Object detection in images using semantic segmentation and CNN models

TDDE19 Project Report

Alfred Hagberg - alfha306
Aron Gosch - arog0305
Erik Örjehag - erior950
Kristian Sikiric - krisi211
Mustaf Musse - musab250
Niclas Byrsten - nicby889

Supervisor : Cyrille Berger
Examiner : Cyrille Berger

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Contents

1 Introduction

Recent years have seen much improvement in object recognition. Many newly developed sophisticated Artificial Intelligence (AI) systems, like self-driving cars, require trustworthy methods to generate precise knowledge about the surrounding environment [6]. One of the key applications used for this problem today is *Semantic Segmentation* which refers to the process of clustering parts of an image together which belong to the same kind of object class [6]. Semantic segmentation provides a way for machine learning algorithms such as *Convolutional neural networks* (CNN) to detect objects in images, by training on annotated images. The annotation process consists of labeling each pixel in the image with a corresponding class, which can be used as *ground truth* in a network model to obtain a label map by classifying each region based on a selected number of features [7].

1.1 Problem

One of the practical challenges in robotics is to acquire enough knowledge of the surrounding environment to make a decision. Since this decision-support can be crucial to accomplish an assigned task it is important that a model can deliver accurate and credible results. A possible approach to detect objects in an image is to use CNN models trained for semantic segmentation.

1.2 Approach

To solve this problem the group decided to examine how the choice of CNN architecture can affect the performance on the same problem instance. The architectures that were chosen for this project were *VGG16*, *Unet* and *Segnet*. The networks were trained on a number of different sized datasets for semantic segmentation and the models were evaluated using metrics such as *Accuracy*, *Precision*, *Recall*, *F1-Score*, *Mean Intersection Over Union (mIOU)* and *loss over epochs*.

1.3 Delimitations

The scope of the project has been limited by a number of aspects. One of these is the hardware available during the course. To limit the scope of the project the group have chosen to focus on semantic segmentation, instead of tracking and 3d-reconstruction. This provides grounds for further work. The results are tied to the choice of datasets. To limit training time of the CNN models only a certain amount of objects is detected in the images.

2 Method

This sections describes the relevant theoretical background needed to understand the scope of the project. All CNN architectures were implemented using Keras and tensorflow 1.9 backend.

2.1 CNN

CNN (Convolutional Neural Network) is a neural network architecture that uses the mathematical operation convolutions in layers with images as input.

2.1.1 Types of layers

A CNN can consist of several different types of layers, here are the types of layers used in this report described.

1. **Convolution:** A convolution layer uses the convolution operation and in most cases an activation function.
2. **Pooling:** Pooling reduces the resolution of the image e.g. by using the maximum value of pixels in areas in the image (Max pooling).
3. **Deconvolution:** A deconvolution layer is a convolution layer that can be used to scale up the image.
4. **Unpooling:** Unpooling performs the inverse operation of pooling by scaling up the image.
5. **Batch normalization:** Batch normalization normalizes the values for one input batch.

2.2 Dataset

The Cambridge-driving Labeled Video Database (CamVid) is a collection of videos with object class semantic labels, complete with metadata. The video is collected by researchers in Cambridge and from captured from the perspective of a driving automobile. Each pixel in the video frames has been marked with ground truth labels that associate that pixel to one

of 32 different semantic classes. Due to memory limitations of the hardware only a subset with 12 classes and 700 labeled classes is used in this study. This subset is further divided into training, validation and test sets with pairs of unprocessed images and annotations that contain the ground truth for each pixel in the image. A list of the classes used in the study can be found in table 2.1. [2, 3]

Class	Description
0	Sky
1	House
2	Pole
3	Road
4	Sidewalk
5	Tree
6	Sign
7	Fence
8	Car
9	People
10	Bicycle
11	None

Table 2.1: The classes of the CamVid dataset used in the study.

2.3 Segnet

Segnet is a deep fully convolutional network that is used for pixel-wise segmentation [1]. Segnet was developed by the research group in Computer vision and robotics at Cambridge university. The architecture of Segnet can be seen in figure 2.1. The first part of the network can be seen as the encoder part, and the second part as a decoder. This type of architectures are called deep encoder-decoder architecture [1]. The encoder part of Segnet is the same as the fully convolutional part in VGG16 (see 2.4.1). For each encoder layers, there is a decoder layer and lastly a pixel wise classification layer (Softmax classifier) [1].

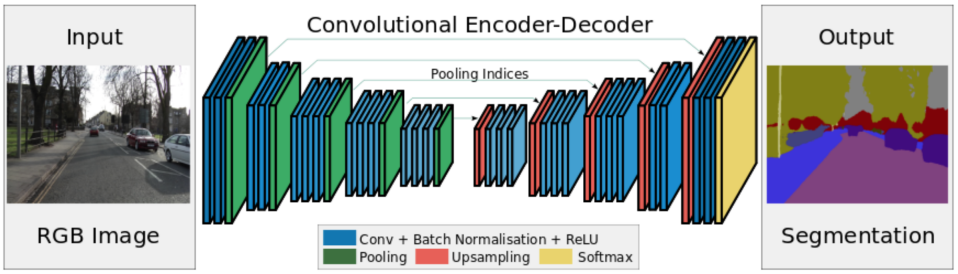

Figure 2.1: Segnet architecture. The pooling used is a max pooling and up-sampling is done by reusing the max indices from the respective pooling layers. [1]

The encoder part consists of 13 convolutional layers. After each convolution, a batch normalization is done and then a element-wise *rectified-linear non-linearity* (ReLU) activation function is applied [1]. The ReLU function can be expressed as $max(0, x)$ were x is the input tensor. The encoder part also consists of five pooling layers that perform max pooling. What makes Segnet special is that the indices from the max poolings are stored and reused in the decoder. This ensures no information is lost when up-sampling [1].

