NEURAL METHODS FOR SEGMENTING CITYSCAPES

Lotfy Abdel Khaliq
7013592
`loab00001@stud.uni-saarland.de`

Mahmoud Fawzi
7002629
`maib00001@stud.uni-saarland.de`

March 31, 2021

ABSTRACT

Image Segmentation is a significant problem. It has many important applications especially in the automotive industry. Cityscapes is a famous dataset used for this task. In this work we implement the R2U-Net architecture, train it on Cityscapes, and try to tune it to get better results. Afterwards, we implement a Fully Convolutional Network (FCN) with a ResNet backbone that outperforms R2U-Net.

1 Introduction

Image segmentation is a key process in Image Processing and Computer Vision. Its goal is to locate objects and their boundaries in given images. Segmented images can be a final output of a system or a preprocessed input to another system. Applications of segmentation include but aren't limited to medical imaging (e.g. locating tumors, analyzing anatomical structures), self-driving cars (object detection, brake light detection), and recognition tasks (Face, Iris, Fingerprint). There are many variants of this process. Instance segmentation is a variant where each instance of a given class should be identified separately (e.g. each tree in a set of trees). Our problem belongs to the semantic segmentation variant where all the instances of a given class are identified as the same thing.

Choosing the right model to solve a problem depends on many factors. A model might perform very well on a given dataset but perform poorly on another dataset for the same problem. The computational resources to train the best known models long enough such that they fit for a custom problem aren't always available. More data might be needed to tune the model for a custom task (e.g. increasing the recall of a given target class). It is the researcher's job to find this model while putting all this in mind.

The cityscapes dataset is one of the most famous labelled segmentation datasets along with Berkeley Deep-Drive (BDD) and Apollo Scape. The pixel of an image in this dataset can be classified into 1 of 35 labels typically found on the highway. This makes it a good benchmark for automotive applications.

R2U-Net is a network that provides promising results for binary segmentation tasks. It is an extension of the famous U-Net architecture. We try first to use this network to perform our task and check its behavior on a multi-class problem. Afterwards, we use the power of transfer learning to outperform R2U-Net through a Fully Convolutional Network (FCN) that uses a pretrained backbone and then apply DenseCRF post-processing technique to enforce local smoothness and consistency of neighbouring pixels.

2 Segmenting Cityscapes with R2U-Net

In this section we will explore in detail each of the techniques that we used to approach our problem. Firstly, we will introduce our implementation for R2U-Net and our attempts to tune it. In the following sections, we will show the architecture of the FCN that outperformed and explain the DenseCRF post-processing and its effect.

U-Net: Since segmentation output needs to be in the form of an image like the input, Networks used for segmentation tasks have the two following stages:

1- An encoding stage that extracts feature maps necessary for learning and executing the task.
2- A decoding stage that up-samples the extracted feature maps to the desired size.
U-Net is one of the most popular networks that follows this structure. It makes use of the global location of a pixel as well as its context, works with very few training samples, and gets a segmented output in a single forward pass.

R2U-Net: It is a variant of U-Net that incorporates the concepts of RCNNs (Recurrent Convolutional Neural Networks) and residual connections to upgrade the CNN units of U-Net to RRCNNs (Recurrent Residual Convolutional Neural Networks). These units have superior performance for object detection tasks and their integration into U-Net improves performance for binary segmentation tasks on medical datasets.

