POLITECNICO DI TORINO

Master's Degree in
Computer Engeneering

Master's Degree Thesis

KNOWLEDGE DISTILLATION FOR SEMANTIC SEGMENTATION APPLIED TO AUTONOMOUS DRIVING IN ADVERSE WEATHER CONDITION

Supervisors
Prof. Paolo Garza
Dott. Edoardo Arnaudo
Dott. Marco Galatola
Dott. Stefano Bergia

Candidate
Federico Bussolino

Academic year 2024-2025

To my parents and my brother, who supported me during this journey.

Summary

Semantic segmentation is a fundamental perception task for autonomous driving systems, lot of works compare their result in a clear weather in daylight scenario using Cityscapes dataset, however one of the main challenges of autonomous driving nowadays is to build systems that are robust to adverse weather. Another important factor to take into account when it comes to semantic segmentation is that most state-of-the-art architectures are not designed to run in real-time on an embedded system that can also have limited memory. For this reason, this work first evaluates real-time architectures on ACDC, a popular adverse weather dataset, then after selecting one of the best performing model on this dataset, proposes to improve the real-time network using different teacher-student knowledge distillation techniques. The experiments first highlight several improvements in term of model compression using a network with reduced number of parameters without significant performance loss. We then conduct other experiments using a network capable of low-latency inference obtaining a distilled model that perform well on ACDC.

Acknowledgements

I would like to express my gratitude to professor Paolo Garza for his willingness to help and availability during my thesis.

I would like to sincerely thank LINKS Foundation for the support in both term of resources and tutoring support. In particular I want to thank my supervisors: Edoardo Arnaudo, Marco Galatola and Stefano Bergia who followed me during the thesis project.

Heartfelt thanks also to my parents and my family for their economical and emotional support during my whole university journey.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Autonomous driving is a sector that has received particular attention in the industry and research community in recent years due to a series of factors like potential increase in road safety, increased mobility and accessibility, more comfort for driver and rise of new business models related to autonomous vehicles. Those systems can help drivers by reducing reaction times and avoiding possible human errors caused by distraction or fatigue. Furthermore, the use of advanced sensors such as RADAR and LiDAR can significantly enhance scene understanding, potentially outperforming human perception in challenging conditions such as low visibility at night, improving safety and reliability. Autonomous cars can improve accessibility to private transportation for people who aren't able to drive granting them greater independence. Commercial interest in autonomous driving extends beyond the growing enthusiasm of consumers for this innovative technology. It also opens the door to new business models, such as the deployment of self-driving robo-taxis.

Despite the enthusiasm, the deployment of this technology still faces several challenges that prevent the adoption of those systems in different environments and weather conditions. Today, autonomous vehicles that better face all adverse weather conditions usually make use of many sensors, increasing the cost of production of the vehicle. Other missing key point in order to achieve a higher level of autonomy of those vehicles are, in increasing order of complexity: the capability to navigate without a surrounding smart infrastructure, in absence of GPS signal or with lack of road sign and markings. In addition, off-road navigation remains a largely unexplored and underdeveloped area.

To have an idea about the current progress in this field, the standard SAE J3016 1.1 defines the level of automation of self-driving vehicle. Current autonomous vehicles can reach Level 4 autonomy, which means that vehicles can navigate in well-defined places in relatively good lighting and weather conditions.

1.1 Objective of the thesis

The objective of this thesis is to build a robust deep learning model to allow a computer to correctly perceive the surrounding scene in adverse weather conditions, such as night,

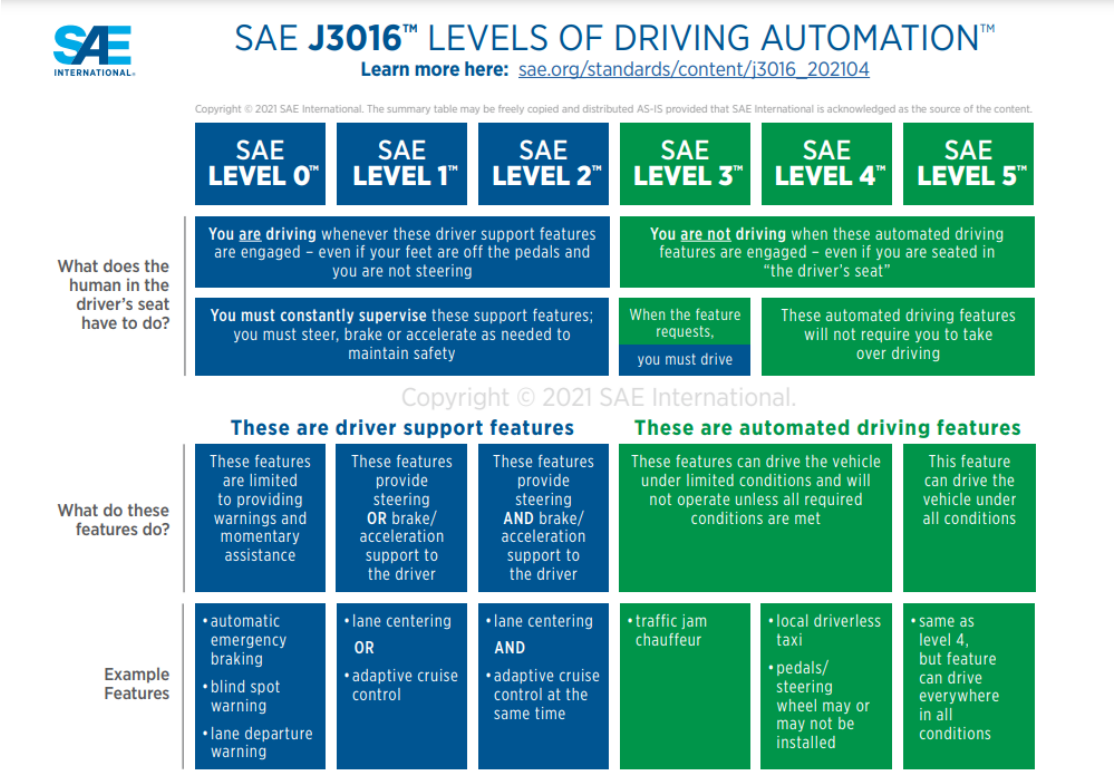

Figure 1.1: SAE J3016 standard, levels of automation

fog, rain, and snow, with a particular focus on low-latency, using only monocular camera images. The focus on low-latency in those systems is closely tied to the fact that deep learning algorithms are deployed on edge devices. These devices, often constrained by the need to minimize deployment costs, operate within the power limits imposed by battery systems, with limited computational power. In this scenario, optimizing for low latency becomes critical in maintaining real-time performance while ensuring that these algorithms can run efficiently on hardware with restricted resources.

Another key factor that can reduce the cost of production of autonomous driving systems is the usage of low-cost sensors, so camera sensors are sometimes preferred over LiDAR. An additional point in favor of using cameras to detect objects is the fact that RADAR are less good at detecting object details and boundaries, so they are less suitable for a task like lane detection. As a downside, camera is more subject to adverse weather and low-light conditions, hence another objective of the thesis is finding computer vision algorithms that perform well in those conditions.

1.2 Thesis contribution

- Assess the effectiveness of the existing model in the context of real-time semantic segmentation under adverse weather (ACDC) dataset.
- Apply known knowledge distillation techniques to improve the performance of existing models on ACDC dataset.
- Explore the effect and feasibility of knowledge distillation technique with non-standard losses (different from Cross-Entropy).

1.3 Thesis organization

Next chapters of the thesis are organized as follows:

- **Chapter 2** cover deep learning background and SOTA architecture for semantic segmentation, presents the datasets used to train, validate, and compare results.
- **Chapter 3** explain methodologies and knowledge distillation frameworks used and gives details about the experimental setting focusing on model adaptations and training procedures.
- **Chapter 4** presents quantitative and qualitative results obtained in comparison with baselines, reporting hyperparameters for each experiment.
- **Chapter 5** summarizes the conclusions we can draw from the experiments and finally gives an overview of possible improvements and future work.

Chapter 2

Background and related works

2.1 AI, Machine learning and Deep learning

The first definition of AI comes from John McCarty (1956, Dartmouth Summer Research Project on Artificial Intelligence) and states: "**Artificial intelligence** is the branch of computer science characterized by the fact that algorithms developed are able to accomplish tasks that require human intelligence."

In 1959 Arthur Samuel coined another fundamental definition for the field of computer science, stating: "**Machine learning** is a subfield of AI that comprehends algorithms that can learn automatically to accomplish task, without the need for being explicitly programmed for the task."

Deep learning is a subfield of machine learning that makes use of artificial neural networks to learn patterns behind more complex data.

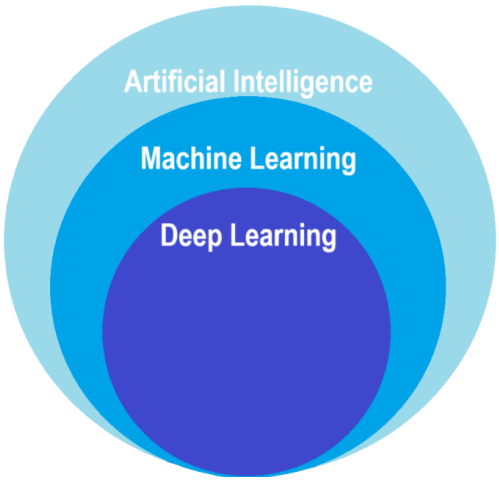

Figure 2.1: AI, Machine Learning, Deep learning

2.1.1 How machines learns

Machine can learn by updating a set of weights (tensors) using the information contained in input data (and eventually their labels). In particular, the network (set of parameters to learn) is updated by performing in sequence: the forward pass, the loss function calculation, and the backpropagation.

Forward-pass indicate the calculation of mathematical operations, defined in a neural network and its parameters, performed on the input to produce predictions.

The loss function is the measure, calculated from the result of forward pass, of how much the algorithm's predictions are distant from the desired output.

Backpropagation [34] is the algorithm that allows us to diminish parameters proportionally to their contribution to the increase of the loss.

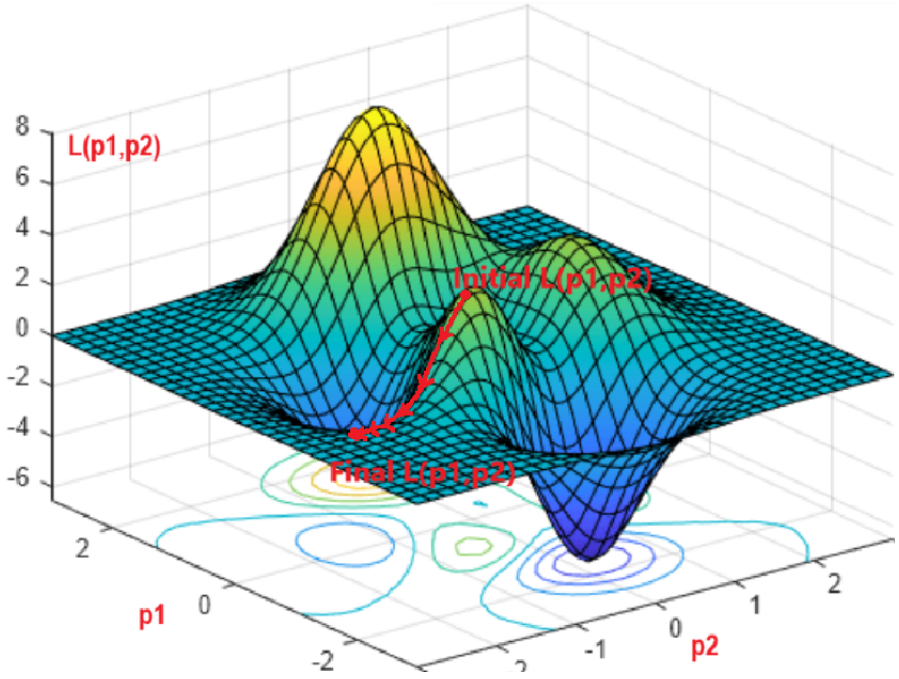

Figure 2.2: Parameter updates on loss landscape

In Figure 2.2 model with randomly initialized parameters performs a forward pass and produces initial loss. At each step $\mathbf{p}$ are updated by a quantity proportional to learning rate α and their contribution to the loss (gradient):

$$\mathbf{p}^{(t+1)} = \mathbf{p}^{(t)} - \alpha \nabla L(\mathbf{p}^{(t)}) \tag{2.1}$$

The model with the newly calculated parameters performs again a forward pass on data, produces again loss and repeat the parameter update. This process is repeated an arbitrary number of times, called steps (6 in Figure 2.2).

2.1.2 Artificial neural networks

One of the most effective ways in which an algorithm can learn to perform tasks such as classification is through the **artificial neural network** made by artificial neurons [26].

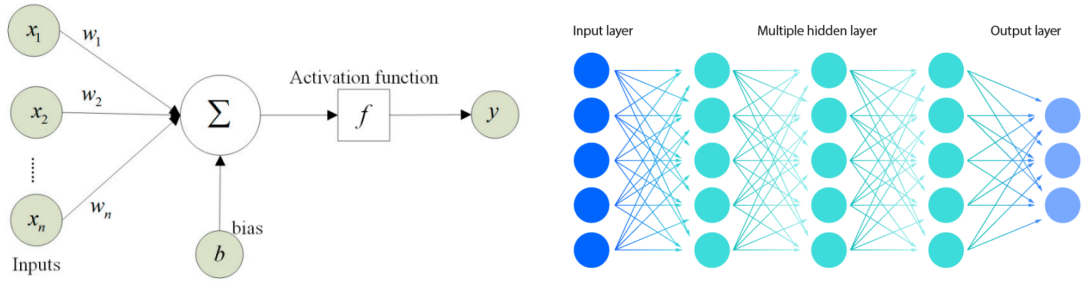

Figure 2.3: Artificial neuron (left, image credits: [38]). Artificial neural network (right, image credits https://www.ibm.com/topics/neural-networks)

In figure 2.3 (right) we can see that a layer of a neural network consists of multiple neurons and the network itself has several layers. Input data are feed-forwarded to the first layer of neuron producing as output the first hidden layer features. Then outputs of first layer are passed to second hidden layer, and so on and so forth until data reach output layer that produces the scores for classification. A layer of a neural network can therefore be expressed as:

$$\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{2.2}$$

Where $\mathbf{x}$ is the input of the layer, $\mathbf{W}$ is the matrix containing the weights of each neuron, with each row representing the weights of the same neuron, $\mathbf{b}$ are the optional biases, $\mathbf{y}$ are the features calculated by all the neurons in a layer and f is the activation function.