The decoder part consists of five up-sampling layers, 13 convolutions, batch normalization's and ReLU activation functions. Lastly a Softmax classifier is applied, which returns class probabilities for each pixel in the image. The pixel is then colored according to the most probable class [1].

2.4 FCN with CRF postprocessing

This approach uses a combination of a fully-convolutional deep learning model and the statistical modeling method Conditional Random Fields (CRF). In this report, We tested the model of FCN-CRF that Hao Zhou et al. wrote in their paper *Image Segmentation Based on FCN-CRF Model* [11].

The process of the FCN-CRF model, as shown in figure 2.2, takes the original image which is then trained using the FCN and the output is then sent to the CRF part of the model. Hoa Zhou et al. explained in their paper [11] that this model is trained end-to-end which the error is then transferred from output to the input. The reason is that FCN alone can not get high accuracy in image segmentation because of the maxpooling layers which can cause that some parts of the image to be discarded.

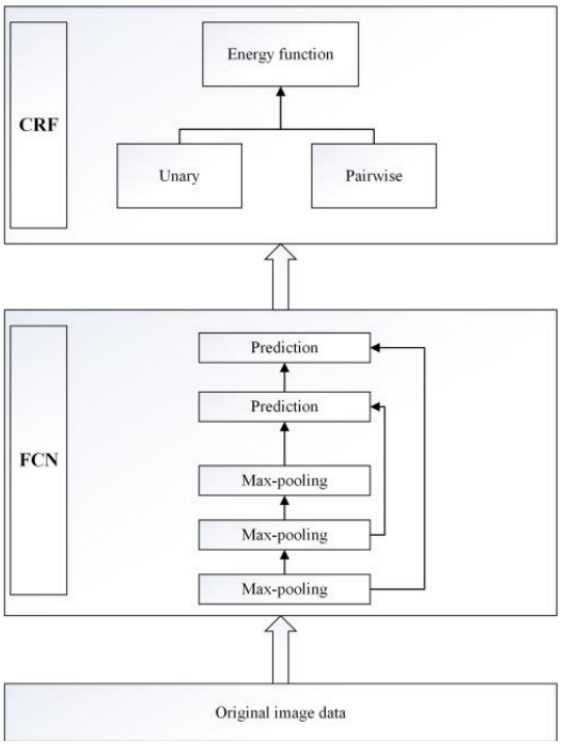

Figure 2.2: The process of FCN-CRF [11]

2.4.1 FCN (VGG16)

The FCN architecture used is the VGG16 architecture. The model takes an RGB image as input and outputs one image for each class with the pixel-wise probabilities. The architecture consists of convolution with *ReLU* (Rectified Linear Unit) activation, batch normalization and max pooling in the down sampling phase. Three predictions are made in three different resolutions from the In the up-sampling phase, it consists of deconvolution and summing of the three predictions from the down-sampling layers. [10]

$$ReLU(z) = max(0, z) \tag{2.1}$$

Layers

This section gives a description of the layers in figure 2.3.

- **Conv + ReLu layers**
  Convolution layers that uses 3×3 kernels and ReLu activation function.
- **Pool layers:**
  These layers are maxpooling layers that downsamples the image to a
- **Predict layers:** These layers use 1×1 convolution to output a predictions from three different levels in the down-sampling phase.
- **Deconv layers**
  The deconvolution layers uses strides to upsample the image. The strides are set to 2, 4, and 8 to Deconv1, Deconv2, and Deconv3 respectively.

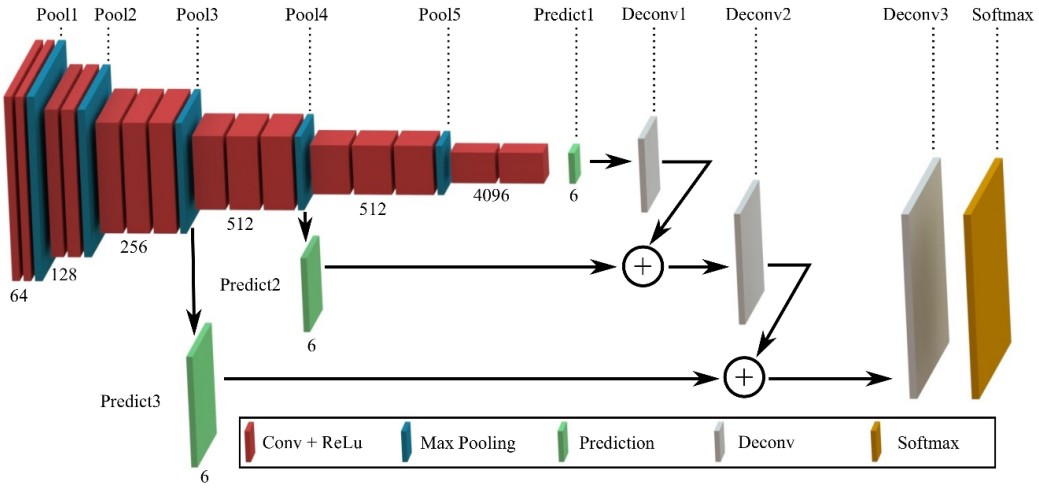

Figure 2.3: Vgg16 architecture [10]

2.4.2 CRF

CRF is a discriminative probability graph model that combines the best properties of hidden Markov models and maximum entropy model [5]. As Hao Zhou et al. wrote in their paper [11], CRF mdoels have a very strong reasoning ability. This gives CRF models the ability to train and inference with complex, overlapping and non-independent features.