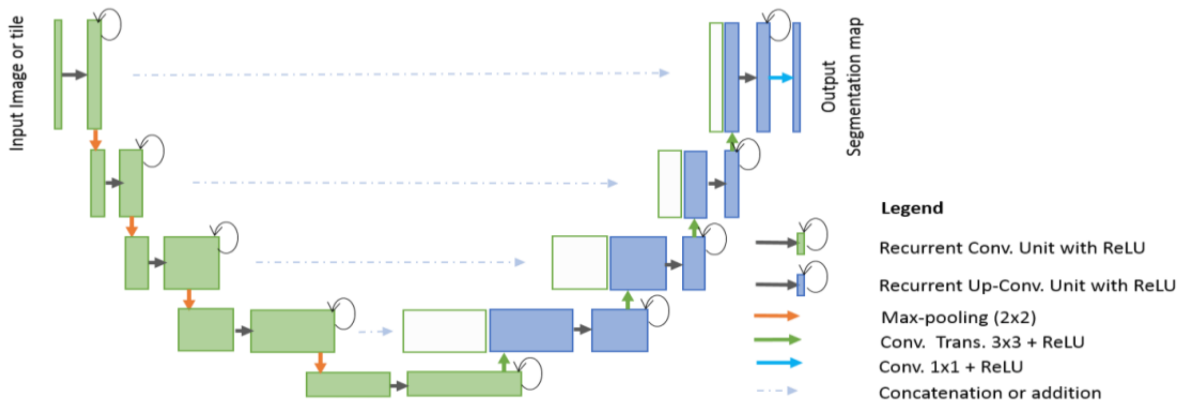

Figure 1: RU-Net architecture with convolutional encoding and decoding units using recurrent convolutional layers (RCL) based U-Net architecture. The residual units are used with RCL for R2U-Net architecture

Loading Data: Training data used are the left 8-bit images provided by Cityscapes. Since our task is pixel-level semantic labelling we use the variant of the fine labels with class IDs as our ground truth. We implement the function ***load_images*** to pick these files then we add the class ***cityscapes_dataset*** that inherits from the Pytorch class ***Dataset*** by implementing the required members. Images are resized from 2048x1024 to 512x256 using nearest neighbour interpolation to suit computational resources. This interpolation technique insures mapping pixels to their correct original classes. Some class labels are grouped into a single class label. This mapping is provided by Cityscapes' official repository [3] to have 20 classes relevant for segmentation instead of 35 fine-grained ones.

Network Implementation: Our code is implemented in Pytorch. The RRCNN unit is implemented using the built-in basic building blocks provided by the ***torch.nn*** module. 5 units are stacked with a max-pooling layer in between each two to form the encoder stage. The depth of the output increases as the input moves forward to get more features learnt. The decoder stage consists of 4 similar steps where each has an up-convolution layer followed by a concatenation with the output of the corresponding input layer to leverage the residual concept and an RRCNN that reduces the depth comes afterwards. The last layer is a 1x1 convolution that is analogous to fully-connected layers in classification networks. We have changed the size of the output of the final steps to 20 instead of binary output used in [6] to match the number of classes that we have. We also replaced the sigmoid cost function used in [6] with a softmax.

Tuning Hyperparameters: We start with the learning process on all the 35 classes provided by Cityscapes with a learning rate of $1 * 10^{-4}$ but the value of the loss diverges after few epochs. We reduce the learning rate to $1 * 10^{-6}$ but the same behavior happens again. After grouping the 15 similar classes into one class using Cityscapes mapping, the loss functions converges during the learning process. We used a batch size 10 to adapt to computational resources. We also tried to add dropout to layers with many parameters and we tried L2 regularization.

3 Transfer Learning with FCN

In this section, we try to improve our results from part 2 using Fully Convolutional Network equipped with transfer learning technique, then we use DenseCRF post-processing [4] to further enhance the results.

3.1 FCN

Traditional CNNs usually use convolutional layers followed by fully connected layers in the end to convert the original two-dimensional feature map into a one-dimensional fixed-length feature vector of a specific length indicating the probabilities that the input image belongs to each class. This results in loss of spatial information between the pixels and such information is vital for tasks such as image segmentation when we try to output an image of the same size as the input image. Additionally, this limits the input of CNNs to a specific size since the number of parameters in the fully connected layers depend on the input size.

Unlike traditional CNNs, FCNs can accept input images of any size since it converts the fully connected layers into convolutional layers by treating them as convolutions with 1x1 kernels that cover the entire input regions. It uses an encoder-decoder architecture which consists of convolutional layers to get the feature representation followed by deconvolutional (sometimes called Transpose Convolution) layers to get an output with dimensions same as the input.