Activation functions

ReLU [28] is a simple activation function that serves to introduce non-linearity into the network, enabling it to learn more complex patterns while being easy to compute. It is expressed as $y = max(0, x)$ and is applied element-wise to all values of the vector obtained from **Wx+b**.

Softmax [2] is an activation function used to produce a vector of which elements sum up to one and are greater than zero. This makes softmax useful for multiclass classification, where we need to estimate the probability of each class. Softmax($\mathbf{z}$) is the vector $[e^{z_1}, e^{z_2}, ..., e^{z_m}] / \sum_{i=1}^{m} e^{z_i}$. In this case, it is applied to the vector produced by **Wx+b**.

2.1.3 Convolutional Neural Network

After the success of MLP another type of model is proposed that takes advantage of the prior knowledge we have about the type of information contained in images: the

convolutional neural network (CNN, [19]).

Figure 2.4: 2-d convolution. From left to right we have input, filter, output.

In **2-d convolution** shown in Figure 2.4 the convolutional filters are the weights of the model. The same filter "slides" across the image row by row, moving by a number of pixels equal to the stride S, and at the end of each row, it moves to the next one, skipping S-1 rows. The output is given by the sum of all the products between the filter weights and the value of the input in the corresponding position.

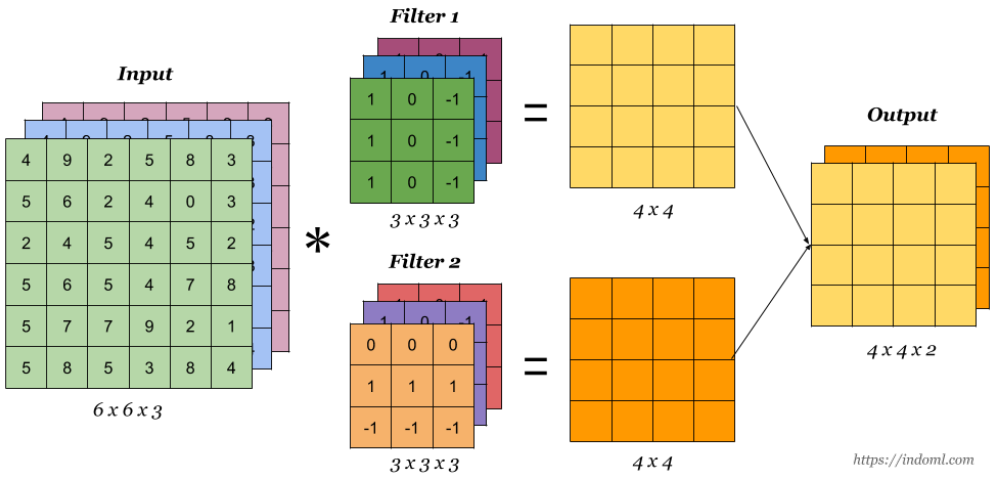

Figure 2.5: 3d-convolution with 3 filters. Image credits: https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

In the **3-d convolution** shown in Figure 2.5 the dimension of the channel (depth) is also taken into account, so the filter has size $C_{in} \times K \times K$. In CNNs usually 3-d convolution is applied using multiple filters, producing a feature map with a number of channels C_{out} equal to number of filters used.

Pooling layer like the one shown in Figure 2.6 is another component that usually appears in CNN and is used to reduce the resolution of the feature maps by a factor equal to the

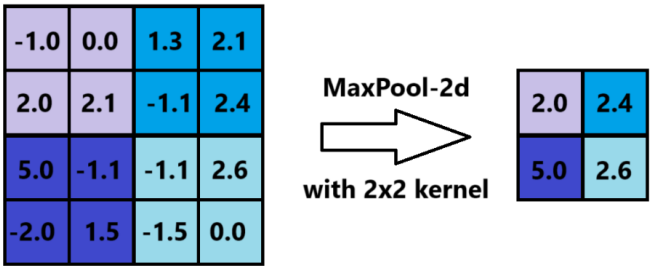

Figure 2.6: Example of max pooling 2-d with kernel of size 2x2 and stride 2.

stride. Usually, max-pooling is used over avg-pooling because it preserves better details like angles and contours.

Batch normalization [18] and **layer normalization** [1] are also fundamental components used in CNN. In larger model layer normalization is often preferred because it allows the model to be trained with very small batch sizes, making it easier to fit on a GPU. The output of normalization $\mathbf{y}$ applied to feature $\mathbf{x}$ is described as follows:

$$\mathbf{y} = \frac{\mathbf{x} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}\boldsymbol{\gamma} + \boldsymbol{\beta} \tag{2.3}$$

In batch normalization $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are mean and standard deviation feature calculated along batch dimension, height and width. Their shape is $1 \times C \times 1 \times 1$ parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are learnable and have the same shape.

In layer normalization, instead $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are mean and standard deviation of features calculated along channels, height, and width. Their shape is $N \times 1 \times 1 \times 1$. The parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are instead learnable and have the shape $1 \times C \times H \times W$.

In CNNs also **activation functions** are applied after convolution operation.

In conclusion, CNNs applied to images allow the following:

- reduction in the number of parameters w.r.t. fully connected network (ANN)
- utilization of the translation equivariance property of convolution
- increase in the depth of the network and thus learning of hierarchical patterns

2.2 Vision Transformers (ViT)

Transformer architecture, introduced for the first time in natural language processing [39], is then ported to computer vision [12] with great success, and now most of the SOTA models for computer vision make use of it.

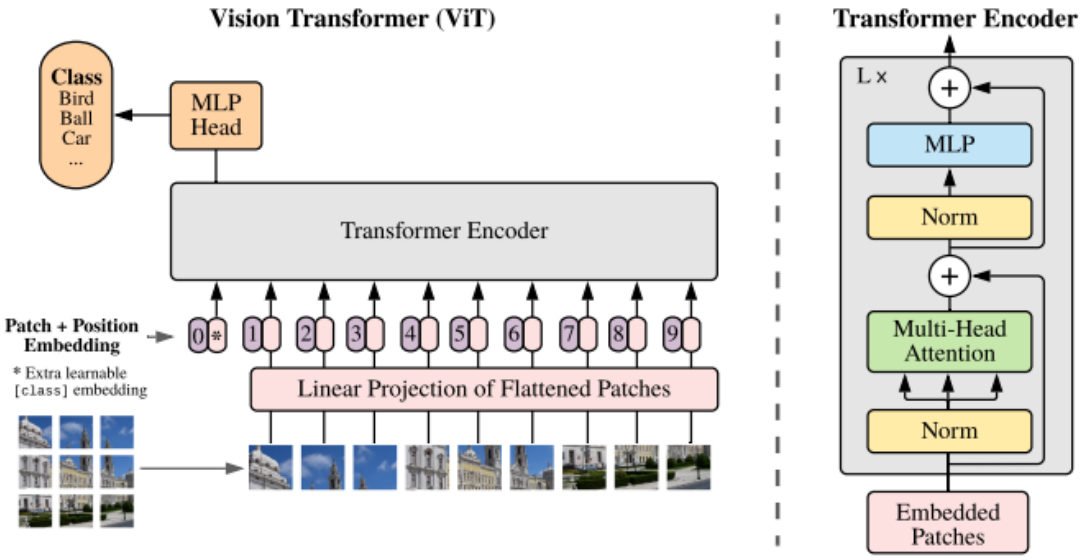

Figure 2.7: Vision Transformer. Image credits: [12]

Positional embeddings (figure 2.7, left)

Vision transformer architecture, take as input the flattened patches of the image, this means that we need to divide the image into patches of size $P \times P$ then feed each of the $N = HW/P^2$ to the fully connected layer that extracts patch embeddings of size D. Before feeding embeddings of patches to the image, a learnable class embedding is concatenated to input, then the positional encodings (number of the patch from 0 to N, where 0 is associated with the class token and other numbers are actual patches of the image) are added to the embedding vector. Given a batch of images of size $B \times C \times H \times W$ the output of positional encoding has shape $B \times (N+1) \times E$ where E is the embedding size (or hidden dimension of the transformer). This process is applied only once to transform the image in a form compatible with the transformer encoder.

Transformer encoder (figure 2.7, right)

In the right part of Figure 2.7 we can see the transformer encoder architecture that is composed of components like MLP (composed of 2 layers with dropout and intermediate activation function), layer normalizations (Norm) and skip connection as introduced in [14]. But the novelty of this block is the usage of multi-head attention (figure 2.8) for image processing. The whole process hence consist into passing the N+1 embeddings which are expanded through the Linear layer from E to an higher dimensional representation, then this higher dimensional representation is split into smaller ones that are the new Q', K', V' and each of those is passed to the respective Scaled-dot-product attention along all spatial locations (i.e. different embeddings). So thanks to Q and K the model learns how to build an "attention map" that tells the model where is important to pay attention

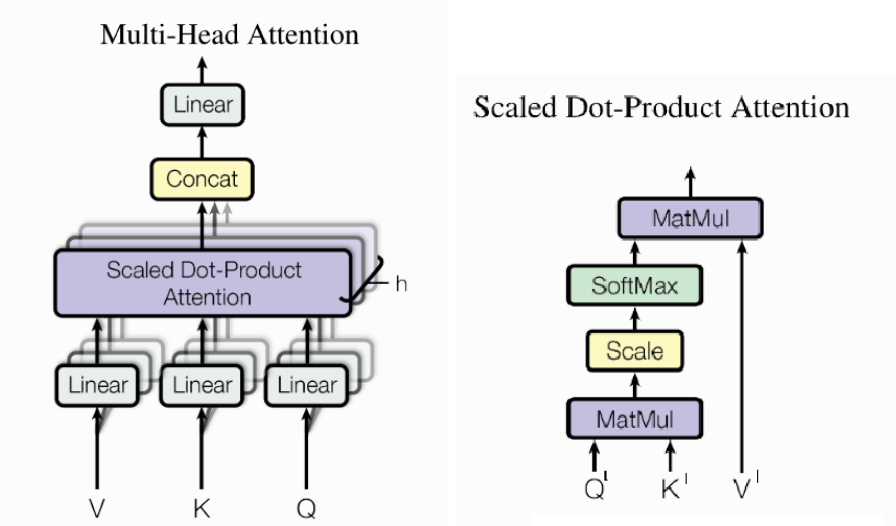

Figure 2.8: Image credits: [39]

during forward pass, then pay attention to V, based on this map, extracting a condensed version of input that retains important information for classification or other task.

2.3 Semantic segmentation

Semantic segmentation is a task that involves dividing an image into semantically meaningful parts and classifying each part into one of the predefined categories (or classes).

In practice it is done by predicting pixel-wise the probability for each class to be the actual class (i.e. object category) to which the pixel belongs. This is done by applying softmax to the output scores of the model (logits).

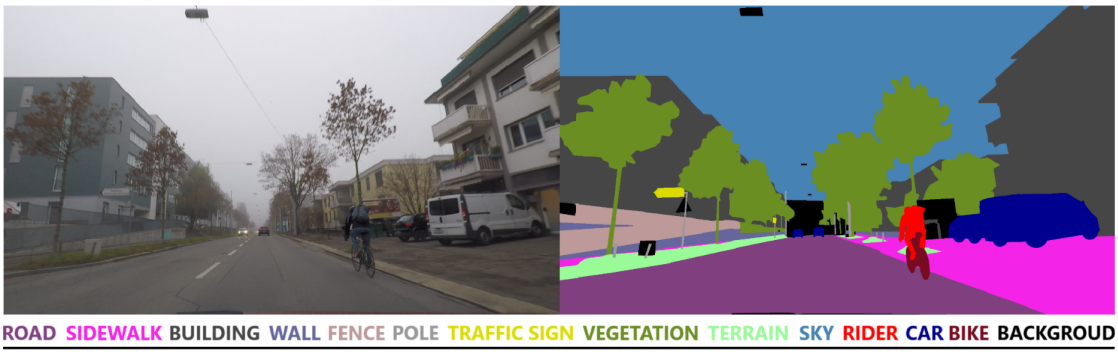

Figure 2.9: Example of result of a semantic segmentation from ACDC dataset

Semantic segmentation architectures

First architecture developed for semantic segmentation is UNet [33] in which an encoder (backbone) shrinks the resolution while increasing the number of channels, then, in the decoder, the features are upsampled with transpose convolution, and, at each up-sampling, the quality of the original feature map is restored due to concatenation with upsampled features, followed by a convolution. In recent years, the DeepLab series of architectures proposed by Google have achieved great success, evolving from DeepLabV1 [3] to DeepLabV3+ [5]. Particularly important is DeepLabV2 [4] in which the last 2 downsampling stages of the backbone are converted into non-downsampling stages and ASPP module, which is a multi-branch module that uses atrous convolution at different dilatation rates to allow sampling from distant feature-map locations, is introduced and applied to last feature map before classification head. In PSPNet [50] they make use of atrous convolution in the last backbone layers, instead of downsampling, then they concatenate to the features of the last layer the global context obtained thanks to the PSP module. In 2019 HRNet is released, it consist of using a backbone that is capable of aggregating features at different resolution by creating at each of 4 stages a new branch at lower ($0.5\times$) resolution while maintaining previous resolution path, on which are applied block similar to down-sampled path but with reduced channel, at each transition between a layer and another, all feature maps at each resolution are properly interpolated and passed as input to path of all resolution of next layer, finally all features obtained from last layer are up-sampled to 1/4 of original image size and concatenated, then processed to output final prediction. Since 2021 computer vision SOTA model's trends shift from pure CNN models to transformer models, one example is SegFormer [42] developed by NVIDIA: in order to make dense predictions, this model's backbone uses a patch size of 4 applying between blocks a patch merging module that consist of a convolution (with $1 <$ stride $<$ kernel size). Each block hence extracts features at different resolution, then feature maps are passed to decode head, in which first lower resolution feature maps are passed through an MLP to create uniform sized embeddings, then upscaled to $H/4 \times W/4$ resolution, concatenated and fused through convolution and passed for final convolution to make predictions. Another important architecture is Mask2Former [8] that allowed us to first train a network capable to perform instance, panoptic and semantic segmentation all with one architecture thanks to the integration of masked transformer decoder to generate masks of pixels and their respective classification labels, instead of directly generating segmentation maps. Another architecture introduced also for semantic segmentation to improve representation of intermediate feature extracted by the backbone is ViT-Adapter [7], we can see that its combination with Mask2Former achieves near-to SOTA performances on Cityscapes.