The top part of figure 2.2 shows the different stages of the CRF part. The CRF model has three main parts: the Unary function, the Pairwise function and the Energy function which is a combination of the unary and pairwise functions. The formulas mentioned below, taken from Hao Zhou et al. paper [11], explains the CRF model's main parts. Unfortunately, we refer to the original paper [11] for more explanation and understanding of what these formulas do.

$$P(Y|X) = \frac{1}{Z(X)} exp(-E(Y,X)) \tag{2.2}$$

$$E(Y,X) = \sum_{p \in N} \Phi(y^p, x) + \sum_{(p,q) \in S} \Phi(y^p, y^q, x) \tag{2.3}$$

The first equation (2.2) is the Gibbs distribution of the CRF model and the second equation (2.3) is the *energy function*. The first part of equation 2.3 is the unary function and the second part of equation 2.3 is the pairwise function.

2.5 Unet

U-Net is a fully convolutional neural network that was developed for biomedical image segmentation. The network architecture is based on FCN, but its has been modified to work with fewer training images and yield more precise segmentations. It is called Unet because of its u-shaped architecture that can be seen in figure 2.4. [8]

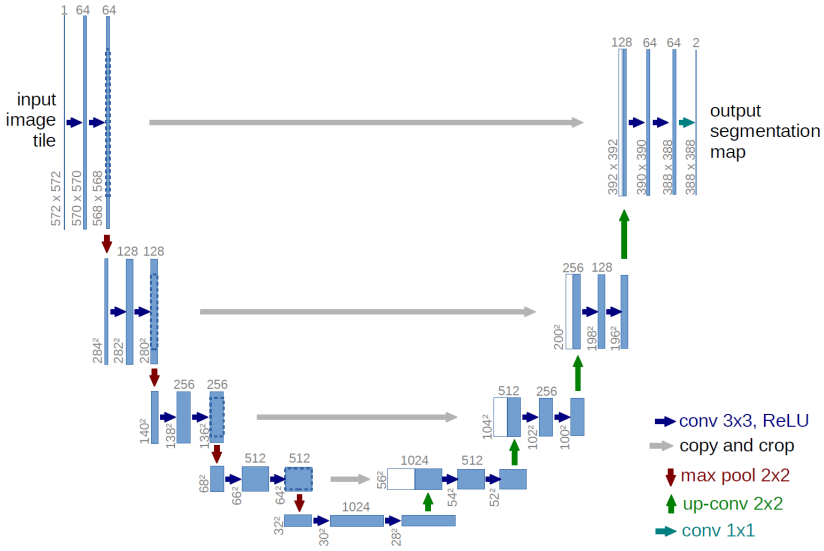

Figure 2.4: Unet architecture. [8]

1. **Contraction Path** - Each block takes an input and applies two 3×3 convolution layers followed by 2×2 max pooling.
2. **Bottleneck** - Applies two 3×3 CNN layers followed by 2×2 up-convolutional layers.
3. **Expansive Path** - Each block passes the input from the corresponding block in the contraction path to two 3×3 CNN layers followed by a 2×2 up-sampling layer.

The processing of the image in the contraction path consists of several blocks that apply max pooling with stride 2 for down-sampling, which will halve the complexity of the image and reduce it to its most prominent features. At each down-sampling step the number of feature channels is doubled until the lowest level where the bottleneck is reached. The main goal of the contraction path is to produce a a scaled down image representation that is easy for the neural net to learn.

The expansive path in the decoder is more or less symmetric to the contraction path; the number of expansion blocks is as same as the number of contraction blocks, but instead use "up-convolution" to expand the image to its original size. For each down-sampling feature in the contraction path information is lost regarding *where* features are located in favour of the *what* features that are most prominent in the image. Since this information is still present at each level we can regain it by concatenating the matrix at the right level between the corresponding blocks in each path.

Lastly, the resulting mapping passes through another 3×3 CNN layer with the number of feature maps equal to the number of output segments desired. Something that we have appreciated when working with this architecture is that it is very modular. In our implementation we have reduced it to functions for down-blocks, bottleneck and up-blocks, which potentially could be used to expand the architecture in the future.

3 Results

This section contains the results of how the CNN-architectures performed on the selected datasets. The architectures models were evaluated using various performance metrics such as *loss over epochs* and *mean intersection over union on validation data*. By checking against test data, a confusion matrix was generated, where it is possible to see which classes that are most commonly misclassified and vice versa. From the training, the epoch that generated the best accuracy was chosen as the model to evaluate.

3.1 Segnet

Segnet was trained on the Camvid dataset using the built in Adam optimizer in Keras with a learning rate of $5 \cdot 10^{-5}$. All images in the data set was normalized using the mean and standard deviation of the entire training set. Further, a decay factor of 0.5 and a decay interval of 5 was chosen, this helped with avoiding overfitting. Also, the classes in the dataset was not weighted, and the patience parameter for early stopping was set to 5 and the validation loss was monitored. This means that training will stop after 5 epochs if no decrease in validation loss can be seen. The training was set to run for a maximum of 50 epochs, but it stopped training earlier because of early stopping at epoch 31. This configuration resulted in an accuracy of 77% and a mean IoU-score of 34.5%, this was also best result for Segnet. All parameters used can be seen in table 3.1.

In figure 3.1 the learning rate decay schedule can be seen. This shows which learning rate will be used in each epoch. The figure 3.2 shows trining loss, validation loss, training accuracy and validation accuracy during training. In figure 3.3 a confusion matrix for Segnet is shown. The confusion matrix shows that Segnet classified class 0 and 3 correctly 91% and 93% of the time, respectively. Class 6 was classified as class 2 71% of the time and classes 6,7,9 and 10 was never classified correctly. Lastly, in 3.2, the precision, recall and f1-score for each class can be seen together with the overall accuracy. In figure 3.4 the resulting segmentation from Segnet is presented (middle row) together with the original image (top row) and the true segmented image (bottom row).