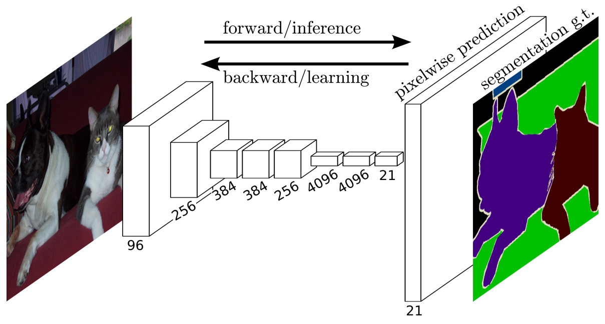

Figure 2: FCN encoder-decoder architecture [5]

3.2 Skip Connections in FCN

One major issue with in-network down-sampling in FCN is that it reduces the resolution of the input by a large factor which results in loss of spatial information between the pixels, thus during up-sampling it becomes very difficult to reproduce the finer details even after using sophisticated techniques like Transpose Convolution. As a result a coarse output is obtained (Figure 3).

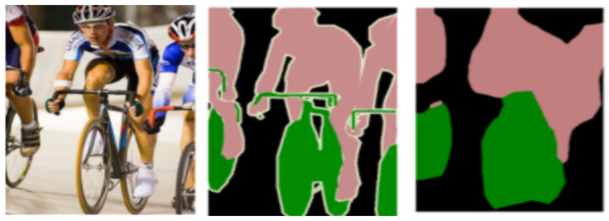

Figure 3: Left: Input image. Middle: Ground Truth. Right: Coarse output from FCN

One-way to deal with this is by adding ‘skip connections’ in the up-sampling stage from earlier layers and summing the two feature maps. These skip connections provide enough information to later layers to generate accurate segmentation boundaries. This combination of fine and coarse layers leads to local predictions with nearly accurate global (spatial) structure. This process is illustrated in Figure 4. Adding Skip connections can be considered as a Boosting method for a FCN, which tries to improve performance of layers by using predictions (feature maps) from previous layers.

3.3 Transfer Learning

Transfer learning is using a trained network as a starting point for the model to be trained for the new task. This is very useful when this starter network is trained to extract features that are required for the new task. It will do the job with efficiency that requires a big amount of training data and costly computational resources to achieve by training. ResNet is a very popular backbone when it comes to tasks dealing with images. It is based on adding residual connections to overcome the degradation of performance when stacking a large number of layers.

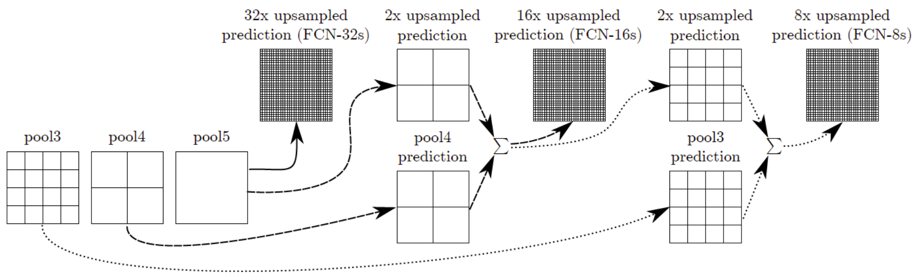
32x upsampled prediction (FCN-32s)
2x upsampled prediction
16x upsampled prediction (FCN-16s)
2x upsampled prediction
8x upsampled prediction (FCN-8s)
pool3
pool4
pool5
pool4 prediction
pool3 prediction

Figure 4: FCN-32s : Upsamples at stride 32, predictions back to pixels in a single step (Basic layer without any skip connections). FCN-16s : Combines predictions from both the final layer and the pool4 layer with stride 16, finer details than FCN-32s. FCN-8s : Adds predictions from pool3 at stride 8, providing even further precise boundaries.