Real-time semantic segmentation architectures

Beside the high impact of transformer architectures in computer vision, we notice that the real-time best results are still obtained by CNN based architectures, while transformer usually outperforms them when it comes to obtain higher mIoU or accuracy without focusing on low latency, this is related to the attention operation that scales quadratically

with image resolution. However, some transformer architectures like SegFormer-B0 can achieve comparable results with CNN. Architectures developed for real-time semantic segmentation can consist of a backbone that captures image features of different sizes and a decoder that aggregates the features at different semantic levels (resolution) to produce the logits in a way that takes into account all the different features produced by the backbone at different semantic levels (hierarchically). This is the case, for example, of SwiftNet [30] and SFNet [20]. Another popular family of architecture used in similar scenario are multibranch networks: the first example of this kind of network was BiSeNet [46] that achieved an interesting result in terms of latency while maintaining relatively high mIoU with ResNet-18 as the backbone; another example of this structure is PIDNet [44] which is the 2024 SOTA architecture for real-time semantic segmentation in Cityscapes [11].

The first aggregation network architecture to obtain good results in this task was U-Net [33], a segmentation architecture applied to medical images, that uses classical convolution in the downsampling path and transposes convolution in the upsampling path passing to the upsampling path also the feature maps at the same resolutions obtained in the downsampling path.

In order to achieve good results in a real-time scenario like autonomous driving, the idea of early downsamling feature map, reducing computational burden, was essential and, along with other techniques such as dilated convolution, is the basis of ENet [31].

In 2018 BiSeNet [46] showed an increase of around 10.0 mIoU on the Cityscapes dataset while maintaining similar latency of ENet. This was achieved by the removal of transpose convolution and directly using a 8x bilinear interpolation upsampling instead of the decoder.

2.4 Knowledge distillation

Teacher-student knowledge-distillation is a training setting in which a more accurate model's logits or feature maps are used as additional target that student have to mimic to achieve a similar internal representation of the (more accurate) teacher. The framework for knowledge distillation in supervised setting usually comprehends cross-entropy as additional loss as shown in figure 2.10. In classification task has been widely proved that this technique can improve student performance when compared with classical training procedure that uses only cross-entropy as loss.

The main knowledge distillation techniques proposed for semantic segmentation are the following:

- **Logit distillation** [16]: consists in matching the probability distribution of logits of teacher and student network using as loss the Kullback-Leibler divergence.

- **Feature distillation** [49]: consists in matching the teacher and student internal representation (i.e. intermediate feature maps) through a regression loss (L2, L1 or Huber).

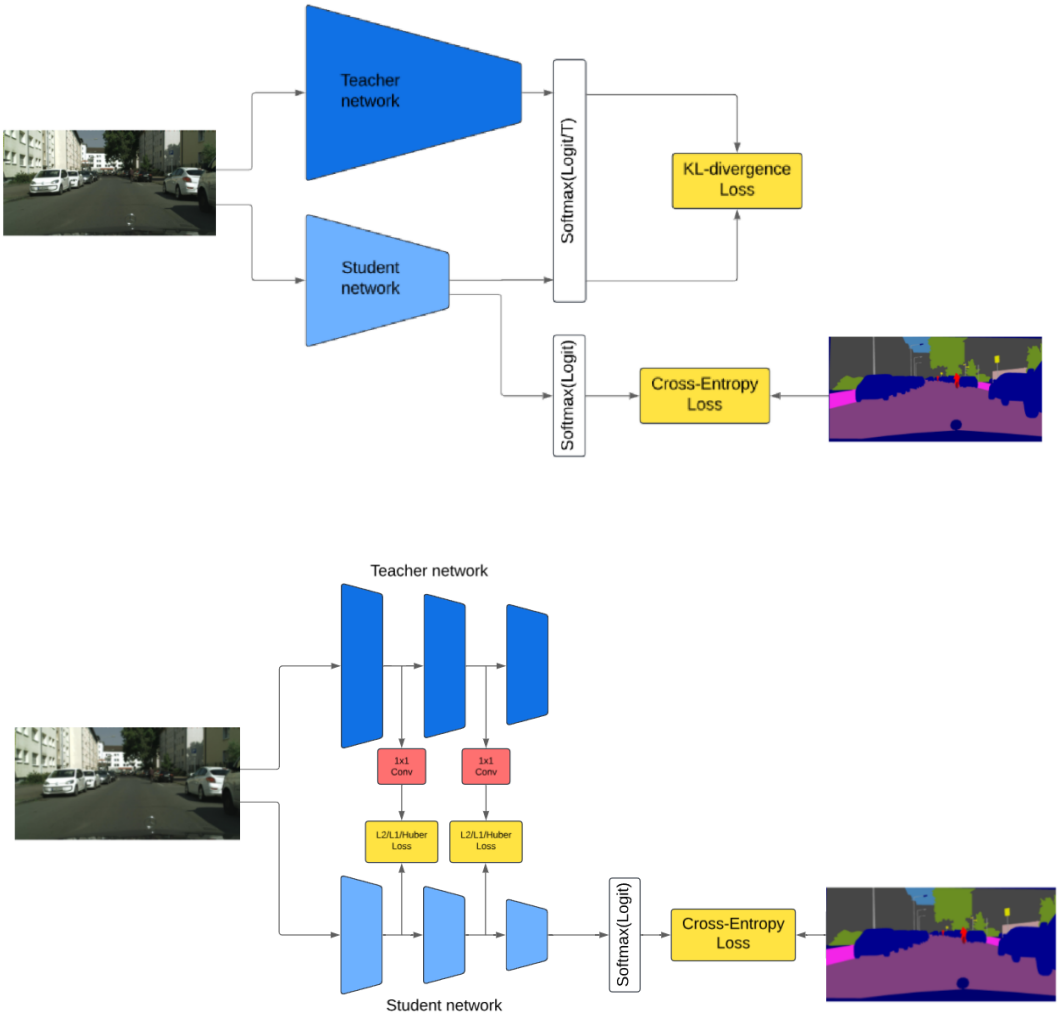

Figure 2.10: First proposed knowledge distillation frameworks: logit distillation [16] (top), feature distillation [32] (bottom)

- **Attention-transfer distillation** [48]: this method consist of distill spatial attention which is obtained by sum or max over all channels of a feature map regulating the sharpness of attention through exponentiation to a hyperparameter. They also propose to distill attention using gradient instead of feature maps.
- **Self-similarity loss**: firstly proposed in "Structured Knowledge Distillation for Semantic Segmentation"[24], the self-similarity matrix of a feature map is a $(H \times W) \times (H \times W)$ containing all similarity scores of pairs of spatial location, where each score in position $((h_1, w_1), (h_2, w_2))$ is obtained as a dot product along the channel

dimension of normalized features at spatial location (h_1, w_1) and (h_2, w_2). Originally, the self-similarities between the feature maps of the student network were regressed to the one produced by the teacher through the distance L2. Also, dissimilarity maps between different samples in same batch can be computed in the same way: in "Cross-Image Relational Knowledge Distillation for Semantic Segmentation"[45] for example dissimilarity maps of teacher and student are regressed using a Kullback-Leibler divergence with low temperature.

- **Local similarity loss** [43]: this method proposes to regress with loss of L2 the consistence map of the teacher and student, where for each spatial location its consistence map is the sum of the L2 distance of the 8-neighbor of the spatial locations with the central spatial location.

- **Intra-class similarity loss**[40]: This kind of feature distillation, has its name because it focuses on distilling intraclass differences between feature map points in different spatial locations belonging to the same class, intraclass similarity is measured using the cosine distance between the vector (composed of all channels) of a spatial location with a prototype (average over $H \times W$) of all spatial locations of the class.

- **Channel-wise distillation**[37]: with this method they exploit the fact that we want to focus on distilling knowledge from channels that contain smaller objects, and hence they compute kl-divergence with temperature over all spatial location for each channel separately instead of computing kl-divergence over channel as in logit knowledge distillation.

- **Normalized-feature distillation**: Recent work "Rethinking Knowledge Distillation with Raw Features for Semantic Segmentation" [23] points out that the loss of L2 between the feature map is dominated by the difference in magnitude between the features of the teacher and the student, but the difference in magnitude is not a factor that significantly impacts the performances, so they propose to normalize the feature in 3 possible ways: layer-wise (LAD), channel-wise (CAD) and spatial-wise (PAD) obtaining interesting results with a simple method.

- **Adversarial distillation**: consist in using a discriminator network to distinguish between student and teacher logits; this network is trained in adversarial settings. In [24] or [40] we see some examples of this kind of loss, also called holistic loss.

Practical limitations of knowledge distillation and proposed solutions

In [27] and [9] they point out that knowledge distillation can lead to poor performances when directly distilling a large model into a very small one, also in our case we noticed that a too big performance gap between the normally trained teacher and the student network correlates negatively with student improvements, particularly when applying logit knowledge distillation.

Therefore, [27] proposes to mitigate this issue by distilling an intermediate network

(teacher assistant) and then use the intermediate network to distill the final target, while another study [9] suggests instead to stop training the teacher early to mitigate the capability gap between the student and the teacher. In other work like [41] to mitigate the difference in the magnitude of features between teacher and student, teacher features are whitened with layer normalization, then student features are regressed to whitened features. In [21] instead they propose to automatically find the optimal temperature by setting a higher temperature for the teacher, since its logits after training tend to have a higher standard deviation. One other paper [6] points out that in case of overconfidence in teacher prediction, it may be beneficial to use multiple projectors on the student side to give the student more flexibility to mimic teacher logits/prediction, hence applying also the projector to logits, not only to intermediate features, and also to use multiple projector for feature distillation.

Loss used in experiments

Among the losses described in this chapter the losses used in experiments and hence explained in methodology are:

1. logit distillation (kl-divergence)
2. feature distillation (with L2-loss)
3. channel-wise distillation (CWD)
4. normalized feature distillation (LAD)

2.5 Datasets

2.5.1 Cityscapes

Cityscapes [11] is one of the first dataset for autonomous driving, widely recognized as a benchmark, containing road-scene images at resolution of 1024×2048px, taken: from the driver's point of view, in 50 different cities, during spring, summer, or fall, in normal meteorological conditions, in daylight condition. The images are extracted from selected frames of videos. The dataset provides labels for semantic, instance, and panoptic segmentation. The semantic segmentation labels consists of 19 distinct classes and a background class, in figure 3.1 we can see pixel's distribution among classes. The semantic segmentation dataset is split as indicated in table 2.1

Train	Val	Test
2975	500	1500

Table 2.1: Dataset Split Distribution

2.5.2 BDD100K

Berkeley Deep Dive 100K [47] is a dataset consisting of 100K video of 40s collected through crowd-sourcing for a total of 1.2M images at resolution 720 × 1280px, of which the 10s frame are annotated for object detection, lane detection and drivable area segmentation for a total of 100K images annotated for those task. For semantic segmentation, instance segmentation, and panoptic segmentation, only a subset of 10K images is selected by random sampling from the 100K already selected for the previously mentioned tasks. Also in this dataset semantic segmentation classes are the same 19 classes of Cityscapes and a background class. The frames were taken in multiple cities, under diverse weather and lightning conditions. This dataset is hence indicated to measure the effectiveness of multitask algorithm and due to its variety is also useful to address the robustness to adverse conditions even though ACDC or other on-purpose datasets better represents adverse weather conditions. The semantic segmentation portion of dataset (BDD10K) is split as indicated in table 2.2.

Train	Val	Test
7000	1000	2000

Table 2.2: Dataset Split Distribution

2.5.3 ACDC

Adverse Conditions Dataset with Correspondences [35] is a data set containing road scene images at a resolution of 1080 × 1920px that respect the following criterion: taken from the driver's point of view, with the condition of fog, night, rain, snow or normal (ref split), in urban, rural, or highways regions. The images are extracted from selected frames of videos. The dataset provides labels for semantic, instance, and panoptic segmentation. The semantic segmentation task involves the 19 classes of Cityscapes and a background class. The background class is not considered during the calculation of the metrics. This dataset, in addition to having the same classes as the popular Cityscapes benchmark [11], is more suitable to assess domain generalization capabilities and robustness of autonomous driving algorithms due to the variety of weather conditions that it covers. Some examples of images are shown later in figure 2.11.