Parameter	Value
Dataset	CamVid
Net	Segnet
Batch size	10
CRF iterations	0
Learning rate	5e-05
Learning rate decay type	Step
Learning rate decay interval	5
Learning rate decay factor	0.5
Max epochs	50
Normalize data	Yes
Optimizer	Adam
Early stopping patience	5
Use class weighting	No

Table 3.1: Training parameters for segnet

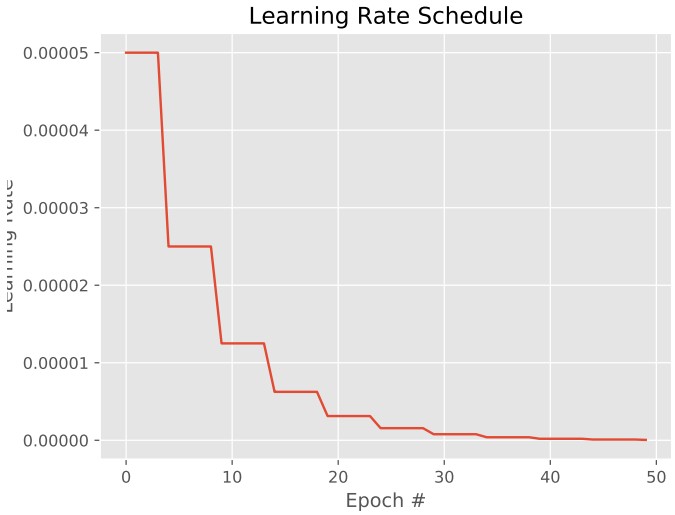

Figure 3.1: Step decay schedule used during segnet training.

Figure 3.2: Loss over epochs, training accuracy and validation accuracy for Segnet on the Camvid dataset.

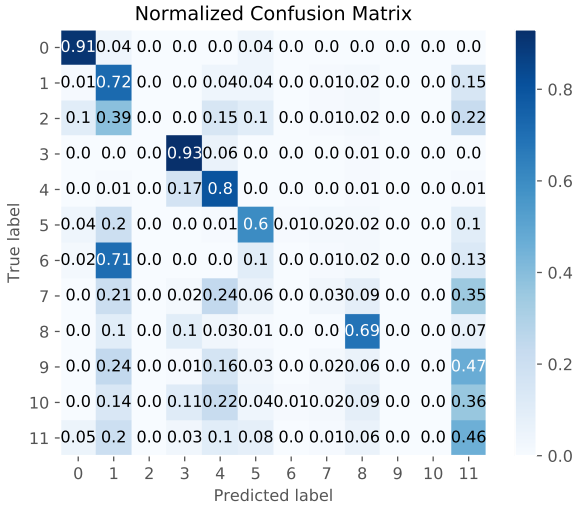

Figure 3.3: Confusion matrix for segnet.

class	precision	recall	f1-score	support
0	0.94	0.91	0.93	2012022
1	0.73	0.75	0.74	2879855
2	0.03	0.00	0.00	136961
3	0.96	0.91	0.94	2994758
4	0.68	0.90	0.77	1083948
5	0.78	0.54	0.63	1322424
6	0.03	0.00	0.00	118279
7	0.05	0.01	0.02	138654
8	0.69	0.70	0.70	463292
9	0.02	0.00	0.00	74214
10	0.01	0.00	0.00	22106
11	0.20	0.51	0.29	444495
accuracy			**0.77**	11691008
macro avg	0.43	0.44	0.42	11691008
weighted avg	0.78	0.77	0.76	11691008

Table 3.2: Precision, recall and f1-score for each class

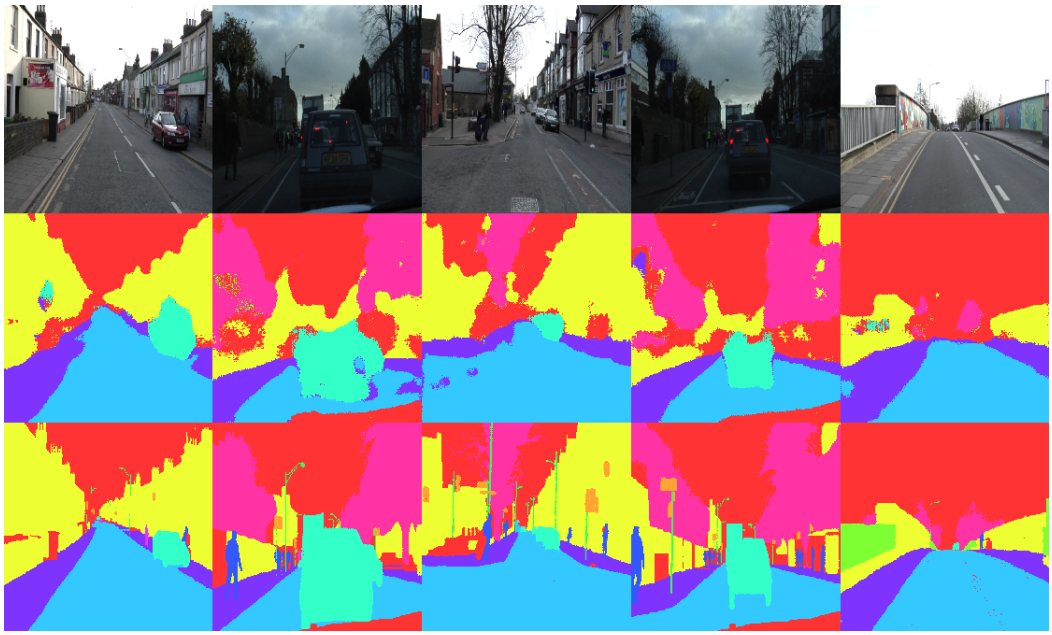

Figure 3.4: Image, prediction and ground truth for 5 images.

3.2 FCN with CRF post-processing

The fully convolutional architecture was trained on the CamVid dataset with the parameters set according to table 3.3. The data was not normalized before it was fed into the network, instead batch normalization layers where used in the model.