4 DenseCRF Post-processing

One of the most common ways to improve the output of a segmentation network is using post-processing techniques such as superpixels [2] and region proposals [1] to get more accurate results. One of the most popular techniques is proposed by Philipp Krahenbuhl [4], in which a fully connected conditional random field model is used to enforce pixel consistency by modeling spatial location and color difference between each pixel and every other pixels. In the fully connected pairwise CRF model, the image is thought of as a graph in which each pixel has two potentials: unary potential and pairwise potential, which can be written as an energy function:

$$E(x) = \sum_i \psi_u(x_i) + \sum_i \psi_p(x_i, x_j) \tag{1}$$

The unary potential $\psi_u(x_i)$ is the output of the a classifier, which is the segmentation network in our case, and the pairwise potential $\psi_p(x_i, x_j)$ models the spatial location and color relationships between all the pixels. Minimizing this energy function results in the (MAP) estimate of the most likely assignment of each pixel to a certain class. The pairwise potentials in the model have the form

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) \sum_{m=1}^{k} w^{(m)} k^{(m)}(f_i, f_j) \tag{2}$$

where $\mu(x_i, x_j)$ is pots model compatibility function and each $k^{(m)}(f_i, f_j)$ is a the summation of three Gaussian kernels that have three parameters α,β, for location and color kernels jointly, and γ for location kernel only.

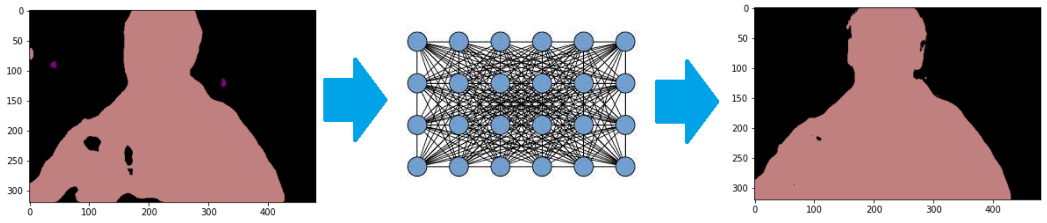

Figure 5: Left: Input image. Middle: DenseCRF model. Right: Refined output mask

Minimizing Energy Function: We used MeanField Approximation algorithm implemented in **denseCRF.py** to minimize this energy function. Since minimizing the energy function is NP-Hard, MeanField Approximation apply the above kernels to the CRF model itereatively and update the unaries of each pixel in each iteration until it converges. Usually, it takes from 6 to 15 iterations to converge. [4]. We used 10 iterations, with α = 67, β= 3, and γ = 3.

5 Results

As mentioned earlier, some experiments cause the loss function to diverge so, we won't present them. Although the number of classes is reduced from 35 to 20, The task of segmenting images with this number of classes remains hard.

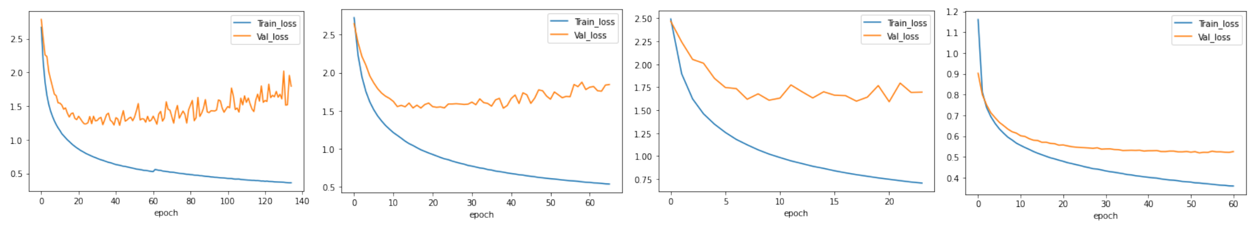

Figure 6: Training and Validation Loss per Epoch for different Experiments (a) R2U-Net without Regularization (b) R2U-Net with Dropout (c) R2U-Net with L2 Regularization (d) FCN

As a result, adding dropout (0.2 dropout rate used) to layers with many parameters or using L2 regularization (0.1, 0.00001 lambda values were tried) doesn't improve the validation loss and the model has large bias as shown in Figure 6. It is also clear that FCN outperforms all variants of R2U-Net although the best R2U-Net variant is trained for double the number of epochs.