Condition	Train
Fog	400
Night	400
Rain	400
Snow	400
Normal_w_label	800
Normal_no_label	800

Condition	Val
Fog	100
Night	106
Rain	100
Snow	100
Normal_w_label	203
Normal_no_label	203

Condition	Test
Fog_no_label	500
Night_no_label	500
Rain_no_label	500
Snow_no_label	500
Normal_no_label	2000

Table 2.3: ACDC dataset splits

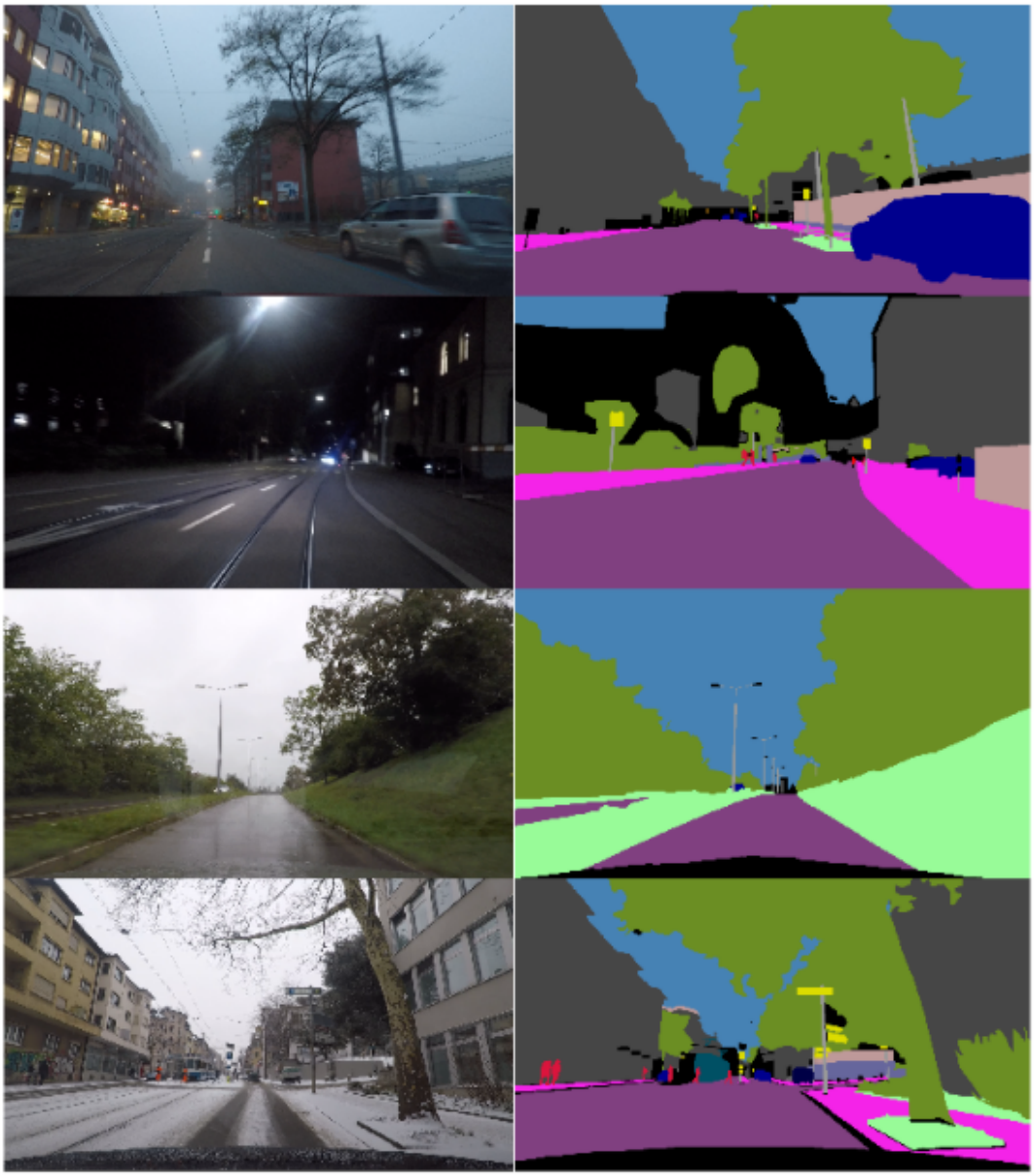

Figure 2.11: Examples of images from ACDC, taken in different conditions, we can see that night images can be particularly challenging to classify and also some minority classes can be difficult to distinguish.

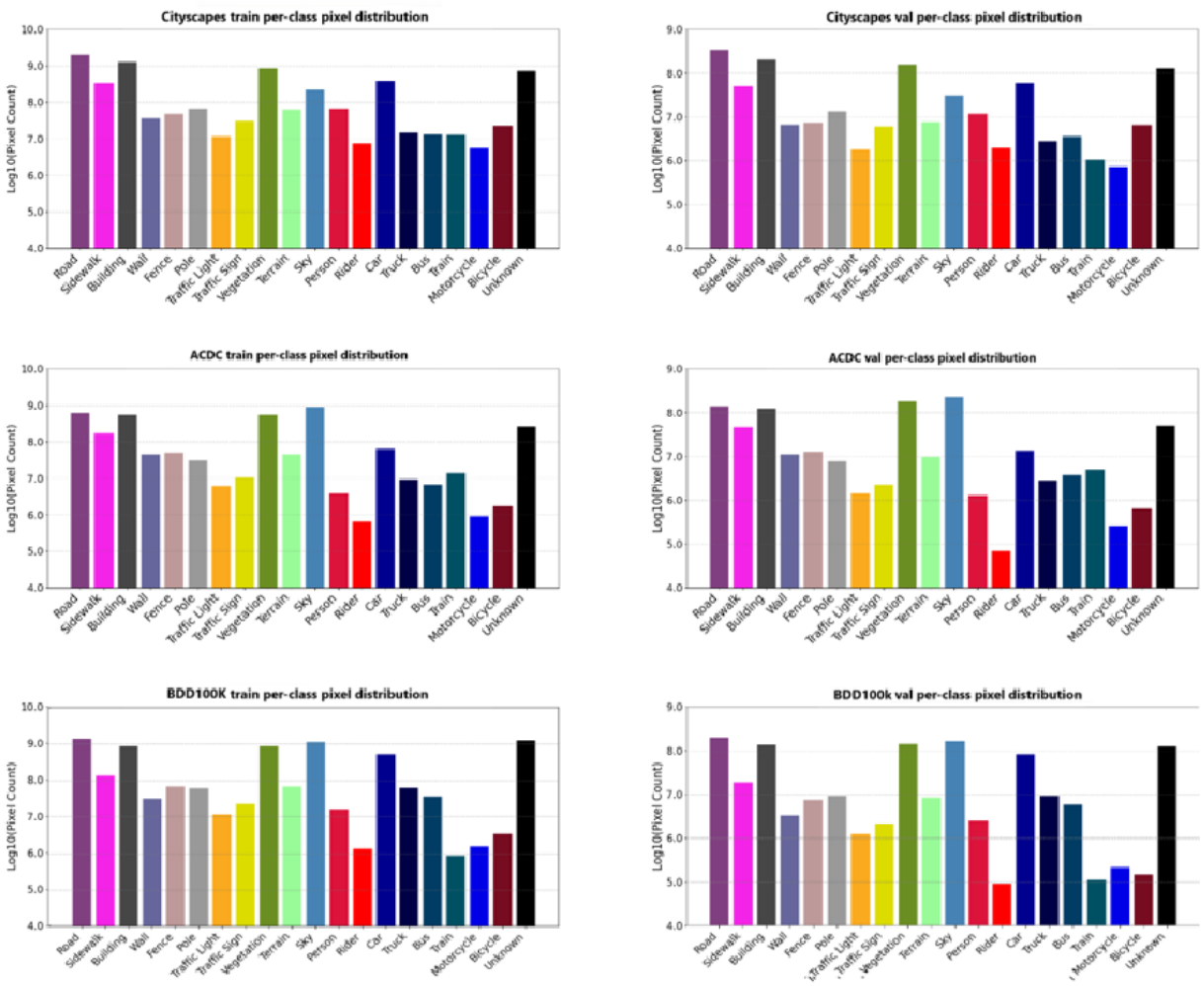

Figure 2.12: Class distribution of pixel over Cityscapes (top), ACDC (center), BDD100K (bottom). Divided into train split (left) and val split (right).

Chapter 3

Methodology

Our objective is to compress a model obtaining similar performances while diminishing latency and number of parameters, in order to deploy it on edge devices with limited resources. Among methods to compress model we discarded pruning since the large network we have has an increased number of layer that are the main factor that slow down the model, moreover structured pruning methods, which could be the only method that can improve latency, usually require different iterations of pruning and fine-tuning leading to an important demand in terms of computational resources. Moreover, since pruning can create a more irregular network, that is different from existing network, there is the risk that the latency improvements measured are highly hardware dependent, and there is also a lack of latency benchmark for the new network obtained. For this reason, we decide to compress the model through knowledge distillation: as we said in Chapter 2 teacher-student knowledge distillation consists of using a well-trained teacher's output (or intermediate features) and regress student outputs (or intermediate features) to them.

3.1 Problem statement

Formally, in a context of supervised knowledge distillation, for a student model f_s and teacher model f_t , using a Cross-Entropy loss we impose that given x, the input, y_{gt} the labels, with a teacher network we generate y_t (the soft labels) as y = $f_t(x)$ and student network generate y (prediction) as y = f_s and we use as loss for student model:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_{\text{CE}}(y_s, y_{gt}) + \lambda\mathcal{L}_{\text{KD}}(y_s, y_t) \tag{3.1}$$

In our version, we will use the border focal loss instead of $\mathcal{L}_{CE}$. And add other losses that regress intermediate features of the student network (f_s) to intermediate features of the teacher network (f_t), so the formulation is as follows:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_{\text{sem}}(y_s, y_{gt}) + \lambda\mathcal{L}_{\text{KD}}(y_s, y_t) + \alpha\mathcal{L}_{\text{features}}(f_s, f_t) \tag{3.2}$$

Where $\mathcal{L}_{\text{sem}}$ is the semantic segmentation loss.

To do that in the following sections we describe different losses between student and teacher features or logits are described, motivating their usage, and explaining how they

are applied. Later in this chapter, the main architectures involved in our experiments are explained.

3.2 Losses

The main loss used along with SwiftNet architectures is Focal-Border loss expressed as:

$$\mathcal{L}_{\text{border}}(P, GT, \alpha) = \frac{1}{\sum_{b,h,w} \alpha_{b,h,w}} \sum_{b=1}^{B} \sum_{h=1}^{H} \sum_{w=1}^{W} -\alpha_{b,h,w} \sum_{c=1}^{C} (1 - P_{b,h,w,c})^{\gamma} \log(P_{b,h,w,c}) \mathbb{I}(c = GT_{b,h,w}) \tag{3.3}$$

Where regional weight α is zero in correspondence of ignore region.

This loss allows the network to focus more on the border, implicitly giving more importance to smaller and more articulated classes that usually are more difficult to segment precisely (usually have a low IoU). It is calculated using distances from the border as coefficients.

For the focal-border loss distances and respective weights are obtained by preprocessing labels as in [30] as follows: for each class obtain a mask of the pixel belonging to that class, then apply Distance Transform algorithm that updates the L2 distance map of each pixel of the class by convolving a 2 fixed filter in 2 convolutional pass. Once the distance from the nearest border to the other class is obtained, based on distance, a weight is assigned to each logit location and the contribution to the loss of the logit in the spatial location is obtained by multiplying the loss of the location times the weight of the region α.

Figure 3.1: Image(left), GT(center), Border distance weight(right)

Since we want to use knowledge distillation to help our model generalize better and be more robust, one of the losses used in our experiment is the KL-divergence loss for knowledge distillation [16], which, in the context of semantic segmentation, is expressed as:

$$\mathcal{L}_{kldiv} = \frac{1}{\sum_{b,h,w} M_{b,h,w}} \sum_{b=1}^{B} \sum_{h=1}^{H} \sum_{w=1}^{W} M_{b,h,w} \sum_{c=1}^{C} P^{T}_{b,h,w,c} \log \left(\frac{P^{T}_{b,h,w,c}}{P^{S}_{b,h,w,c}} \right) \mathcal{T}^2 \tag{3.4}$$

Where P^T are teacher prediction (output of softmax of the teacher) and P^S are student prediction (output of softmax of the student) obtained by $\mathcal{T}$-scaled softmax. M is the ignore region mask (0 in correspondence of pixel to ignore, 1 if pixel contains a valid class).

This loss aims to preserve the distribution of class probabilities over pixels; in this sense, it is adapted from the version used for classification where the distribution of all classes (instead of just hard labels) can give more hints about how the teacher network makes predictions and hence about its parameters. For example, differently from Cross-Entropy loss we can regress logits such that they imitate better the real probability distribution of an object; for example, using hard labels such pedestrian with Cross-Entropy prevents the model to learn that there are similarities between pedestrians and riders, instead with KL-Divergence we can learn that probability of rider is higher than car when we see a person and this conveys to the model higher quality information about our classes and their relationships.

Since we wanted to incorporate region importance that are not present in classical KL-divergence, we experimented with KL-divergence border loss, which is an attempt to incorporate the class-wise relationships that Hinton proposes as a target to guide student without losing the capability to give implicit importance to minority classes, fundamental for an imbalanced task like semantic segmentation. This is achieved by also giving importance based on the context around each pixel, and hence keeping into account spatial information that is also essential for semantic segmentation.

$$\mathcal{L}_{kldiv}(P^S, P^T, \alpha) = \frac{1}{\sum_{b,h,w} \alpha_{b,h,w}} \sum_{b=1}^{B} \sum_{h=1}^{H} \sum_{w=1}^{W} \alpha_{b,h,w} \sum_{c=1}^{C} P^{T}_{b,h,w,c} \log \left(\frac{P^{T}_{b,h,w,c}}{P^{S}_{b,h,w,c}} \right) \mathcal{T}^2 \tag{3.5}$$

Where P^T are teacher prediction (output of softmax of the teacher) and P^S are student prediction (output of softmax of the student) obtained by temperature-scaled softmax. Notice that regional weight α is zero in correspondence of ignore region.

Since KL-Divergence loss alone usually does not achieve particularly better results when compared to loss that distills intermediate features, we use L2 loss to distill intermediate features, it was first proposed in [32] and is still considered a valid way to perform feature-based knowledge distillation, its expression is the following:

$$\mathcal{L}_{L2-feat}(F^S, F^T) = \frac{1}{B \cdot C \cdot H \cdot W} \sum_{b=1}^{B} \sum_{c=1}^{C} \sum_{h=1}^{H} \sum_{w=1}^{W} \left(F^{S}_{b,c,h,w} - F^{T}_{b,c,h,w} \right)^2 \tag{3.6}$$

The objective of this loss, in the context of knowledge distillation, is to match the internal representation of teacher and student by making the intermediate feature map generated by the two networks as similar as possible. Usually, this method is used jointly with logit distillation, but in some of our experiments, we will find counterproductive to use KL-Divergence.

Due to the fact that the issue of poor segmentation capability of small objects persists when distilling with L2 loss, we employ CWD which, proposed in [37], consist in applying softmax with temperature to feature-map as follows:

$$\phi(F_c) = SpatialSoftmax(F_{b,c}, \mathcal{T}) = \frac{\exp\left(\frac{F_{b,c,i}}{\mathcal{T}}\right)}{\sum_{i=1}^{W \cdot H} \exp\left(\frac{F_{b,c,i}}{\mathcal{T}}\right)}, \tag{3.7}$$

Then the loss is computed as:

$$\mathcal{L}_{CWD}(F^T, F^S) = \frac{\mathcal{T}^2}{B \cdot C} \sum_{b=1}^{B} \sum_{c=1}^{C} \sum_{i=1}^{W \cdot H} \phi(F^T_{b,c,i}) \cdot \log\left(\frac{\phi(F^T_{b,c,i})}{\phi(F^S_{b,c,i})}\right) \tag{3.8}$$

This loss is expected to be particularly useful in small object segmentation since its objective is to imitate activation distribution channel-wise, giving equal importance to each channel, so channel that contains information about big object will have smooth distribution and differences between teacher and student in those channels will affect less the loss w.r.t. differences in channel which contains information about smaller object. In case of a smaller object, the activation distribution in fact will be more sharp and smaller changes will lead to an important difference in channel-wise distillation loss.