Parameter	Value
Dataset	CamVid
Net	FCN
Batch size	16
CRF iterations	10
Learning rate	$8 \cdot 10^{-4}$
Learning rate decay type	None
Learning rate decay interval	0
Learning rate decay factor	0
Max epochs	100
Normalize data	No
Optimizer	Adam
Early stopping patience	10
Use class weighting	Yes

Table 3.3: Training parameters for FCN + CRF

class	precision	recall	f1-score	support
0	0.93	0.93	0.93	2012022
1	0.79	0.79	0.79	2879855
2	0.13	0.14	0.13	136961
3	0.92	0.93	0.93	2994758
4	0.75	0.78	0.77	1083948
5	0.77	0.64	0.70	1322424
6	0.28	0.23	0.26	118279
7	0.26	0.07	0.11	138654
8	0.75	0.82	0.78	463292
9	0.30	0.40	0.34	74214
10	0.42	0.31	0.36	22106
11	0.30	0.42	0.35	444495
macro avg	0.55	0.54	0.54	11691008
weighted avg	0.80	0.79	0.79	11691008

Table 3.4: Class-wise Precision, recall and f1-score for FCN + CRF

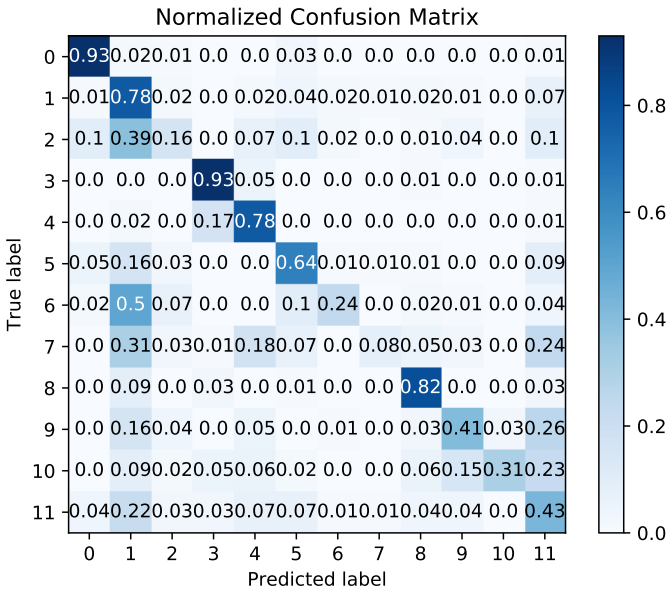

Figure 3.5: Confusion matrix for FCN with CRF post processing

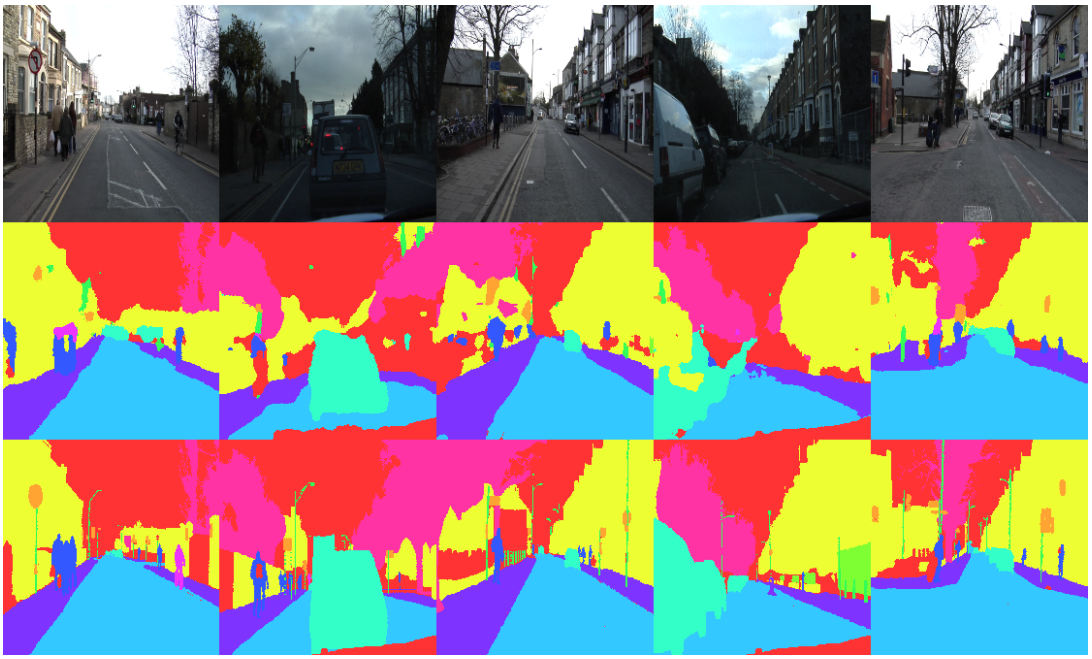

Figure 3.6: Image, prediction and ground truth for 5 images with FCN + CRF

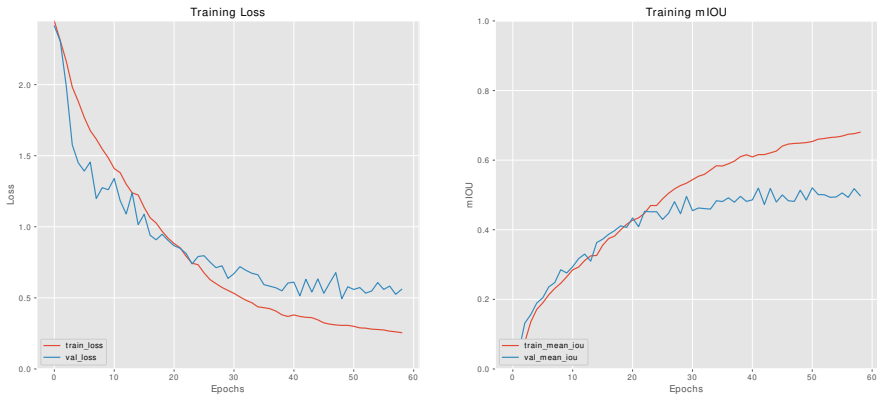

Figure 3.7: Loss over epochs, training accuracy and validation accuracy for FCN + CRF on the Camvid dataset

