeaAI-FA_02_2017_011 project.

REFERENCES

[1] L. Chen, W. Tang, N. W. John, T. R. Wan, and J. J. Zhang, "Context-aware mixed reality: A framework for ubiquitous interaction," *CoRR*, vol. abs/1803.05541, 2018.

[2] J. C. Balloch, V. Agrawal, I. Essa, and S. Chernova, "Unbiasing semantic segmentation for robot perception using synthetic data feature transfer," *CoRR*, vol. abs/1809.03676, 2018.

[3] X. Zhi, X. He, and S. Schwertfeger, "Learning autonomous exploration and mapping with semantic vision," *CoRR*, vol. abs/1901.04782, 2019.

[4] W. Kim and J. Seok, "Indoor semantic segmentation for robot navigating on mobile," in *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2018, pp. 22–25.

[5] Z. Wang, W. Ren, and Q. Qiu, "Lanenet: Real-time lane detection networks for autonomous driving," *CoRR*, vol. abs/1807.01726, 2018.

[6] B. Tan, N. Xu, and B. Kong, "Autonomous driving in reality with reinforcement learning and image translation," *CoRR*, vol. abs/1801.05299, 2018.

[7] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *ICLR*, 2016.

[8] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017.

[9] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *CoRR*, vol. abs/1511.00561, 2015.

[10] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," *CoRR*, vol. abs/1704.08545, 2017.

[11] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," *CoRR*, vol. abs/1612.01105, 2016.

[12] R. P. K. Poudel, S. Liwicki, and R. Cipolla, "Fast-scnn: Fast semantic segmentation network," *CoRR*, vol. abs/1902.04502, 2019.

[13] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," *CoRR*, vol. abs/1808.00897, 2018.

[14] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "Ccnet: Criss-cross attention for semantic segmentation," *CoRR*, vol. abs/1811.11721, 2018.

[15] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European Conference on Computer Vision (ECCV)*, ser. LNCS, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., vol. 9906. Springer International Publishing, 2016, pp. 102–118.

[16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[17] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *CoRR*, vol. abs/1606.02147, 2016.

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[19] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv e-prints*, Mar 2015.

[20] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *CoRR*, vol. abs/1710.09412, 2017.

[21] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," *CoRR*, vol. abs/1905.04899, 2019.

[22] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," *CoRR*, vol. abs/1603.09382, 2016.

[23] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay," *CoRR*, vol. abs/1803.09820, 2018.

[24] L. N. Smith and N. Topin, "Super-convergence: Very fast training of residual networks using large learning rates," *CoRR*, vol. abs/1708.07120, 2017.

[25] P. Micikevicius, S. Narang, J. Alben, G. F. Diamos, E. Elsen, D. García, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed precision training," *CoRR*, vol. abs/1710.03740, 2017.

[26] Y. You, I. Gitman, and B. Ginsburg, "Scaling SGD batch size to 32k for imagenet training," *CoRR*, vol. abs/1708.03888, 2017.

[27] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," *CoRR*, vol. abs/1812.01187, 2018.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[29] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The marginal value of adaptive gradient methods in machine learning," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4148–4158.

[30] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 248–255.

[31] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of imagenet as an alternative to the CIFAR datasets," *CoRR*, vol. abs/1707.08819, 2017.

[32] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "BDD100K: A diverse driving video database with scalable annotation tooling," *CoRR*, vol. abs/1805.04687, 2018.

[33] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recogn. Lett.*, vol. 30, no. 2, p. 88–97, Jan. 2009.

[34] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.