In [22] they show that distilling the feature by matching their magnitude is counter-productive, so they propose to normalize the feature by channel (CAD) or by entire layer (LAD). We believe in the hypothesis since already trained teacher model intermediate feature have larger variance than the student features, but their difference in variance does not reflect in segmentation capability so it can be useful to distill a normalized version of features. In our case LAD is applied to feature map and is expressed as follows:

$$\mathcal{L}_{LAD}(F^T, F^S) = \frac{1}{B} \sum_{b=1}^{B} \sum_{i=1}^{C \cdot H \cdot W} \left(\frac{F^S_{b,i}}{\|F^S_b\|_2} - \frac{F^T_{b,i}}{\|F^T_b\|_2}\right) \tag{3.9}$$

Here $\|F_b\|_2$ is the norm-2 of the flattened vector along C, H, W. We use LAD over other methods proposed in [22] since from the paper it emerges that it usually tends to perform better, particularly when there is a big performance gap between teacher and student networks.

3.3 Architectures

3.3.1 SwiftNet-FPN

SwiftNet allow to substitute the backbone used without changing segmentation head, for this reason we experimented with different backbones described in following subsections. In SwiftNet the same backbone composed by residual block layers is applied to image at different resolution, the intermediate feature maps produced are then downsampled (or upsampled) to a fixed number of channels (2560 in figure), then aggregated feature maps

(with fixed number of channels) that have lower resolutions are upsampled (UP block) by a factor of 2 and added to higher resolution feature maps, then the sum are processed through a 3×3 convolution.

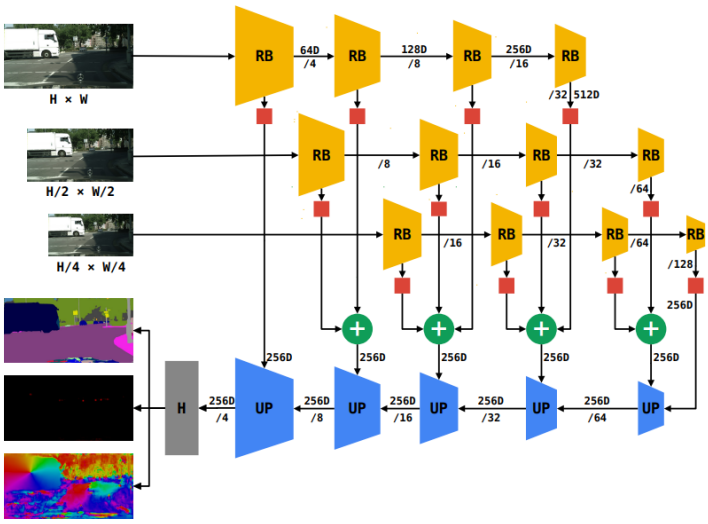

Figure 3.2: Swiftnet FPN aggregator from [30]

3.3.2 SwiftNet backbones

ResNet

ResNet [14] allows to develop a deeper network with good performance thanks to the introduction of **residual connections**. This design allows the network to "propagate" easily the identical representation of a previous feature map. ResNet-18 is still a good choice in some resource-constrained applications that require a low latency and small model.

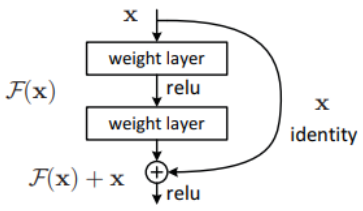

Figure 3.3: Residual connection from [14].

MobileNet series

MobileNet [17] make use of depth-wise separable convolution, introduced in [10], to improve the speed and size of the network. Using a depth-wise separable convolution layer that consist of, instead of C_{out} convolutional filters of size $C_{in} \times K \times K$, applying first C_{in} convolutional filters of size $1 \times K \times K$ followed by C_{out} convolutional filters of size $C_{in} \times 1 \times 1$.

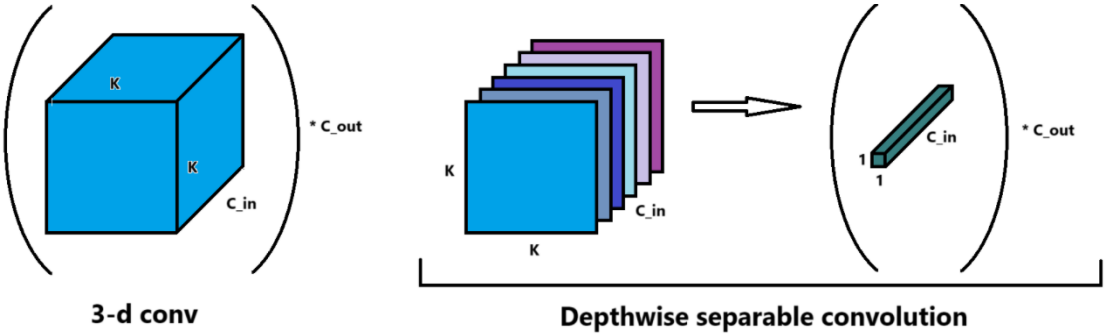

Figure 3.4: Normal 3d-convolution vs. depth-wise separable convolution.

MobileNetV2 [36] is built starting from MobileNetV1 introducing **linear bottleneck** before ReLU activations and residual connections are applied between the output of bottlenecks and goes under the name of **inverted residuals**. This change is introduced under the hypothesis that the activation functions preserve better information if they operate on features of lower-dimensional spaces.

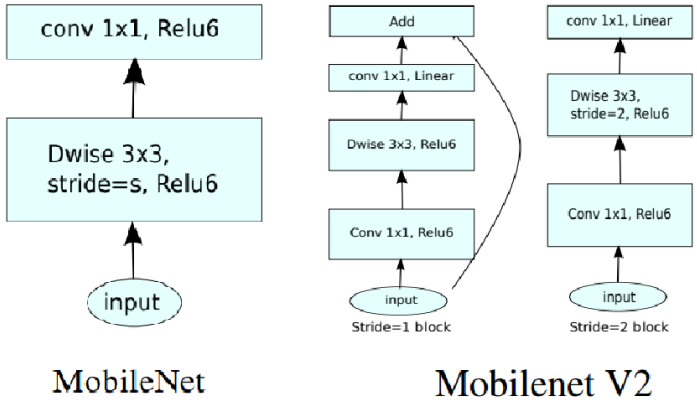

Figure 3.5: MobileNet block first on the left, MobileNetV2 blocks on center and on the right. Credits [36]

MobileNetV3 and MobileNetV4 makes use of similar architectural design w.r.t. their predecessor MobileNetV2 but their overall structure is designed using NAS (Neural Architecture Search), hence are not designed manually.

ConvNeXt

ConvNeXt [25] is a CNN based architecture that in 2022 was still able to outperform SOTA transformer architectures. In this family of models, depth-wise separable convolution, linear bottleneck, and inverted residuals are used. Like in transformer architectures, ReLU is replaced with **GELU** [15] and batch normalization is replaced by **layer normalization**, also, to be more similar to transformer architectures, the classical 3×3 kernel is replaced by a $\mathbf{7 \times 7}$ **kernel** and less normalization layers are used. ConvNext aims to outperform transformer architectures that have a higher number of parameters and are usually slower, so this architecture is not designed for real-time performance and even the smallest ConvNeXt-pico cannot reach latency comparable to the small version of ResNet or MobileNet (on RTX2080Ti).

3.4 Logit and pre-logit distillation

In this set of experiment, loss are applied either on logits (eventually, in case of kl-divergence loss with softmax and temperature) or on pre-logits meaning that the losses are applied before the convolution layer that adapts the number of output channels to number of classes in order to make prediction. In particular, SwiftNet experiments focus on logit and pre-logit ditillation using kl-divergence, or boundary-weighted kl-divergence and/or L2-loss for pre-logit distillation. We also carried out an experiment with logit knowledge distillation on PIDNet halving the contribution of Ohem loss on logit, adding with weight 0.5 the kl-divergence loss with a temperature of 1.0, and confront the results obtained with the baseline PIDNet training procedure. All parameters except loss weights used to train PIDNet are the ones reported in [44].

3.5 Intermediate feature distillation

In corresponding experiment section baseline with Border-Focal loss is reported for a comparison. For SwiftNet with ResNet18 we report experiments using L2 loss loss applied on all 6 feature maps (LAD and CWD are applied in the same way). All experiments make use of. Border-Focal loss (indicated as border), more promising intermediate-feature distillation techniques are also tested with kl-divergence loss (kl).

The framework in figure 3.6 represents the level at which feature are extracted both from teacher and student to regress: student feature in input to L2_loss or KL_Div_loss are the loss that allow the student to learn from teacher, teacher features in input to L2_loss or KL_Div_loss, hence teacher does not learn from that. As we can see in figure 3.6, student and teacher architectures are same, except for the backbone. In the same way, input images are the same for both teacher and student as well as GT.

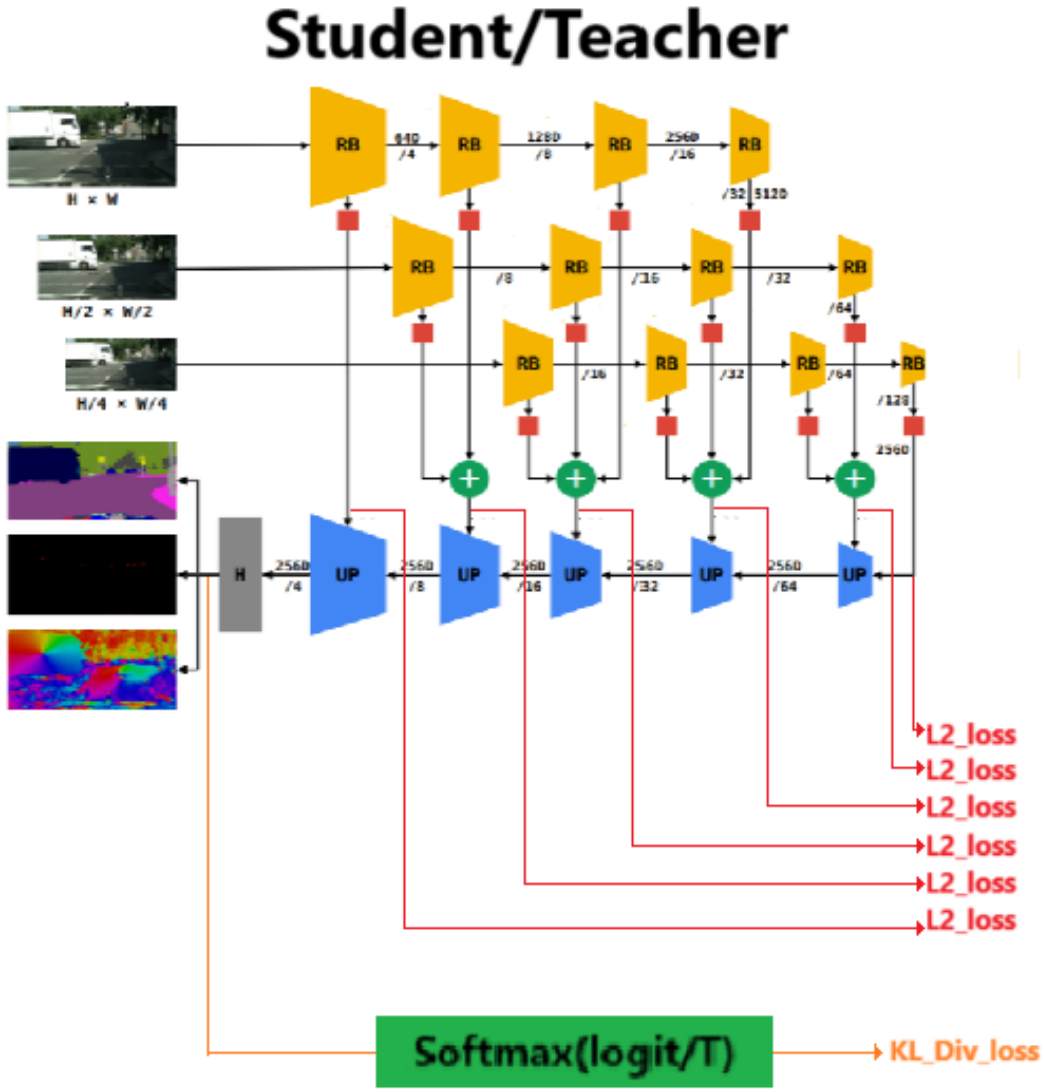

Figure 3.6: Distillation framework (second set of experiments)

Chapter 4

Experiments

4.1 Metrics

IoU and mIoU

The most common metric in multiclass semantic segmentation is **mIoU**, since it takes into account class imbalance, it is easy to interpret and compute. It is the mean IoU score across classes. Each class-IoU score calculated considering one class (c) at a time counting pixels as follows:

- TP: the pixels where the predicted class and the correct class are equal to c.
- FP: the pixels where the predicted class is c, but the correct class is not c.
- FN: the pixels where the predicted class is not c, but the correct class is c.

$$IoU_c = \frac{TP_c}{TP_c + FP_c + FN_c} \tag{4.1}$$

In reference to figure 4.1 IoU for class c is $\frac{Area(A \cap B)}{Area(A \cup B)}$ where A (lightblue blob) is the mask where predicted class is c and B (yellow blob) is the mask where correct class is c.