A more detailed comparison between R2U-Net and FCN using 5 evaluation metrics can be shown in the following table:

	Accuracy	**Recall** (Macro Average)	**Recall** (Weighted Average)	**F1 Score** (Macro Average)	**F1 Score** (Weighted Average)	**Jaccard Score**	**Specificity** (Macro Average)	**Specificity** (Weighted Average)
R2U-Net	0.638	0.197	0.638	0.189	0.609	0.136	0.978	0.922
FCN	0.840	0.515	0.840	0.543	0.832	0.417	0.990	0.969

It is clear the FCN outperforms R2U-Net for **all** the 5 metrics. It is also clear that some classes really get poor results leading to a poor Jaccard score since it is calculated using macro and not weighted average.

5.1 DenseCRF Results

In Figure 7 the effect of DenseCRF post-processing is demonstrated. The yellow spot in Figure 7(c) is a false segmentation for pixels that belong to a building. DenseCRF enforces consistency between neighbouring pixels and solves this problem as shown in Figure 7(d). However, the problem with this concept is that classes with high support (large proportion of pixels) sometimes swallow pixels that don't belong to them like some pixels that belong to the black and rose classes in Figure 7(c) and 7(d) leading to a higher recall for high support classes but overall less recall. Overall, the post-processing leads to a better precision and a better F1 score for some low support classes. This can be useful for specific application that are interested in these classes in particular or for problems that have balanced classes in their datasets.

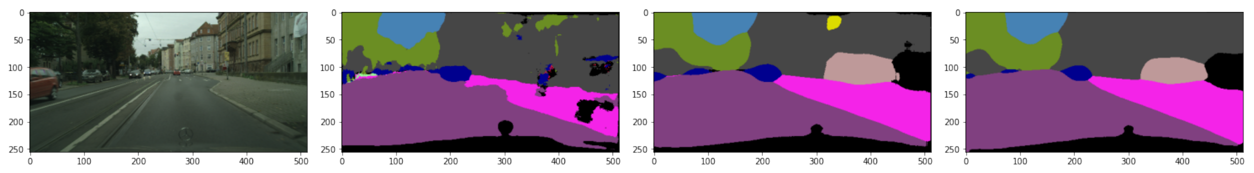

Figure 7: (a) Input Image (b) R2U-Net Output (c) FCN Output (d) Postprocessing Output

Class	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Support	12.40%	32.96%	4.74%	19.21%	0.64%	0.72%	1.29%	0.17%	0.58%	15.18%	0.73%	2.94%	1.14%	0.19%	5.71%	0.26%	0.34%	0.09%	0.07%	0.62%
FCN Precision	0.804	0.920	0.663	0.825	0.635	0.448	0.337	0.406	0.580	0.862	0.577	0.854	0.526	0.335	0.834	0.643	0.640	0.451	0.594	0.553
FCN+DenseCRF Precision	**0.841**	0.881	**0.717**	0.784	**0.711**	**0.518**	**0.456**	**0.671**	**0.770**	0.817	**0.633**	**0.878**	**0.602**	**0.462**	**0.848**	**0.764**	**0.678**	**0.606**	**0.635**	**0.594**
FCN F1	0.723	0.935	0.680	0.864	0.420	0.434	0.144	0.146	0.289	0.848	0.537	0.875	0.540	0.234	0.863	0.611	0.591	0.332	0.229	0.537
FCN+DenseCRF F1	0.660	0.922	**0.689**	**0.868**	**0.431**	**0.439**	0.067	0.087	0.188	**0.861**	0.507	0.869	0.503	0.198	0.863	**0.683**	**0.605**	**0.355**	0.166	0.510

5.2 Confusion Matrix

Comparing the confusion matrices for R2U-Net and FCN provides more details about their performance. As shown in Figure 8 and Figure 9, R2U-Net has a more sparse confusion matrix with some zeroes on the diagonal. Some classes are never classified right leading to a precision and recall of 0 for these classes. On the other hand, there is no 0 values on the diagonal of FCN's matrix.

