University of West Bohemia
Faculty of Applied Sciences
Department of Cybernetics

Master's thesis

Semantic Segmentation of Image Using Deep Neural Networks

Plzeň 2020 Lukáš Soukup

Místo této strany bude zadání práce.

Declaration

I hereby declare that this master's thesis is completely my own work and that I used only the cited sources.

Plzeň, 2nd July 2020

Lukáš Soukup

Abstract

The diploma thesis deals with the problem of semantic segmentation of automotive images using a deep neural networks. DeepLabV3+, the state-of-the-art model on Cityscapes dataset, was re-implemented using Keras and TensorFlow frameworks. The model was pretrained on ImageNet dataset and then using a transfer learning was transformed to perform a semantic segmentation on Cityscapes dataset. The performance of the model was verified on the validation set of Cityscapes dataset, the model achieved the performance of 73.55% IoU. In the end, the model was fine-tuned using the KPIT dataset to perform a semantic segmentation of fish eye camera automotive images. A few experiments were executed on the KPIT dataset. The best model achieved a performance of 59.26% IoU on the validation set.

Abstrakt

Tato diplomová práce se zabývá semántickou segmentací obrazu z kamery auta pomocí hlubokých neuronových sítí. Aktuálně nejlepší model aplikující sémantickou segmentaci na Cityscapes datasetu DeepLabV3+ byl kompletně re-implementován s použitím frameworků Keras a TensorFlow. Tento model byl předtrénován na ImageNet datasetu a poté byl transformován pomocí Cityscapes datasetu k tvorbě sémantické segmentace. Kvalita tohoto modelu byla ověřena pomocí validačního setu ze Cityscapes datasetu, na kterém model dosáhl výkonosti 73.55% IoU. Na závěr byl model přetrénován pomocí KPIT datasetu, aby vytvářel sémantickou segmentaci obrazu ze zadní kamery v autě, na které je čočka zvaná rybí oko. Na KPIT datasetu bylo provedeno několik experimentů. Nejlepší model dosáhl výkonosti 59.26% IoU na validační sadě.

Contents

1 Introduction

1.1 Semantic Segmentation

Semantic segmentation is one of the fundamental topics of computer vision. The goal of semantic segmentation is to assign a class label to every pixel in an image. It can be considered as image classification at the pixel level. Because the label is predicted for every pixel in the image, the task is also commonly referred as *dense prediction*. The labels could include a person, car, or road, i.e. the general pattern that can occur in the image. It is one of a few algorithms helping us to understand the context of an environment we know so far. Therefore semantic segmentation is widely used in autonomous vehicles where the context of the environment is crucial.
In order to properly understand how semantic segmentation is tackled by modern deep learning architectures, it is important to know that it is not an isolated field. Rather it is a natural step in the progression from coarse to fine inference.

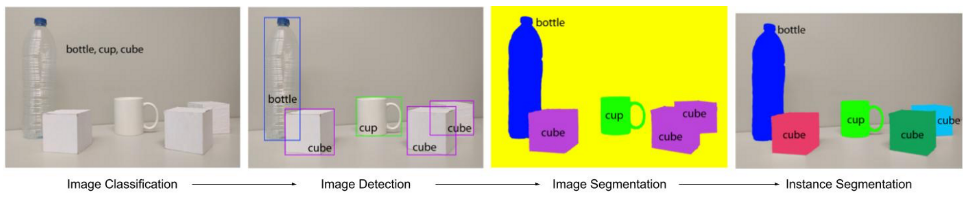


Figure 1.1: An illustration of the progression from coarse to fine inference [25].

The origin could be located at classification, which consist of making a prediction for the whole input image. The next step towards fine-grained inference is localization or detection, which is providing not only the classes but also additional information regarding the spatial location of those classes. Then there is the semantic segmentation providing a dense prediction, i.e. the classification of every single pixel, where the objective is to generate an output map of the same size as the size of the input image. Further improvements can be made, such as instance segmentation, which separate labels for different instances of the same class. In the task addressed in this thesis, instance segmentation is not necessary since it does not make any sense to separate each instance of predicted class.
Figure 1.2 shows some examples of labels from Pascal VOC 2010 dataset [12]. Each class is represented by a specific color. From segmentation map like these it is possible to evaluate a brief context. For example from upper right image

can be extracted an information that there are two people riding a bicycle on the road in front of the building and there is a car behind the fence.