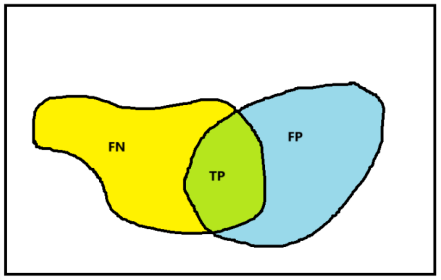

Figure 4.1: Visualization of IoU

ECE

Particularly for autonomous driving and in general in the scenario in which a system can make multiple decisions based on a prediction, it is important to also take into account model confidence. For example if a model predicts a car in front of us but with low probability (even if it is class with highest probability) it is possible to brake but maybe if probability is relatively small we can wait next frame or use another prediction system (maybe based on RADAR) to avoid a brake when actually there is nothing, avoiding also discomfort for driver. From examples like this, we can say that having a model that makes prediction that has a confidence that reflects its accuracy is a positive characteristic in order to account on the model to take complex decisions. For this purpose, ECE is introduced in [29]. It consists of:

- Divide probability range (0,1) into bins;
- Assign each prediction into a bin based on predicted probability of most probable class (noted with P);
- For each bin compute overall accuracy of the predictions in the bin and mean P (confidence) of predictions in the bin;
- ECE is then expressed as:

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)| \tag{4.2}$$

Where $|B_m|$ is the number of predictions in bin m and n is the total number of predictions. In our case, we use 15 bins and compute ECE over batches of 16 then average due to memory constraint reason.

AUPRC

AUPRC is an important metric to assess the ability of the model to detect OOD data, particularly. When it comes to imbalanced class dataset or scenario where is important to classify correctly rare classes (ex. pedestrian, rider) AUPRC is more suitable than AUROC since it tends to penalize more the FN of rare classes. This metric is designed for binary classification tasks but can be applied to multi-class tasks comparing each class individually with the rest of the others.

For one class versus other is calculated as follows:

- Set different thresholds over the probability range (0,1);
- Calculate, for each threshold, Recall and Precision of the selected class as follows:
  - Recall: TP/(TP+FN)
  - Precision: TP/(TP+FP)

- Sort tuples $(Recall, Precision)_t$ based on Recall;
- Calculate the area under curve created by the tuples $(Recall, Precision)_t$ to obtain the AUPRC.

In our experiment, we will show the AUPRC by class obtained using 20 thresholds.

4.2 Experimental setup

Software configuration used for knowledge distillation experiments. Quantization is ap-

Package	**Version**
einops	0.6.1
numpy	1.24.4
open-clip	2.28.0
opencv-python	4.8.0.74
pillow	10.2.0
psutil	5.9.5
pytorch-lightning	2.0.6
scipy	1.10.1
tensorboardX	2.6.2.2
timm	1.0.11
torch	2.0.0+cu117
torchmetrics	1.5.1
torchvision	0.15.1+cu117
tqdm	4.66.5

Table 4.1: Required package version

plied by training the network with AMP to deliver the final model that can achieve good performance in FP16 precision, showing that this technique usually improves FPS by a factor of approximately 2 on RTX2080Ti.

4.3 Baseline results

Both for the lower and upper bound models, FPS are measured on RTX2080Ti, with a batch size of 1 at full ACDC resolution of 1080×1920px. Note that distilled network can achieve more FPS on same hardware since distillation process uses images at 0.8 their original resolution simply by downscaling and re-upscaling with bilinear interpolation.

Lower bound mIoU (real-time, low memory usage)

The baseline for the low-latency small model comprehends that performed well, in terms of mIoU, on the Cityscapes test split in a real-time scenario using the same hyperparameters used on Cityscapes except for training crop that is 1080×1080 then resized to 0.8 its original size via interpolation for faster training; the results are reported in table 4.2.

Model	**ACDC val mIoU**					**Params(M)**	**FPS**
	full	fog	night	rain	snow		
PIDNet-L	71.1	77.5	52.2	68.8	74.7	36.9	47
SN-FPN-MNv4	70.1	77.0	51.4	68.2	75.3	13.2	14
PIDNet-M	70.0	76.6	51.7	71.1	71.8	28.5	59
PIDNet-S	68.8	74.6	52.2	65.9	72.0	7.6	114
SN-FPN-RN18	67.7	74.8	50.1	65.2	68.2	16.7	33
SF-RN18	66.9	74.1	48.8	65.6	70.4	12.9	34
SF-lite-RN18	64.8	71.6	47.8	63.0	68.5	12.3	34
SegFormer-B0*	64.4	71.3	47.0	61.9	68.8	3.7	25*

Table 4.2: ACDC realtime model baseline. *Model trained in PaddlSeg, latency measured in torch(timm) for fair comparison.

Upper bound mIoU (not real-time, high memory usage)

A baseline comprehending larger and slower network that perform well on ACDC is reported in table 4.3, some of the models are taken from ACDC supervised semantic segmentation public benchmark (test split) and 2024 update of ACDC paper [35], table 4.4, teacher model (SwiftNet-ConvNeXt-L) and HRNetV2-W48 latencies are measured on RTX2080Ti, validation split metrics of HRNetV2-W48 are reported from thesis related to paper [13].

Model	**ACDC val mIoU**					**Params(M)**	**FPS**
	full	fog	night	rain	snow		
SWiftNet-ConvNext-L	85.0	87.7	74.0	82.1	84.0	206	4.8
HRNetv2-W48*	73.5*	74.7*	65.3*	77.7*	76.3*	66	9.5

Table 4.3: Baseliness on ACDC val. *From [13]

4.4 Distillation results

All experiments on SwiftNet have the following hyperparameters configuration:

Method	ACDC test mIoU
ViT-Adapter	78.4
Mask2Former	77.3
HRNetv2-W48	75.0

Table 4.4: Baselines on ACDC test. From [35]

Parameter	Value
Batch Size	8
Gradient Accumulation Steps	2
Max Epochs	250
Resize Augmentation	(0.5, 2.0)
Crop Size	(1080, 1080)
Crop Rescaling Factor	0.8
Normalization	ACDC std and mean
Precision	FP16
Optimizer	Adam
Betas (Adam)	(0.9, 0.99)
Learning Rate	4×10^{-4}
Weight Decay	1×10^{-4}
Scheduler	Cosine Annealing
KL-div-temerature	1.0
KL-div-weight	0.5

Table 4.5: Training Configuration Parameters. KL-div loss in all set of experiment, if present use a temperature of 1.0 balancing with main loss with equal contribution (both 0.5).

4.4.1 Logit and pre-logit distillation

First set of experiments consists of a series of short-run experiments in order to evaluate the effectiveness of kl-border weighted kd, vanilla kl-based knowledge distillation and kl-border combined with feature distillation of the penultimate layer. Since we want to improve on a model that already implements an on-purpose loss for semantic segmentation, the main loss is the focal border-weighted loss that showed best performance on pyramidal SwiftNet [30].

Figure 4.2 shows slightly worse performances for classical knowledge distillation with kl-divergence, and near to no improvement when using a regional weighted kl-divergence.

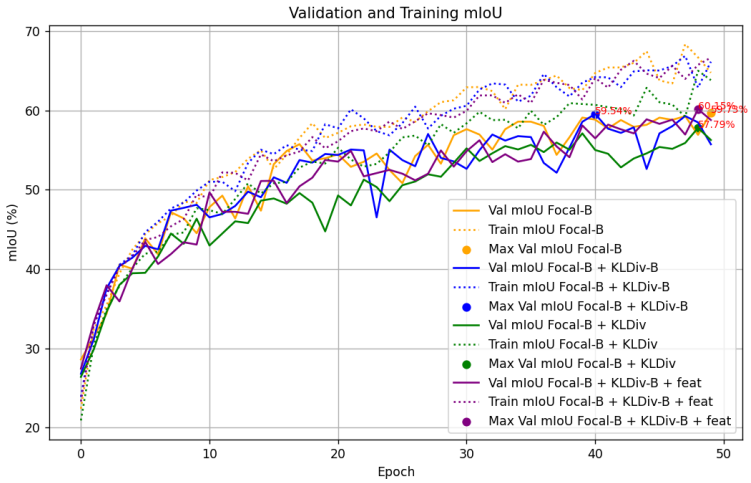

Figure 4.2: Visualization of mIoU over 50 epochs. All the experiments use a SwiftNet-FPN with ResNet18. Early stopping at epoch 50 is applied

Moreover, feature distillation when applied on the penultimate layer (before channel adaptation to the number of classes) also shows no improvement. Extending training to 250 epochs yielded worse results than classical training with only Focal-B loss also for all experiments using border logit distillation. However, we can notice that with a larger network such as MobileNetv4 as a backbone, this issue is mitigated, and logit knowledge distillation improves performance as expected (see the next subsection).

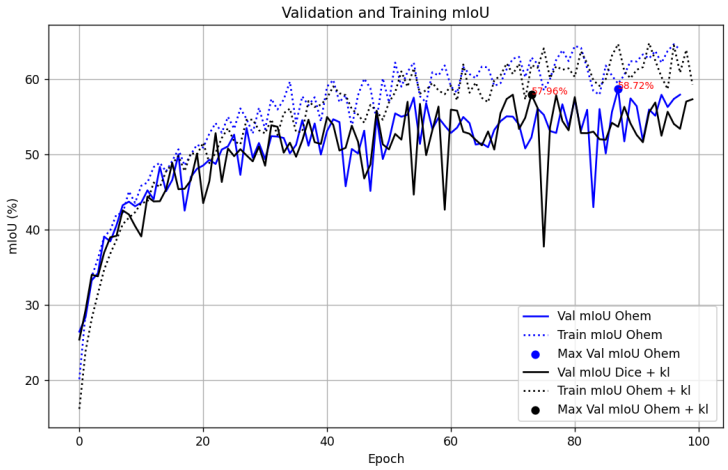

Figure 4.3: Visualization of mIoU over 100 epochs. All the experiments use a PIDNet-S as student. Early stopping at epoch 100 is applied

The experiments on PIDNet in figure 4.3 follow the same trend: logit Knowledge Distillation seems not to be particularly beneficial.

4.4.2 Intermediate feature distillation

In the second set of experiments, I used intermediate features that are closer to the backbone feature maps to avoid conflict between the final loss and the distillation target obtaining the result shown in table 4.6. For CWD loss, the temperature used is always 4 as in [37].

Backbone	**Loss**	**ACDC val mIoU**					**ECE**
		full	fog	night	rain	snow	
MobileNetv4	border	69.9	77.3	52.0	74.1	67.1	0.076
MobileNetv4	L2+border	72.4 (+2.5)	76.4	55.0	76.4	70.6	0.071
MobileNetv4	CWD+kl+border	72.7 (+2.7)	77.6	55.9	76.6	69.7	0.068
MobileNetv4	L2+kl+border	73.1 (+3.2)	78.1	55.2	76.3	71.6	0.070
ResNet18	border	67.7	75.7	50.5	66.3	68.7	0.059
ResNet18	L2+kl+border	68.1 (+0.4)	74.6	52.0	65.6	70.4	0.059
ResNet18	LAD+border	68.8 (+1.1)	76.5	52.7	66.6	72.5	0.065
ResNet18	L2+border	69.1 (+1.4)	75.1	52.8	67.4	71.5	0.061
ResNet18	CWD+kl+border	70.4 (+2.7)	75.7	53.7	68.2	73.4	0.059
ResNet18	CWD+border	71.2 (+3.5)	75.7	54.6	69.8	74.6	0.063

Table 4.6: Result of various distillation techniques. The segmentor is SwiftNet with indicated backbone, ECE is meassured on ACDC val.

As we can see from 4.6 even though best performing model (mIoU) not always make use of kl-divergence as main loss we can notice that the use of this loss can bring benefit especially in term of ECE, resulting in a more calibrated model.

Per-class IoU and AUPRC

We report IoU and AUPRC for each class for all models and distillation frameworks reported in 4.6. In order to understand which classes are more difficult to segment, to make considerations about difficulties related to size and rarities of the classes.

As a general trend, small and rare classes tend to score lower mIoU hence are difficult to segment correctly. In MobileNetv4 distilling with L2 + kl loss tends to favor the recognition of large objects, meanwhile, CWD + kl due to its formulation helps the student to learn to segment better smaller objects.

With ResNet backbone, we see that CWD contribution to extract important features is still more appreciable for smaller classes: The rider class, for example, is highly affected by this CWD loss as well as bycicle, motorcycle, and person.

In the following tables, we report the per-class metrics of the experiments reported in table 4.6, indicating by letter the method used in the experiments as follows:

A. MobileNetv4+Swiftnet-FPN, border loss

B. MobileNetv4+Swiftnet-FPN, L2+border loss

C. MobileNetv4+Swiftnet-FPN, CWD+kl+border loss

D. MobileNetv4+Swiftnet-FPN, L2+kl+border loss

E. ResNet18+SwiftNet-FPN, border loss

F. ResNet18+SwiftNet-FPN, L2+kl+border loss

G. ResNet18+SwiftNet-FPN, LAD+border loss

H. ResNet18+SwiftNet-FPN, L2+border loss

I. ResNet18+SwiftNet-FPN, CWD+kl+border loss

J. ResNet18+SwiftNet-FPN, CWD+border loss

—	Road	Sidewalk	Building	Wall	Fence	Pole	Light	Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle	Tot.
A	96.1	81.8	87.3	59.5	49.8	63.8	75.9	67.5	87.1	51.2	95.6	58.9	20.2	88.9	72.2	90.9	85.9	41.9	54.2	69.9
B	97.0	84.6	88.5	63.0	55.0	66.1	77.9	68.0	87.9	53.3	95.9	64.0	24.8	89.8	78.9	93.6	91.7	41.2	55.3	72.4
C	96.8	84.1	88.3	62.3	54.4	66.1	78.2	70.6	87.6	52.1	95.8	66.0	29.2	89.2	78.8	90.4	92.3	46.3	53.1	72.7
D	96.8	84.1	88.6	62.4	54.6	65.4	77.9	68.5	87.8	53.8	96.0	65.2	29.6	89.7	79.8	93.4	92.6	45.3	57.8	73.1
E	95.1	79.0	86.2	55.9	48.3	61.0	73.4	63.1	86.4	48.5	95.6	50.8	23.9	86.7	71.2	89.7	89.4	31.9	49.8	67.7
F	95.8	80.5	87.7	60.2	48.9	62.3	75.3	64.2	87.5	49.7	96.0	55.1	23.8	87.6	60.6	88.4	89.1	35.1	45.6	68.1
G	96.1	82.0	87.7	59.1	47.3	63.6	76.4	66.9	87.3	50.8	95.8	58.0	23.7	87.9	55.9	90.1	90.7	38.0	50.7	68.8
H	96.1	82.0	87.7	59.1	47.3	63.6	76.4	66.9	87.3	50.8	95.8	58.0	23.7	87.9	55.9	90.1	90.7	38.0	50.7	69.1
I	96.2	82.3	87.8	61.4	51.5	63.2	76.1	68.3	87.2	51.4	95.7	59.1	28.4	88.1	71.3	90.0	90.9	38.7	49.4	70.4
J	95.8	81.6	87.9	59.0	49.0	63.8	78.2	68.3	87.1	51.4	95.7	59.8	34.3	87.9	69.1	93.1	91.4	46.9	53.2	71.2

Table 4.7: IoU (%) of the method reported in list

Comparison of memory required and latency

As we can see from figure 4.4 and figure 4.5 the major improvements are in terms of memory usage when comparing algorithms with the same mIoU. In particular, MobileNetv4 with a relatively low number of parameters obtains comparable results with HRNet which has six times more parameters. As a downside on the hardware we tested the algorithms we notice that MobileNetv4, beside being designed for low latency, did not outperform ResNet18, so ResNet18 as a backbone for SwiftNet is more suitable for real-time application considering that we can achieve lower latency than the one reported maintaining same IoU due to downscaling to 0.8 image resolution.