3.3 Unet

This section contains the results from Unet on the Camvid dataset. The experiments were performed using the *Adam* optimizer, where stochastic gradient descent method based on adaptive estimation of first-order and second-order moments is used, with the static learning rate of 0.01. The model utilizes *batch normalization* to normalize all the data that is processed from the dataset. Weighting of different classes was not considered in this case. To avoid overfitting the model an early stopping criteria on validation loss with patience of 5 was selected. This means that the model will stop training if it does not improve, i.e. get higher loss for validation data, for five epochs. The best results for Unet was reached after 32 epochs because of the early stopping criteria. In table 3.5 all parameters used in the test run can be found.

Parameter	Value
Dataset	CamVid
Net	Unet
Batch size	20
CRF iterations	0
Learning rate	1e-03
Learning rate decay type	Step
Learning rate decay interval	0
Learning rate decay factor	0
Max epochs	100
Normalize data	Yes
Optimizer	Adam
Early stopping patience	5
Use class weighting	No

Table 3.5: Training parameters for unet

The loss and mIOU on training and validation data from the Camvid dataset can be seen in figure 3.8.

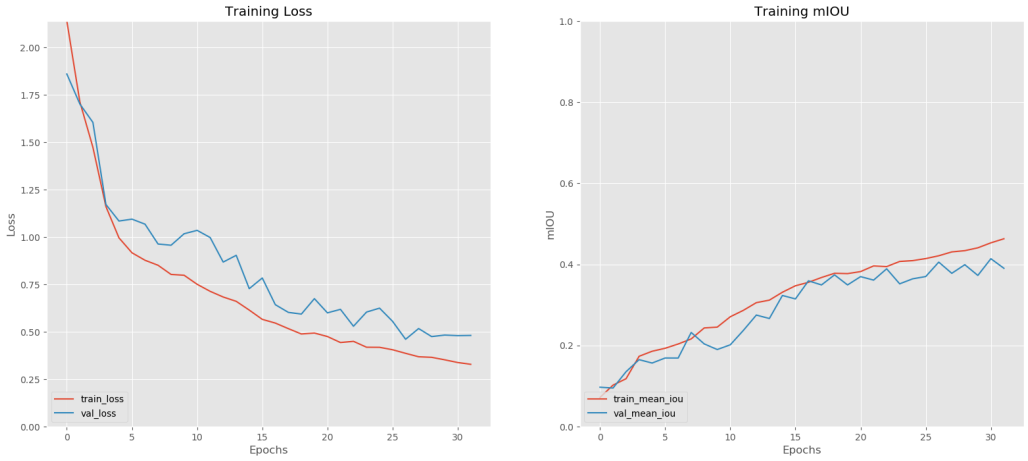

Figure 3.8: Loss over epochs, training mIOU and validation mIOU for Unet on the Camvid dataset.

The best model results in the confusion matrix seen in figure 3.9:

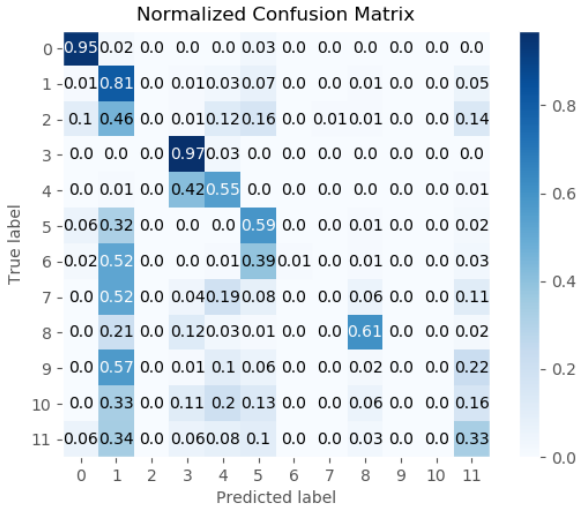

Figure 3.9: Confusion matrix for Unet on the Camvid dataset.

The confusion matrix shows that Unet classified class 0 and 3 correctly 95% and 97% of the time, it is also noteworthy that class 1 seems to be confused with other classes the most; especially class 9 (57%) and 6-7 (52%). The model also fails to classify the classes 2, 7, 9 and 10. Lastly, in table 3.6, the precision, recall and f1-score, from the confusion report, for each class can be seen together with the overall accuracy. In figure 3.10 the resulting segmentation from Unet is presented (middle row) together with the original image (top row) and the true segmented image (bottom row).

class	precision	recall	f1-score	support
0	0.92	0.95	0.94	2012022
1	0.71	0.81	0.76	2879855
2	0.24	0.00	0.00	136961
3	0.84	0.97	0.90	2994758
4	0.68	0.55	0.61	1083948
5	0.67	0.59	0.63	1322424
6	0.05	0.01	0.01	118279
7	0.03	0.00	0.00	138654
8	0.77	0.61	0.68	463292
9	0.00	0.00	0.00	74214
10	0.00	0.00	0.00	22106
11	0.37	0.33	0.35	444495
accuracy			**0.77**	11691008
macro avg	0.44	0.40	0.41	11691008
weighted avg	0.73	0.77	0.75	11691008

Table 3.6: Precision, recall and f1-score for each class for Unet.

From the classification report we got that the Unet architecture reaches an accuracy of 76.6% and 32.8 mean intersection over union on the test data.

