Semantic Segmentation Competition Report

FERREIRA Kévin a1882774

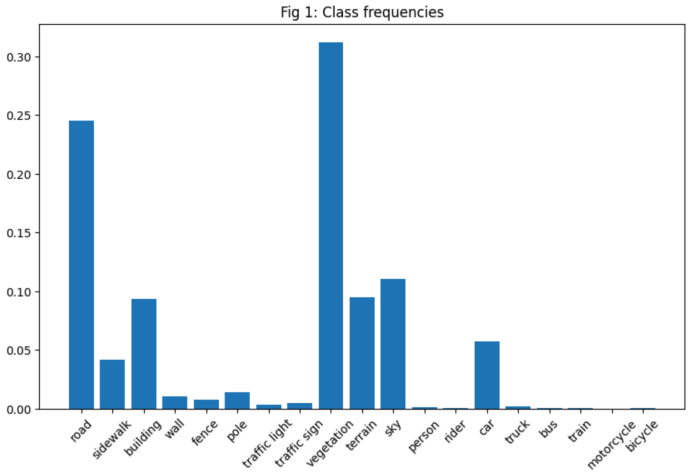
Introduction

Semantic segmentation aims to assign a semantic label to each pixel in an image. This technique enables the image to be segmented into regions corresponding to different classes of interest.

We are working here on a dataset composed of images taken by an on-board camera in a car. In the context of car driving, semantic segmentation plays a crucial role in providing a detailed understanding of the road environment, which is essential for autonomous driving and driver assistance systems. When performing semantic segmentation on images taken in a car, there are several specific challenges to overcome. First of all, the images can present significant variations in terms of brightness, weather conditions, occlusions and deformations.

Our aim is to develop a model capable of accurately segmenting these road scenes at a reasonable cost. To accomplish this task, we will use a convolutional neural network implemented with PyTorch. The basic model will be modified to improve its performance through various techniques such as data augmentation, modification of training parameters, architectural modifications and the use of a new loss function.

Dataset analysis



To begin with, we need to analyze the composition of the dataset. By analyzing the frequency of appearance of classes in the dataset images, we can see that certain classes are over-represented. For example, the vegetation and road classes are the most represented, unlike the motorcycle and train classes. This over-representation can lead to biases in model learning, where minority classes may be poorly learned compared to majority classes. This can lead to poor segmentation performance for under-represented classes. To compensate for this, we assign different weights to classes according to their frequency in the dataset when calculating the loss function. This gives greater weight to under-represented classes, helping the model to focus more on learning them.

Neural network analysis and modification

On the other hand, the basic model consists of a simple convolutional neural network composed of several convolution layers, ReLU activations, pooling layers and deconvolution layers to predict semantic masks from input images. The loss function is cross-entropy. The optimizer used for model training is Adam with a learning rate of 3e-4. However, this results in an accuracy of 0.2793 and uses a large number of resources (132.85G) for 180 epochs. To improve performance, the literature mentions several networks that are particularly effective for this task, including SegFormer, UNet and DeepLabV3. After testing the different networks, we found that the SegFormer network was the most efficient and least expensive: it's a neural network architecture combining encoders pre-trained on ResNet and a decoder.

There are several forms of SegFormer of varying depths, including resnet18, resnet50 and resnet101. The deeper the network, the higher the mIoU and the higher the number of GFlops used. A good compromise is to choose resnet50. However, for the following tests, resnet18 is used for rapid testing.

To prevent overlearning, dropouts have been added. They randomly disable certain neurons with a given probability, forcing the network to adapt and learn from different combinations of neurons.

In addition, to avoid the problem of gradient vanishing, skip connections are created to facilitate the flow of information in deep neural networks by allowing information to be transferred directly to subsequent layers rather than having to pass through intermediate layers.

Finally, normalization is used to make the network's input data more compatible and easier to process. Here, for example, batch normalization, which normalizes the activations of the

various network layers using the statistics of mini-sample batches during training, helps to stabilize and accelerate learning by reducing the problems associated with initializing weights and choosing learning rates.

These modifications increase accuracy while reducing computational costs. Indeed, the architectural modifications made to the residual layers enable the formation of deep networks with fewer parameters, thus reducing computational requirements.

Choice of optimizer and loss function

To improve convergence robustness, several optimizers were tested (Adam, SGD and RMS). However, the Adam optimizer gave the best compromise between convergence speed and result.

On the other hand, the Cross Entropy loss function with different weights for each class according to their frequency in the dataset maximizes the result.

Parameter selection

Finally, having chosen the neural network, loss function and optimizer, trade-offs between accuracy and cost can be achieved by also adjusting the learning rate, optimizer and batch size. As a result, the parameters batch size and learning rate were modified step by step to find the value that maximizes the mIoU.