—	Road	Sidewalk	Building	Wall	Fence	Pole	Light	Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle	Tot.
A	98.6	91.8	95.5	72.1	62.1	83.3	89.8	82.9	95.2	63.6	98.4	75.2	27.2	96.3	85.4	97.2	92.8	54.3	68.9	80.6
B	99.1	93.9	95.9	74.0	67.6	85.1	92.7	83.6	95.6	65.8	98.6	80.7	32.5	96.9	89.2	97.8	96.0	50.1	68.2	82.3
C	99.1	93.6	96.1	73.2	67.1	85.7	92.5	85.2	95.6	64.3	98.5	82.4	36.4	96.8	90.4	96.7	96.9	60.6	63.9	82.9
D	99.1	93.9	96.3	73.3	66.5	84.5	92.0	84.3	95.9	66.4	98.7	82.3	45.4	96.9	89.3	98.2	97.3	56.0	72.4	83.6
E	98.7	91.8	96.5	71.6	65.9	81.1	88.9	79.4	96.6	63.0	99.0	68.2	33.9	96.5	86.2	98.0	96.4	42.6	64.4	79.9
F	98.9	92.3	96.6	74.8	63.6	82.2	90.4	82.6	96.2	64.2	99.0	72.8	30.2	96.5	77.6	97.3	96.9	48.3	60.3	80.0
G	98.8	92.5	96.0	72.6	59.7	83.1	90.8	83.8	95.8	64.1	98.9	75.0	26.2	96.6	71.7	98.0	96.5	51.0	62.5	79.7
H	98.8	92.3	96.1	73.2	63.1	82.8	90.3	82.6	96.0	63.7	98.9	74.0	24.8	96.7	82.1	97.7	96.5	51.7	63.0	80.2
I	99.0	93.2	96.5	75.4	65.9	83.0	92.0	84.2	96.0	64.9	98.8	77.2	39.4	96.9	83.9	97.9	96.9	55.2	65.9	82.2
J	98.7	92.2	96.4	72.0	63.2	83.2	91.9	84.6	95.9	66.3	98.8	78.4	46.1	96.9	82.0	98.3	96.8	62.2	69.6	82.8

Table 4.8: AUPRC (%) of the method reported in list

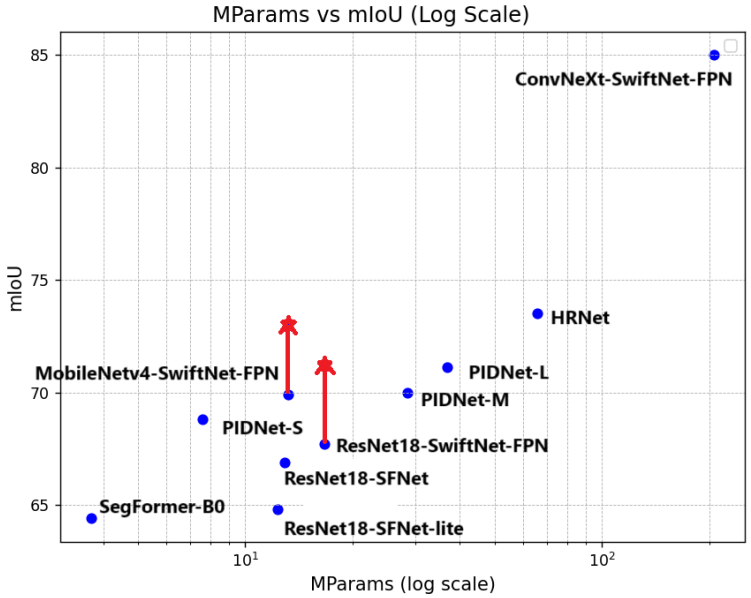

Figure 4.4: Distillation improvements in term of mIoU are reported with an arrow.

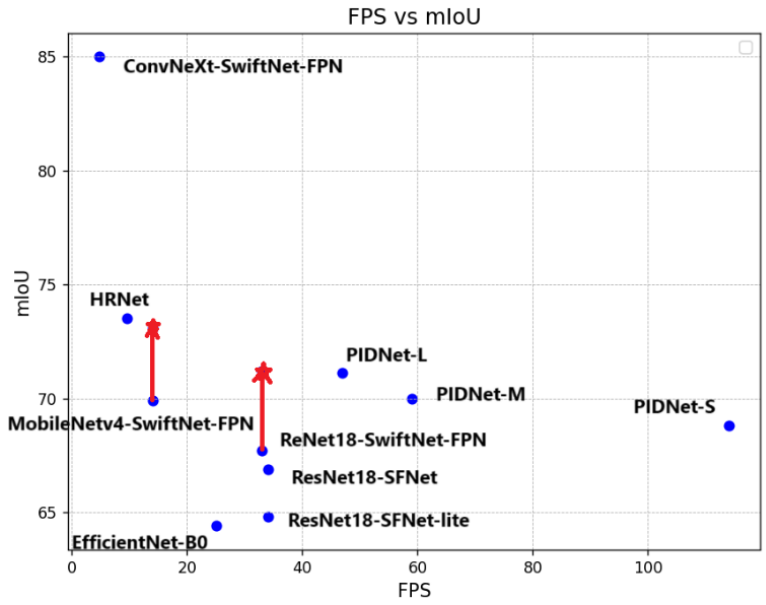

Figure 4.5: Distillation improvements in term of mIoU are reported with an arrow. FPS reported can be higher for distilled network since mIoU reported are obtained by a network that operate at 0.8 the original resolution.

Qualitative results

In the images selected, distilled networks seem to improve particularly segmentation quality of classes like sidewalks, vegetation, and grass (as we can see from figures 4.6, 4.8 and 4.9), on the other hand when it comes to segment traffic objects like cars or in general in more trafficked environments the distilled version of SwiftNet with ResNet18 seems to present more flaws in segmentation maps, as we can see in figure 4.9. Also, from what we can see in images reported, all networks easily segment also small objects like pole, traffic light, and traffic sign.

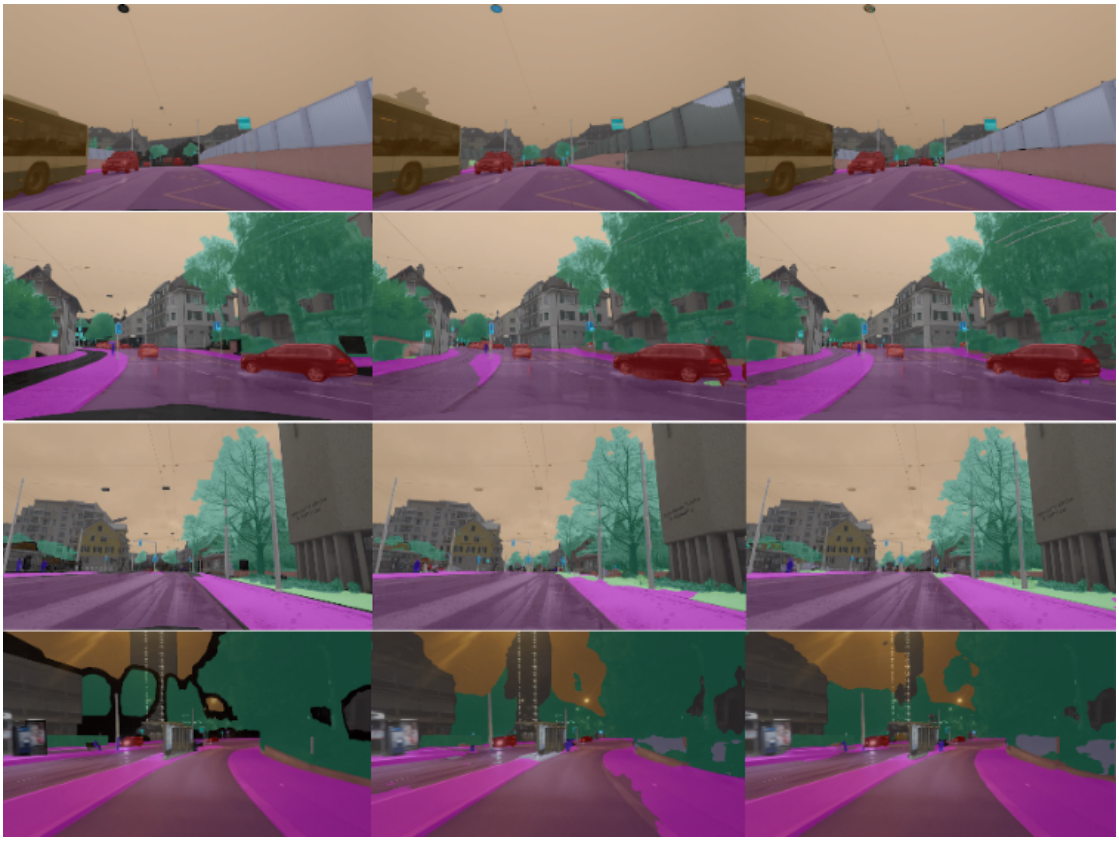

Figure 4.6: Visualization of distillation improvements: gt(left), SwiftNet+ResNet18(center), SwiftNet+ResNet18-distilled (right). From top to bottom meteo conditions are: fog, rain, snow, night. In non-trafficked context.

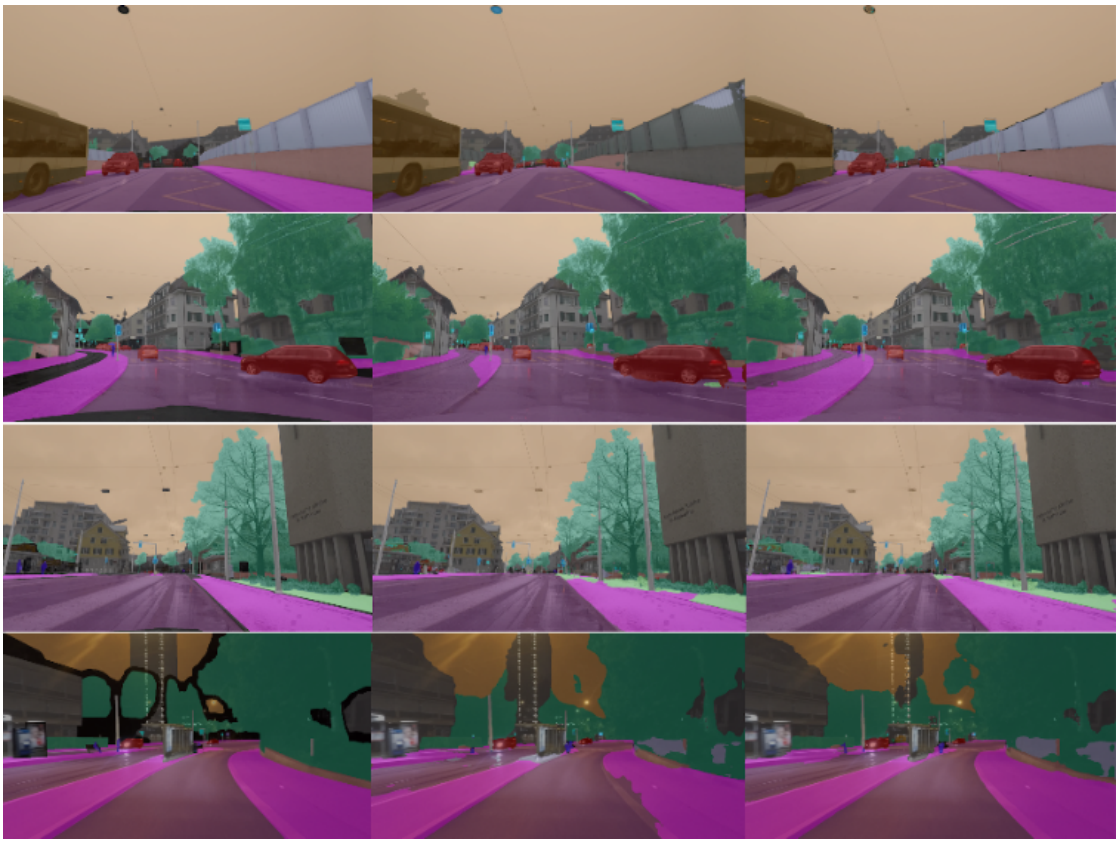

Figure 4.7: Visualization of distillation improvements: gt(left), SwiftNet+ResNet18(center), SwiftNet+ResNet18-distilled (right). From top to bottom meteo conditions are: fog, rain, snow, night. In trafficked context.