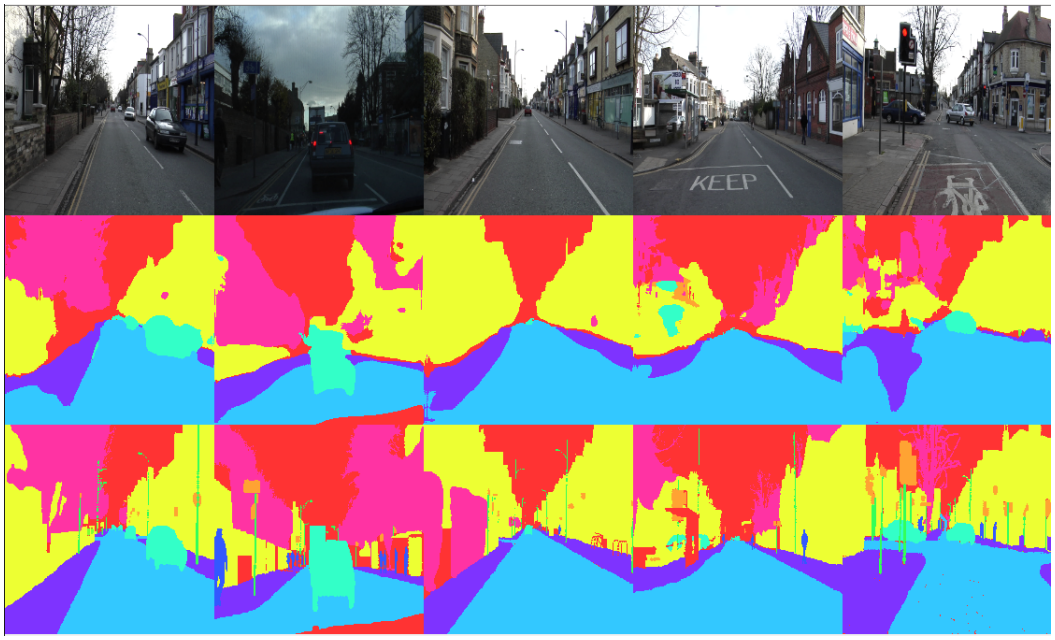

Figure 3.10: Image, prediction and ground truth for 5 images with Unet.

3.4 Evaluation

In table 3.7 the results from the best models generated from the CNN architectures on the Camvid dataset can be found.

Table 3.7: Performance metrics for CNN architectures on Camvid dataset.

	Accuracy	mIOU	1 epoch train time (s)
FCN + CRF	**79.5%**	**42.5%**	25
Segnet	77%	34.5%	18
Unet	77%	32.8%	**5**

As seen in table 3.7 FCN+CRF gets the highest Accuracy and mIOU score, while Unet is the fastest network architecture with five seconds for 1 epoch train time.

4 Discussion

This section contains the discussion of the results from Unet, Segnet and FCN+CRF in the study. Each architecture has its own evaluation for choice of model parameters, result on the Camvid dataset and comparison to findings in the previous relevant scientific literature.

4.1 Segnet

Here the results for Segnet is discussed along with the parameters chosen. The results are also compared to the ones presented in Badrinarayanan et al. which is the first published paper on Segnet [1].

4.1.1 Parameters

Badrinarayanan et al. claimed that Segnet is not sensitive to the parameters chosen during training and is thus a robust network. When training Segnet for this article, we concluded that the parameters chosen made a huge difference for overall accuracy and which classes were mostly classified. To get the best possible accuracy for Segnet, and extensive grid search for the best hyper parameters was conducted. This resulted in the parameters chosen in 3.1. To stare blindly on accuracy might have been a mistake though, as discussed in 4.1.2. Badrinarayanan et al. used Stochastic gradient descent optimizer [1], in this paper the Adam optimizer was used. Badrinarayanan et al., also used weighted classes [1], which was not used for Segnet in this paper. These changes in the parameter selection were chosen because of the improvement in accuracy in this case.

4.1.2 Results

The results presented in this article does not replicate the results presented in Badrinarayanan et al. [1]. There are many possible reasons to why that is. One is the data set chosen for this article, Camvid is a small data set and Badrinarayanan et al. only used this data set when training a smaller version of Segnet, called Segnet basic [1]. This smaller Segnet was then used as a baseline. The fact that Camvid is small increases the risk of overfitting, especially when training big networks, such as the networks in these papers. This is probably why Segnet, along with the other networks in this report, often overfitted during training and

had a hard time generalizing (the test accuracy was much smaller than training or validation accuracy).

The confusion matrix 3.3 shows that classes 0 and 3 often are classified correctly, while classes 6,7,9 and 10 never got correctly classified. This is probably because class 0 and 3 corresponds to sky and road, which are the biggest classes in the data set. Biggest in the sense that they cover most of the image. This means that Segnet will gain a high accuracy by only classifying these two classes, since they are the biggest part of the image. Class 6,7,9 and 10 are small objects in the image, such as road signs, fence, pedestrians and bikes. Signs for instance usually are located next to roads and buildings (which are also a big class). Thus the classifier misses the signs (and the other small classes) in favour of the big ones, since the small classes does not increase accuracy much anyways. To weigh the classes so that the small classes are worth more, the classifier guesses the small classes more often and more correctly (this can be seen in the result for FCN+CRF where class weighting was used 3.3). This does not however, in the case of Segnet, increase overall accuracy. But, for a program that should be used for autonomous driving, to classify pedestrians is far more important than to classify the sky. Hence, one should not stare blindly on overall accuracy since the accuracy for predicting pedestrians should be somewhat prioritized.

4.2 FCN with CRF

In this section, the results of the FCN-CRF model will be discussed.

4.2.1 Parameters

The parameter that affected the performance of the model most was the learning rate. The other important parameters for the model were: *CRF iterations, Use class weighting and Early stopping patience*. The selected learning rate gave the best mean intersection over union. The number of CRF iterations gave a better results up to ten iterations and made no noticeable difference after that. The use of class weighting gave the model the ability to classify the small classes better. The early stopping patience gave also the model a patience of ten epochs. This means that the model is trained until ten iterations and see if the validation loss improves or not.