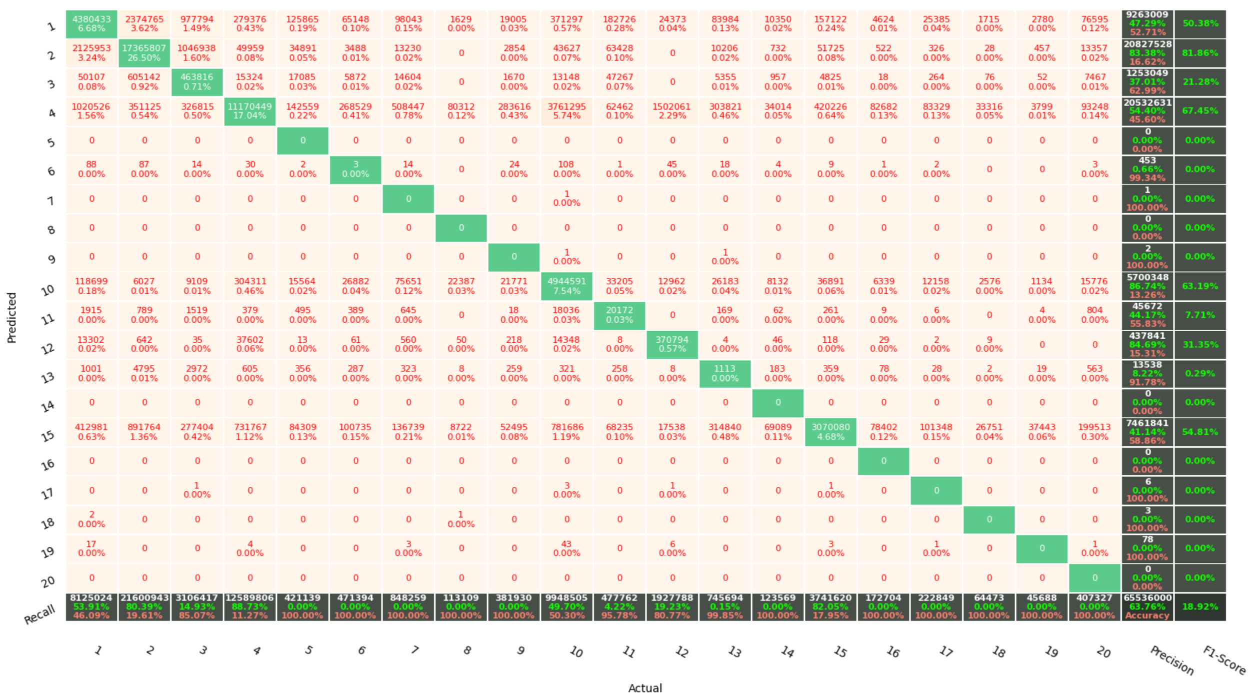

Figure 8: Confusion Matrix for R2U-Net

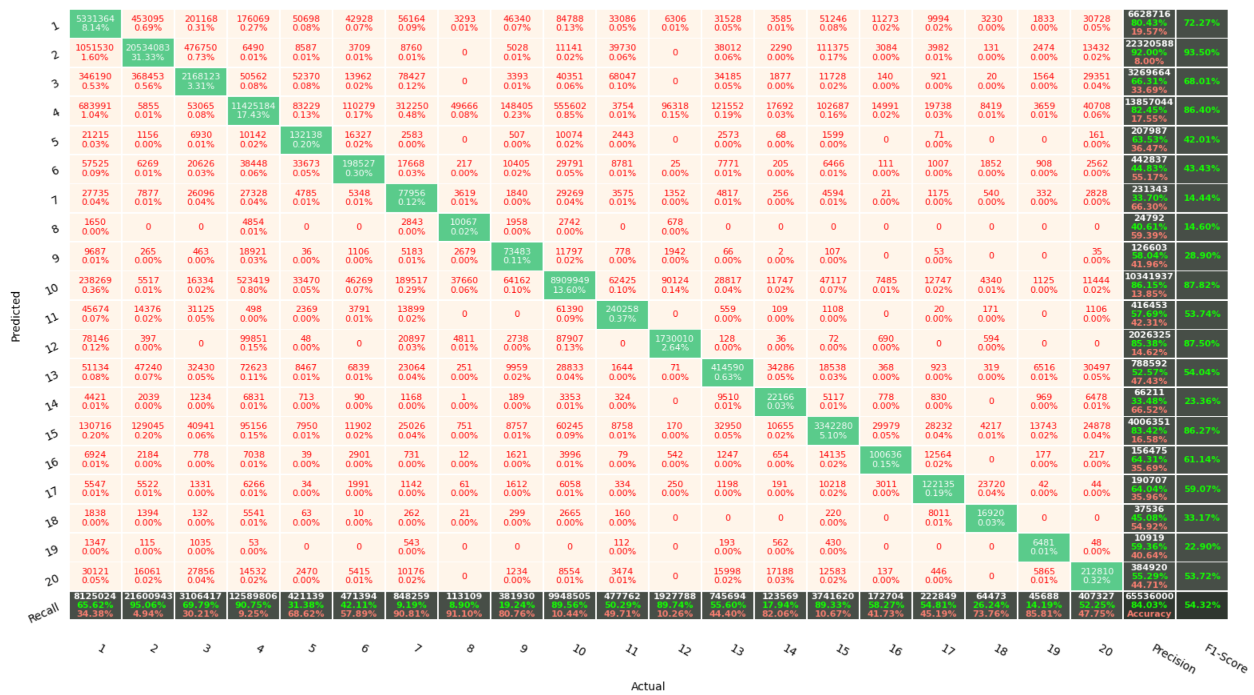

Figure 9: Confusion Matrix for FCN

6 Future Work

A lot of experiments can still be made where hyperparameters can be further tuned to get better measures for accuracy and other metrics. The number of experiments that can be made is always limited by the availability of computational resources. Typical experiments could be letting the models train for a relatively larger number of epochs to escape potential saddle points, increasing the number of parameters to provide the model with more flexibility to learn more features, and exploring other approaches such as [7] and [8] try to tackle the problem by building a model designed specifically for dense predictions by using dilated convolutions to systematically aggregate multi-scale contextual information without losing resolution.

7 Conclusion

In this work we approached the problem of segmenting images from the Cityscapes dataset with different neural methods. A Fully Convolutional Network (FCN) outperformed R2U-Net after a few number of epochs which shows the power of transfer learning and how it compensates for low computational resources. We used a ResNet backbone because of its performance for tasks dealing with images. We also applied post-processing that enhances F1 Score