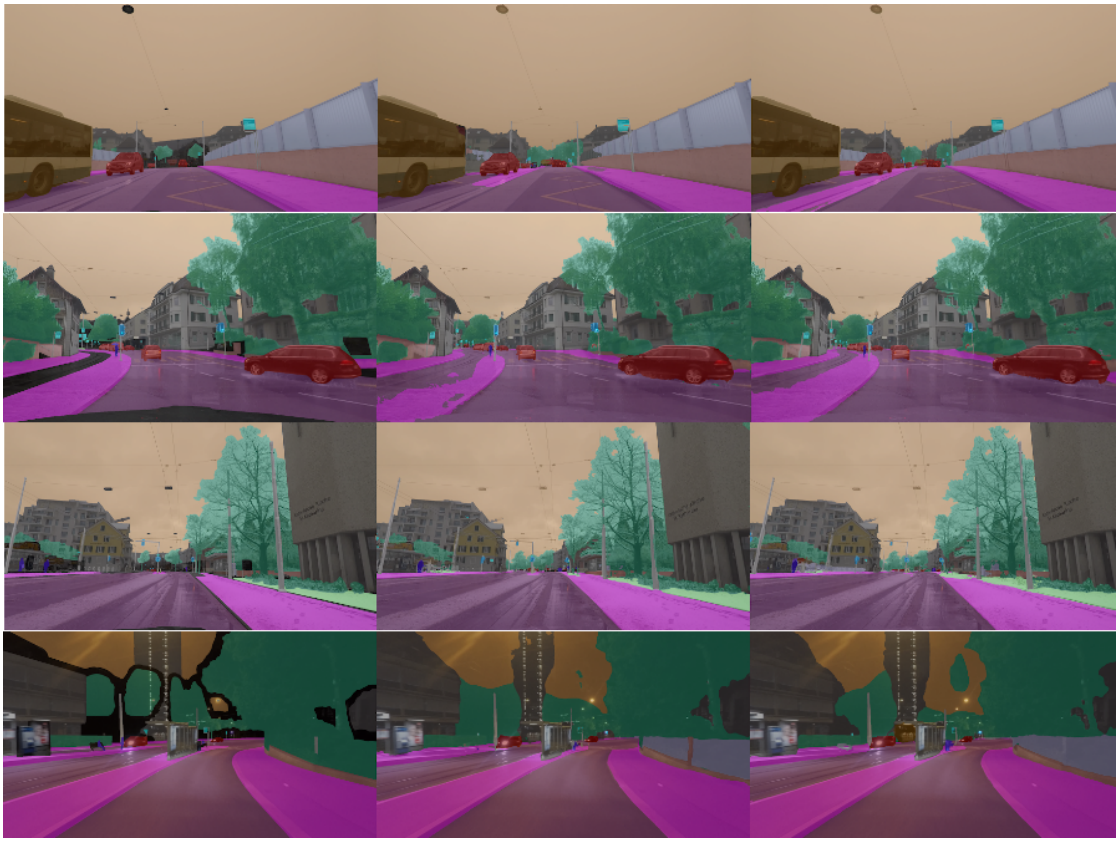

Figure 4.8: Visualization of distillation improvements: gt(left), SwiftNet+MobileNetV4(center), SwiftNet+MobileNetV4-distilled (right). From top to bottom meteo conditions are: fog, rain, snow, night. In non-trafficked context.

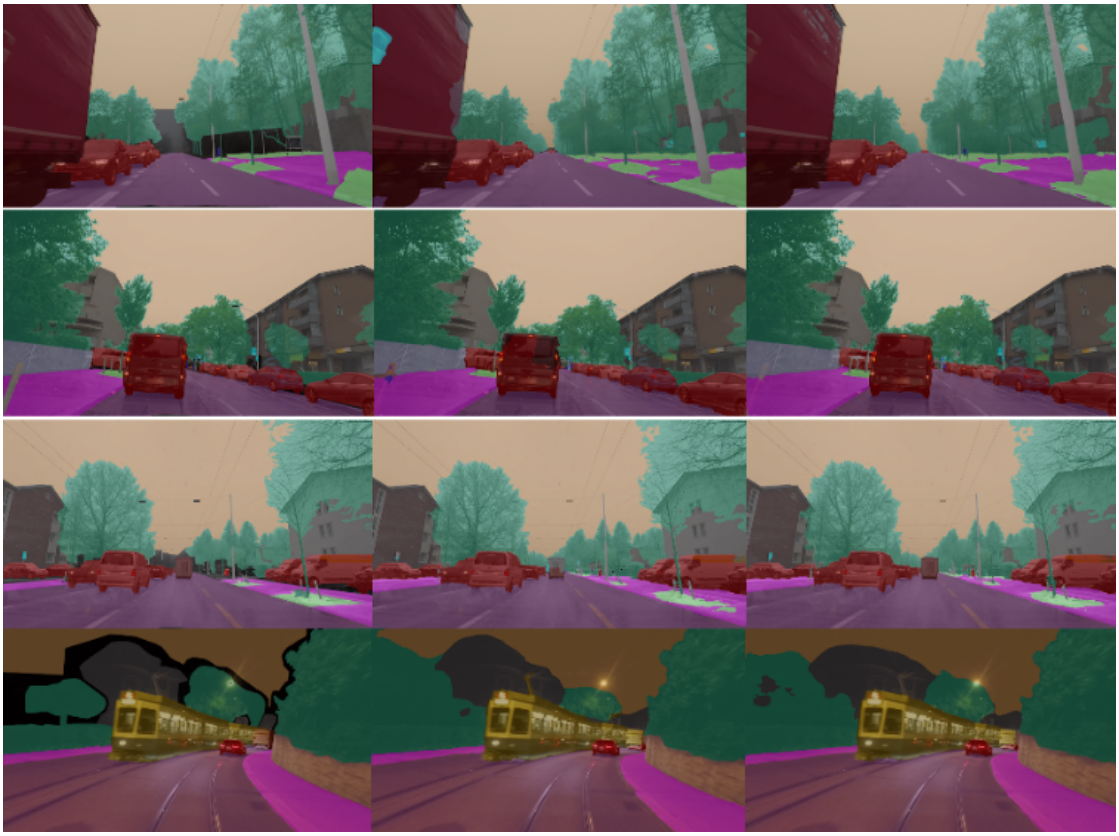

Figure 4.9: Visualization of distillation improvements: gt(left), SwiftNet+MobileNetV4(center), SwiftNet+MobileNetV4-distilled (right). From top to bottom meteo conditions are: fog, rain, snow, night. In trafficked context.

Chapter 5

Conclusions

The applied knowledge distillation methods bring improvements to the baselines, particularly in terms of network size, by achieving results that are competitive with networks that have six times the number of parameters of MobileNetv4+SwiftNet. From a latency point of view, we obtained results similar to those obtained by baselines.

From experiments emerged that using different losses from Cross-Entropy suggest us that in order to be beneficial with non-standard losses it is preferable to perform knowledge distillation in intermediate layers, nearer to backbone, rather than final layers near to logits. However, ECE suggests that logit distillation is beneficial in that sense.

Another important fact that can suggest which loss to use is the difficulty for the network to segment small classes: CWD in those cases may be more beneficial than L2 loss in fact, as we can see from per-class IoU in SwiftNet intermediate feature distillation experiments, it increases more IoU and AUPRC of small classes leading to a far better mIoU in unbalanced tasks or in networks that lack particularly the capability to segment small objects.

5.1 Future works

While effectiveness of kd between similar architecture has been proven, future works can focus on distilling ConvNeXt-L+SwiftNet into already good performing model such as PIDNet series architectures, this can lead to improved performances both in term of mIoU and latency, but will for sure require the design of a more complex adaptation module to fuse multiple branches of PIDNet into one feature map per residual block, moreover, due to the different information captured by different branches of PIDNet due to different loss usage selecting kl-div hyperparameters can be challenging (or lead to poor results a shown in our experiments), so intermediate feature distillation with adapter modules seem a more viable solution.

Another way to improve performance without degrading speed or increasing memory

occupation is to train the network on more datasets such as BDD100K and Citycapes, as is done for teacher network. Moreover, thanks to the supervision of knowledge distillation, we can train our model on unlabeled images, provided that we use kl-divergence as logit target, following a semi-supervised training approach.

Bibliography

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

[2] John Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989. URL https://proceedings.neurips.cc/paper_files/paper/1989/file/0336dcbab05b9d5ad24f4333c7658a0e-Paper.pdf.

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs, 2016. URL https://arxiv.org/abs/1412.7062.

[4] 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, 2017. URL https://arxiv.org/abs/1606.00915.

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation, 2018. URL https://arxiv.org/abs/1802.02611.

[6] Yudong Chen, Sen Wang, Jiajun Liu, Xuwei Xu, Frank de Hoog, Brano Kusy, and Zi Huang. Understanding the effects of projectors in knowledge distillation, 2023. URL https://arxiv.org/abs/2310.17183.

[7] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions, 2023. URL https://arxiv.org/abs/2205.08534.

[8] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation, 2022. URL https://arxiv.org/abs/2112.01527.

[9] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation, 2019. URL https://arxiv.org/abs/1910.01348.

[10] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016. URL http://arxiv.org/abs/1610.02357.

[11] 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. *CoRR*, abs/1604.01685, 2016. URL http://arxiv.org/abs/1604.01685.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. URL https://arxiv.org/abs/2010.11929.

[13] Blake Gella, Howard Zhang, Rishi Upadhyay, Tiffany Chang, Nathan Wei, Matthew Waliman, Yunhao Ba, Celso de Melo, Alex Wong, and Achuta Kadambi. Weatherproof: Leveraging language guidance for semantic segmentation in adverse weather, 2024. URL https://arxiv.org/abs/2403.14874.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[15] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023. URL https://arxiv.org/abs/1606.08415.

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

[17] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL http://arxiv.org/abs/1704.04861.

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

[19] Yann Lecun, Leon Bottou, Y. Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 12 1998. doi: 10.1109/5.726791.

[20] Xiangtai Li, Ansheng You, Zhen Zhu, Houlong Zhao, Maoke Yang, Kuiyuan Yang, and Yunhai Tong. Semantic flow for fast and accurate scene parsing, 2021. URL https://arxiv.org/abs/2002.10120.

[21] Zheng Li, Xiang Li, Lingfeng Yang, Borui Zhao, Renjie Song, Lei Luo, Jun Li, and Jian Yang. Curriculum temperature for knowledge distillation, 2022. URL https://arxiv.org/abs/2211.16231.

[22] Tao Liu, Chenshu Chen, Xi Yang, and Wenming Tan. Rethinking knowledge distillation with raw features for semantic segmentation. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1144–1153, 2024. doi: 10.1109/WACV57701.2024.00119.

[23] Tao Liu, Chenshu Chen, Xi Yang, and Wenming Tan. Rethinking knowledge distillation with raw features for semantic segmentation. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1144–1153, 2024. doi: 10.1109/WACV57701.2024.00119.

[24] Yifan Liu, Ke Chen, Chris Liu, Zengchang Qin, Zhenbo Luo, and Jingdong Wang. Structured knowledge distillation for semantic segmentation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2599–2608, 2019. doi: 10.1109/CVPR.2019.00271.

[25] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *CoRR*, abs/2201.03545, 2022. URL https://arxiv.org/abs/2201.03545.

[26] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 52:99–115, 1990. URL https://api.semanticscholar.org/CorpusID:15619658.

[27] Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant, 2019. URL https://arxiv.org/abs/1902.03393.

[28] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

[29] Jeremy Nixon, Mike Dusenberry, Ghassen Jerfel, Timothy Nguyen, Jeremiah Liu, Linchuan Zhang, and Dustin Tran. Measuring calibration in deep learning, 2020. URL https://arxiv.org/abs/1904.01685.

[30] Marin Oršić and Siniša Šegvić. Efficient semantic segmentation with pyramidal fusion. *Pattern Recognition*, 110:107611, 2021. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2020.107611. URL https://www.sciencedirect.com/science/article/pii/S0031320320304143.

[31] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *CoRR*, abs/1606.02147, 2016. URL http://arxiv.org/abs/1606.02147.

[32] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets, 2015. URL https://arxiv.org/abs/1412.6550.

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.

[34] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. doi: 10.1038/323533a0. URL https://doi.org/10.1038/323533a0.

[35] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. ACDC: the adverse conditions dataset with correspondences for semantic driving scene understanding. *CoRR*, abs/2104.13395, 2021. URL https://arxiv.org/abs/2104.13395.

[36] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019. URL https://arxiv.org/abs/1801.04381.

[37] Changyong Shu, Yifan Liu, Jianfei Gao, Zheng Yan, and Chunhua Shen. Channel-wise knowledge distillation for dense prediction, 2021. URL https://arxiv.org/abs/2011.13256.

[38] Trang Thi Kieu Tran, Sayed M. Bateni, Seo Jin Ki, and Hamidreza Vosoughifar. A review of neural networks for air temperature forecasting. *Water*, 13(9), 2021. ISSN 2073-4441. doi: 10.3390/w13091294. URL https://www.mdpi.com/2073-4441/13/9/1294.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

[40] Yukang Wang, Wei Zhou, Tao Jiang, Xiang Bai, and Yongchao Xu. Intra-class feature variation distillation for semantic segmentation. In *Proceedings of the European Conference on Computer Vision*, 2020.

[41] Yixuan Wei, Han Hu, Zhenda Xie, Zheng Zhang, Yue Cao, Jianmin Bao, Dong Chen, and Baining Guo. Contrastive learning rivals masked image modeling in fine-tuning via feature distillation, 2022. URL https://arxiv.org/abs/2205.14141.

[42] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers, 2021. URL https://arxiv.org/abs/2105.15203.

[43] Jiafeng Xie, Bing Shuai, Jian-Fang Hu, Jingyang Lin, and Wei-Shi Zheng. Improving fast segmentation with teacher-student learning, 2018. URL https://arxiv.org/abs/1810.08476.

[44] Jiacong Xu, Zixiang Xiong, and Shankar P. Bhattacharyya. Pidnet: A real-time semantic segmentation network inspired by pid controllers, 2023. URL https://arxiv.org/abs/2206.02066.

[45] Chuanguang Yang, Helong Zhou, Zhulin An, Xue Jiang, Yongjun Xu, and Qian Zhang. Cross-image relational knowledge distillation for semantic segmentation, 2022. URL https://arxiv.org/abs/2204.06986.

[46] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. *CoRR*, abs/1808.00897, 2018. URL http://arxiv.org/abs/1808.00897.

[47] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning, 2020. URL https://arxiv.org/abs/1805.04687.

[48] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer, 2017. URL https://arxiv.org/abs/1612.03928.

[49] Ying Zhang, Tao Xiang, Timothy M. Hospedales, and Huchuan Lu. Deep mutual learning. *CoRR*, abs/1706.00384, 2017. URL http://arxiv.org/abs/1706.00384.

[50] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network, 2017. URL https://arxiv.org/abs/1612.01105.