4.2.2 Results

As seen in the confusion matrix in figure 3.5 and the class-wise performance in table 3.4, the model detected the different classes with a more even accuracy, where nine of the classes where detected with a precision and recall of at least 30%. This is mostly due to the class weighting, which helped the model with detecting smaller objects like persons and poles. As seen in figure 3.6, most of the humans were detected, but there were also some false positives.

The CRF post processing made the edge detection of the segmentation much more accurate, which can especially be seen on the horizon. The usage of CRF as post processing gave the model a better accuracy compared to when the model only used FCN.

Hao Zhou et al. [11] used Pascal VOC 2012 as their dataset compared to CamVid dataset which was used in this paper. Another main difference was that Hao Zhou et al. used back-propagation in their FCN+CRF model while in this paper we chose not to use back-propagation. For that said, the results that Hoa Zhou et al. got cannot be compared to the results from our model.

A better and bigger dataset should be used to improve the results of the FCN+CRF model in this paper. Another improvement that can be done is to implement Higher-Order CRF as post-processing. This is much better than the normal CRF.

4.3 Unet

Here the results for Unet are discussed along with the parameters chosen for the experiments in the study. The results are also compared and evaluated against results from published papers that have used a similar approach with Unet on Camvid.

4.3.1 Parameters

From the result section of Unet it is possible to see that the model has a tendency to overfit and that it is thus necessary to have a small patience window for the early stopping criteria. Since the model was initiated with default values with a static learning rate and did not use any decay on the learning rate, there were not very many parameters that needed to be taken into consideration. Each of the parameter values that were used in the experiment can be found in table 3.5. The decision to not use any class weighting have most likely affected the overall mIOU score. Choosing a static value of 0.001 for the Adam optimizer proved to be sufficient to acquire results that were not far from those obtained with parameter grid-search for Segnet and thus this might be a good choice as starting value for a stochastic gradient descent optimizer.

4.3.2 Results

The results for the Unet architecture on the Camvid dataset can be compared to several other previous studies. *Mennatullah et al.* found that Unet turned to work second-best on CamVid and got an mIOU of 53.9, but the authors note that it is slow and therefore might not be suitable for some applications [9]. As we can see here, class weighting is clearly critical to achieving a high mIOU on the dataset. These findings differ from the results from this study, where we have found that unet is the fastest architecture of the ones tested, with only 5 seconds per training epoch, and a lower parameter complexity than the other networks. A possible reason for this may be due to the authors having a different approach in the implementation of the architecture than the one used in this study. Furthermore, the fact that Unet reaches results very close to Segnet seems very much in line with the findings by *Farhangfar et al.*, where Unet is only a few percentage points off from the results that Segnet gets for semantic segmentation on UAV images [4].

The confusion matrix 3.9 shows that classes 0 (sky) and 3 (road) were classified correctly most of the time, while the classes 2 (Pole), 7 (Fence), 9 (People) and 10 (Bicycle) never got correctly classified by the model. The reason for this is most likely in part since sky and roads take up the most pixel area in the images, but also the fact that the support for classes in CamVid are very imbalanced. As we can see in table 3.6 the most commonly misclassified classes also have the lowest support in the dataset. Furthermore, we observe that classes seem to be confused with 1 (House) the most. This means that smaller objects in the image like poles, fences and bikes often get subsumed by a larger class like house in the model predictions. A possible solution to alleviate this problem could be to weight the classes to get a more accurate training.

There are many potential changes that could be made to potentially improve the results from Unet in this study. For example, as a possible extension to the architecture, you could try varying the block depth (number of layers), as well as the network depth (number of Unet blocks) to see if it results in a better model. This should be easy to do since the structure of the implemented Unet is very modular. Additionally, a more extensive study of how the choice of parameter values affect the results could be done and finally, weighing the classes in the dataset should result in a model where smaller classes get classified somewhat more correctly.

Bibliography

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[2] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. "Segmentation and Recognition Using Structure from Motion Point Clouds". In: *ECCV (1)*. 2008, pp. 44–57.

[3] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. "Semantic object classes in video: A high-definition ground truth database". In: *Pattern Recognition Letters* 30.2 (2009), pp. 88–97.

[4] Saghar Farhangfar and Mehdi Rezaeian. "Semantic Segmentation of Aerial Images using FCN-based Network". In: *2019 27th Iranian Conference on Electrical Engineering (ICEE)*. IEEE. 2019, pp. 1864–1868.

[5] John Lafferty, Andrew McCallum, and Fernando C.N. Pereira. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: (June 2001).

[6] Xiaolong Liu, Zhidong Deng, and Yuhan Yang. "Recent progress in semantic image segmentation". In: *Artificial Intelligence Review* 52.2 (2019), pp. 1089–1106.

[7] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. "Learning deconvolution network for semantic segmentation". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1520–1528.

[8] O. Ronneberger, P.Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Vol. 9351. LNCS. (available on arXiv:1505.04597 [cs.CV]). Springer, 2015, pp. 234–241. URL: http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a.

[9] Mennatullah Siam, Sara Elkerdawy, Martin Jagersand, and Senthil Yogamani. "Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, pp. 1–8.

[10] Lei Tai, Qiong Ye, and Ming Liu. “PCA-aided Fully Convolutional Networks for Semantic Segmentation of Multi-channel fMRI”. In: *CoRR* abs/1610.01732 (2016). arXiv: 1610.01732. URL: http://arxiv.org/abs/1610.01732.

[11] H. Zhou, Jun Zhang, Jun Lei, Shuohao Li, and Dan Tu. “Image semantic segmentation based on FCN-CRF model”. In: *2016 International Conference on Image, Vision and Computing (ICIVC)*. Aug. 2016, pp. 9–14. DOI: 10.1109/ICIVC.2016.7571265.