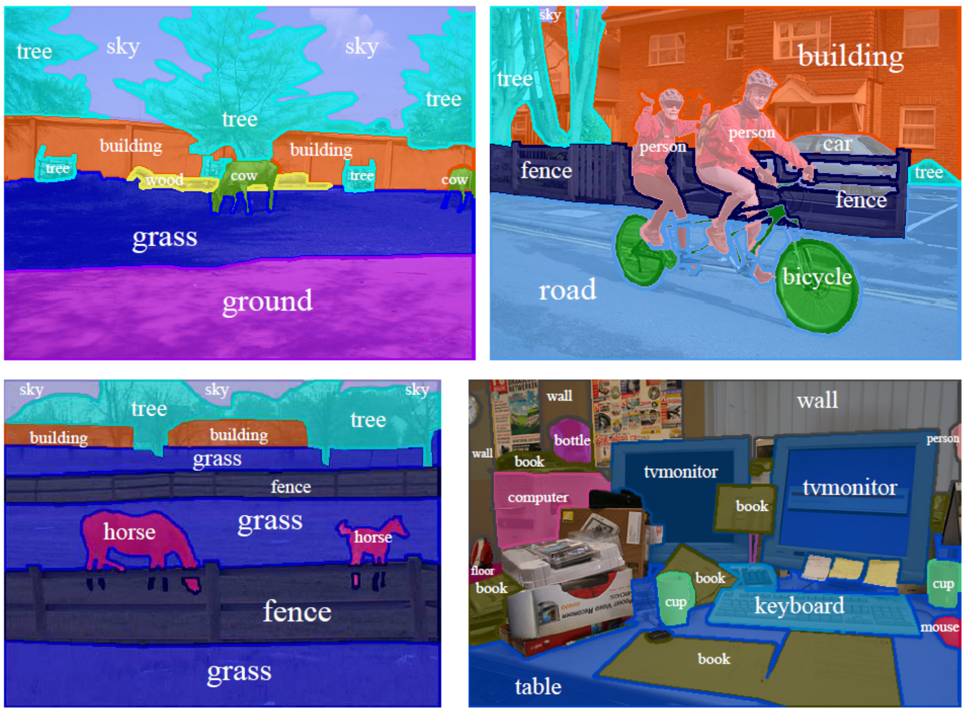


Figure 1.2: Example of semantic segmentation from Pascal VOC 2010 [12].

1.2 Motivation

As mentioned in 1.1 semantic segmentation is widely used in self-driving vehicles. For autonomous cars is crucial to have the semantic information about it's environment. The goal of this thesis is to implement state-of-the-art model for semantic segmentation and use it for processing the images from rear camera in a car. The thesis is developed in cooperation with *Digiteq Automotive s.r.o.* Digiteq is a member of *Volkswagen group* focusing on developing and testing an in-car software. The topic field *Computer Vision and Tools* works on a problem of autonomous reversing and parking using a rear camera of a car. The outcome will be the state-of-the-art semantic segmentation model which can be used to coarsely label the internal data. These data will be then used for training the algorithms which automatically detects the obstacles using the images from rear camera while reversing.
Rear camera in a car uses a fish eye lens to extend its angle of view. The distortion of the image makes image processing even more complicated. The

semantic segmentation model must be adjusted to these conditions. The intention is to train the semantic segmentation model using public datasets and then fine-tune it with internal traces from fish eye camera.

2 Related Work

Semantic segmentation with the goal to assign a label to every pixel in an image is one of the fundamental topics in computer vision. The researches have been addressing this topic since the digital images became a thing. The algorithms have been improving over the years, starting from simple image thresholding classifying pixel into two classes to a deep neural networks performing a multi-class segmentation with outstanding results. Deep convolutional neural networks based on the Fully Convolutional Neural Network show striking improvement over the systems relaying on hand-crafted features on benchmark tasks. Especially, the architectures employing a encoder-decoder structure have been proven to perform the semantic segmentation with a very good results.