and Precision for most classes (low support classes) in exchange for the precision and F1 Score for high support ones. This can be relevant for applications interested in such classes and can significantly improve results for balanced datasets.

References

[1] R. Girshicki B. Hariharan P. Arbel aez. "Simultaneous detection and segmentation." In: *ECCV* (2014).

[2] L. Najman C. Farabet C. Couprie. "Learning hierarchical features for scene labeling.Pattern Analysis and Machine Intelligence". In: *IEEE* (2013).

[3] "https://github.com/mcordts/cityscapesScripts/blob/master/cityscapesscripts/helpers/labels.py". In: (2018).

[4] Philipp Krähenbühl and Vladlen Koltun. "Efficient inference in fully connected crfs with gaussian edge potentials". In: *arXiv preprint arXiv:1210.5644* (2012).

[5] Jonathan Long, Evan Shelhamer, and Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. 2015. arXiv: `1411.4038 [cs.CV]`.

[6] Tarek M. Taha Md Zahangir Alom Mahmudul Hasan. "Recurrent Residual Convolutional Neural Network based on U-Net (R2U-Net) for Medical Image Segmentation". In: (2018).

[7] Fisher Yu and Vladlen Koltun. *Multi-Scale Context Aggregation by Dilated Convolutions*. 2016. arXiv: `1511.07122 [cs.CV]`.

[8] Yuhui Yuan, Xilin Chen, and Jingdong Wang. *Object-Contextual Representations for Semantic Segmentation*. 2020. arXiv: `1909.11065 [cs.CV]`.