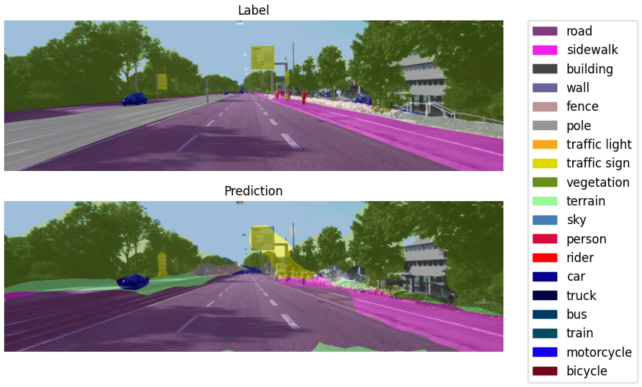
A large batch size means that several samples can be processed simultaneously, which can speed up training but leads to excessive memory usage. However, it stabilizes training by reducing the variance of weight updates compared to a smaller batch, as gradients are calculated from a larger set of samples. We therefore choose a batch size of 2, which is a good compromise between memory usage and segmentation accuracy.

On the other hand, the learning rate determines the magnitude of weight updates during network optimization at each iteration. A higher learning rate can accelerate the convergence of the optimization algorithm, as it allows for larger weight updates at each step. However, too high a learning rate can lead to oscillations, preventing model convergence. Conversely, too low a learning rate can lead to very slow convergence or stagnation in local minima. Therefore, after multiple tests, the learning rate 0.001 seems a good compromise.

Data augmentation

The dataset is made up of a small amount of data. So, to improve learning, it proved useful to increase the amount of data while remaining realistic. Certain images are then modified by vertical flipping, rotation, scaling and brightness adjustments: in this way, the dataset is enriched with new data. This enrichment improves robustness to variations and transformations, while reducing the risk of overlearning.

Limits and conclusions:



With all these modifications, we achieve a mIoU of 0.4009 (i.e. a 43% improvement on the baseline) using 114.02 G Flops (i.e. a 16.5% reduction in memory usage compared with the baseline).

Today, however, libraries such as *mmsegmentation* or *PIDNet* can achieve mIoU of the order of 80%. [1] Although the modifications and improvements discussed in this report can improve the performance of the semantic segmentation model, there are still limitations to be taken into account. Some modifications may be effective for some datasets, but not for others. It's important to experiment and refine parameters according to data characteristics. In addition, there may be additional techniques and approaches not explored in this report that could further improve model performance. In addition, by using computers with more powerful GPUs, the number of epochs and the depth of the network can be increased to train the model for longer periods of time.

	Optimizer	Loss	IoU	Flops (G)	Result
BASELINE: epochs = 180, batch size = 4, lr = 3e-4					
Baseline	Adam	Cross Entropy	0.2793	132.85	21.02 %
Baseline + Data Augmentation	Adam	Cross Entropy	0.2976	132.85	22.40 %
SEGFORMER: epochs = 50, batch size = 4, lr = 1e-3					
SegFormer + ResNet18	Adam	Cross Entropy	0.2903	33.97	85.46 %
SegFormer + ResNet18	RMS	Cross Entropy	0.2731	33.97	80.39 %
SegFormer + ResNet18	SGD	Cross Entropy	0.2578	33.97	75.89 %
SegFormer + ResNet18	Adam	Diss Loss	0.2865	33.97	84.34 %
SegFormer + ResNet18	Adam	Focal Loss	0.2894	33.97	85.19 %
SegFormer + ResNet50	Adam	Cross Entropy	0.3061	77.71	39.39 %
SegFormer + skip connections + ResNet50	Adam	Cross Entropy	0.3353	76.3	43.94 %
SegFormer + Dropout + Dilated Conv + ResNet50	Adam	Cross Entropy	0.3453	112.57	30.67 %
SegFormer + ResNet18 + Dropout + Dilated Convulution + Data Augmentation+ Normalization + Class Weight	Adam	Cross Entropy + Class Weight	0.3682	36.61	100.57 %
SegFormer + ResNet50 + Dropout + Dilated Convulution + Data Augmentation+ Normalization + Class Weight + Skip Connections	Adam	Cross Entropy + Class Weight	0.3891	115.24	33.76 %
SegFormer + ResNet18 + Dropout + Dilated Convulution + Data Augmentation+ Normalization + Class Weight + Skip Connections	Adam	Cross Entropy + Class Weight	0.3486	37.47	93.03 %
UNET: epochs = 50, batch size = 8, lr = 1e-3					
UNet	Adam	Cross Entropy	0.2852	79.64	35.81 %
DEEPLABV3: epochs = 50, batch size = 8, lr = 1e-3					
DeepLabv3 + Resnet50 (librairie)	Adam	Cross Entropy	0.3985	620.54	6.42 %
DeepLabv3 + Resnet50	Adam	Cross Entropy	0.3181	81.63	38.97 %
DeepLabv3 + Resnet101	Adam	Cross Entropy	0.3201	152.67	20.97 %
FINAL: epochs = 50, batch size = 4, lr = 1e-3					
SegFormer + ResNet50 + Dropout + Dilated Convulution + Data Augmentation+ Normalization + Class Weight + Skip Connections	Adam	Cross Entropy + Class Weight	0.4009	114.02	35.16 %

Table 1: Some of the test results

Sources

[1]Real-time semantic segmentation of urban landscapes. Papers with Code.
Available at: https://paperswithcode.com/sota/real-time-semantic-segmentation-on-cityscapes