2.1 Semantic Segmentation Before Deep Neural Networks

In computer vision, a simple image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels). Image segmentation is a long standing computer vision problem. However, semantic segmentation is the technique of segmenting image with understanding of image in pixel level. In other words, semantic segmentation is analysis and classification of each pixel into multiple classes.
Quiet a few algorithms have been designed to solve this non-trivial task, such as Watershed algorithm, image thresholding, K-means clustering, Conditional Random Fields, etc.

2.1.1 Image Thresholding

Image thresholding is the simplest and probably the oldest algorithm to do the segmentation of the image. This method is a process of dividing an image into two (or more) classes of pixels, i.e. *foreground* and *background*. In order to obtain thresholded image, usually, the original image is converted into a grayscale image and then the thresholding technique is applied. This method is also known as *Binarization* as the image is converted into a binary form. In detail, if the intensity value of a pixel is lesser than the threshold value, it is converted to 1 (white). If the value of a pixel is greater than the threshold value, the pixel is converted to 0 (black).

There is only one issue in the implementation of this very simple algorithm, the optimal threshold has to be determined. In a simple applications, the threshold can be set statically by the designer of the method. For the real world case, the automatic determination of the threshold is necessary. In 1979 Nobuyuki Otsu [21] came up with an idea of algorithm called Otsu's method, which became the most common method for automatic determination of the threshold. The threshold is determined by minimizing intra-class intensity variance, or equivalently, by maximizing the inter-class variance. The algorithm exhaustively searches for the threshold that minimizes the intra-class variance, defined as weighted sum of variances of the two classes:

$$\sigma_\omega^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t), \tag{2.1}$$

where weights ω_0 and ω_1 are the probabilities of two classes separated by the threshold t, and σ_0^2 and σ_1^2 are variances of these two classes. The desired threshold T corresponds to the minimum intra-class variance:

$$T = \min_t(\sigma_w^2(t)) \tag{2.2}$$

Fig. 2.1 shows an example of thresholding using Otsu's method. On the left side there is an original image in grayscale, in the middle is the binary image after thresholding, and on the right is shown the histogram of intensity with denoted threshold.

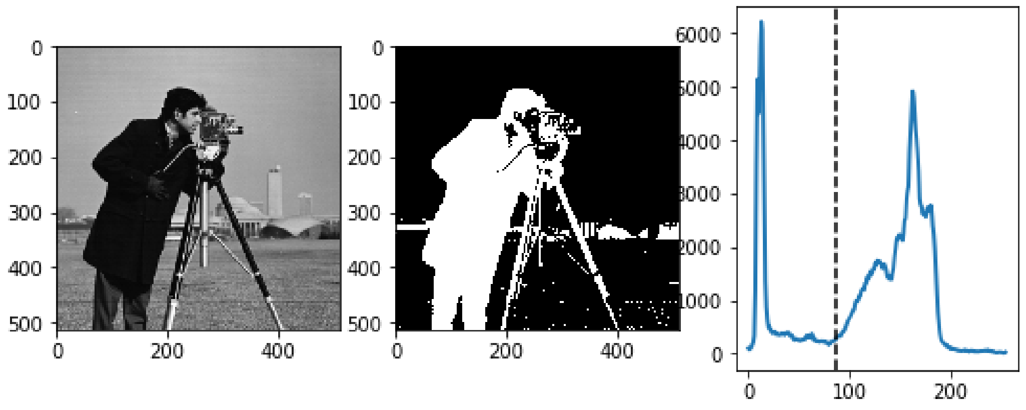


Figure 2.1: An illustration of thresholding using Otsu's method.

2.1.2 Conditional Random Fields

Random field approaches are a popular way of modeling spatial regularities in images. Their application range from low-level noise reduction to a high-level

object or category recognition and semi-automatic object segmentation. Early work focused on generative modeling using Markov Random Fields. Conditional Random Field (CRF) models have become more popular owing to their ability to directly predict the segmentation (labeling) given the observed image. Conditional Random Fields are an effective tool for a variety of different data segmentation and labeling tasks including visual scene interpretation, which seeks to partition images into their constituent semantic-level regions and assign appropriate class labels to each region. For accurate labeling it is important to capture the global context of the image as well as local information.
Jakob Verbeek and Bill Triggs [29] in their model represent the image as rectangular grid of patches at a single scale, associating a hidden class label with each patch. The CRF model incorporate 4-neighbor couplings between patch labels. The local image content of each patch is encoded using texture, color and position descriptors. For texture they compute the 128-dimensional SIFT descriptor of the patch and vector quantize it by nearest neighbor assignment against a $k_s = 1000$ word texton dictionary learned by k-means clustering of all patches in the training set. Position is encoded by overlaying the image with an $m \times m$ grid of cells ($m = 8$) and using the index of the cell in which the patch falls as its position feature. Each patch is thus encoded by three binary vectors (with k_s, k_h and $k_p = m^2$ bits), each with a single bit set correspondingly to the observed visual word. Their CRF observation functions are simple linear functions of these three vectors.
The experiments show that models based on histogram of visual words were very successful for image categorization (deciding whether or not the image whole belongs to a given category of scenes). Motivated by this, Verbeen et al. [29] in their models take the global context into account by including observation functions based on image-wide histograms of the visual words of their patches. The hope was that this would help to overcome the ambiguities that arise when patches are classified in isolation.
In the end, they defined a conditional model for patch labels that incorporates both local patch level features and global aggregate features. Let $x_i \in \{1, ..., C\}$ denote the label of patch i, y_i denote the W dimensional concatenated binary indicator vector of its three visual words ($W = k_s + h_h + k_p$), and h denote the normalized histogram of all visual words in the image, i.e. $\sum_{patches,i} y_i$ normalized to sum one. The conditional probability of the label x_i is then modeled as:

$$p(x_i = l|y_i, h) \propto exp(-\sum_{w=1}^{W}(\alpha_{wl} y_{iw} + \beta_{wl} h_w)), \tag{2.3}$$

where α_{wl}, β_{wl} are $W \times C$ matrices of coefficients to be learnt. This can be

thought of as a multiplicative combination of a local classifier based on the patch-level observation y_i and a global context or bias based on the image-wide histogram h.
The performance of the segmentation models was measured on Microsoft Research Cambridge (MSRC) dataset. The dataset consists of 240 images with partial pixel-level labels. The labels assign pixels to one of nine classes, but about a 30% of the pixels are unlabeled. For evaluation 20×20 pixel patches with centers at 10 pixel intervals were used.
To obtain a labels of patches, pixels were assigned to the nearest patch center. Patches are allowed to have any label seen among their pixels, with unlabeled pixel being allowed to have any label. To map the patch-level segmentation back to the pixel level they assign each pixel the marginal of the patch with the nearest center. The performance of 84.9% was attained by the model taking into account local and global context and not deleting unlabeled pixel (only exclude them from metrics evaluation).
In Fig. 2.2 are visualized a few segmentation results of MSRC dataset. The segmentation maps were post-processed by applying a Gaussian filter over the pixel marginals with the scale set to half of the patch spacing.

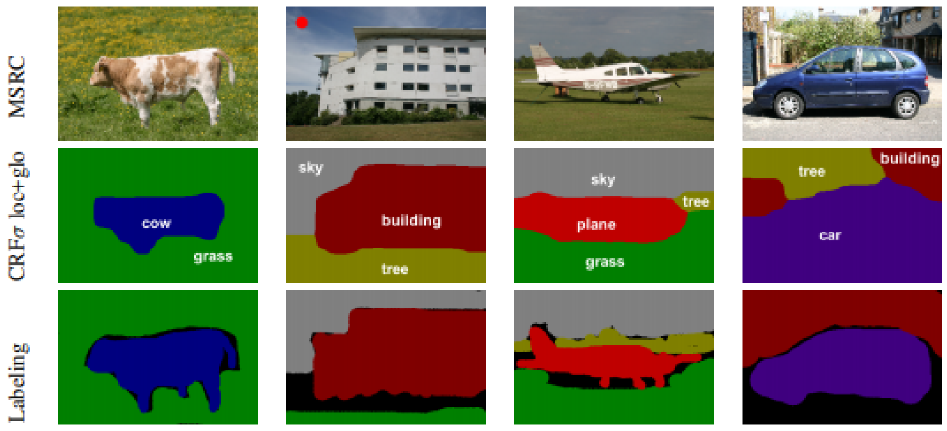


Figure 2.2: Samples from MSRC with segmentation and labeling [29].

2.2 Semantic Segmentation Using Deep Neural Networks

In spite of many traditional image processing techniques the deep learning methods has been a game changer in the field of computer vision. Neural networks set state-of-the-art in all image processing tasks including semantic segmentation. The most successful deep learning semantic segmentation architectures can be broadly thought of as an *encoder-decoder* architectures. The encoder is usually a pre-trained classification network like VGG or ResNet. The decoder part is supposed to semantically project the discriminative features from encoder onto the pixel space to get a dense classification of every pixel in the input image. One of the very early deep convolutional neural networks used for semantic segmentation is Fully Convolutional network (FCN). A variety of more advanced FCN-based approaches have been proposed to address this issue, including SegNet, U-Net, and DeepLab.

2.2.1 The History of Neural Networks

The original idea of neural network came up in half of 20th century. The first step towards neural networks took place in 1943, when Warren McCulloch and Walter Pitts [19] wrote a paper on how neurons might work. A 15 years later, a neuro-biologist Frank Rosenblatt began work on *The Perceptron* [23]. A result coming from his research was a built-in hardware and is the oldest neural network still in use. A single layer perceptron was found to be useful in classifying a continuous-valued set of inputs into one or two classes. The perceptron computes a weighted sum of the inputs, subtracts a threshold, and passes one of two possible values out as the result. In 1959, Bernard Widrow and Marcian Hoff of Stanford developed models *ADALINE* and *MADALINE* [31], which were the first neural networks applied to a real world problem - Adaptive filtering eliminating echoes in phone lines. The research on neural networks was stagnated after Minsky and Papert [20] in 1969 discovered the limits of perceptron capabilities. A key trigger for renewed interest in neural network learning was re-discovering a *backpropagation* algorithm by Paul Werbos [30] who first proposed that it could be used for neural networks. A few years later after Werbos method was re-discovered Rumelhart, Hinton and Williams [24] showed experimentally that this method can generate internal representations of incoming data in hidden layers of neural networks. The training algorithm was finalized by Yann LeCun in 1998 when *Graident-Based Learning Applied to Document Recognition* [16] was published. It is a long detailed paper on convolutional nets, graph transformer networks, and discriminative training method for sequence labeling. The real breakthrough for convolutional neural networks came up in 2012 after improving the optimized GPUs. In that year *AlexNet* by Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton [15] won *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC). The model used a convolutional neural network obtaining a Top-5 error rate of 15.3% (the next best result trailed far behind with 26.2%). Since then, convolutional neural networks (CNN) set state-of-the-art in many computer vision tasks and were not outdated so far.

2.2.2 FCN for Semantic Segmentation

Fully Convolutional Network (FCN) for Semantic Segmentation [17] was the first model that re-architects and fine-tunes classification networks to direct, dense prediction of semantic segmentation. In classification networks, conventionally, an input image is downsized and goes through convolution layers and fully connected layers, and output a predicted label. If the fully connected layers turn into 1×1 convolutional layers and the image is not downsized, the output will not be a single label. Instead, the output is a feature map with smaller resolution than the input image (due to max pooling). If the output is upsampled to the original resolution, then the pixelwise label map is obtained. In detail, ILSVRC classifiers were cast into FCNs and augmented for dense prediction with in-network upsampling and a pixelwise lost. Next, the authors built the novel skip architecture that combines coarse, semantic and local appearance information to refine the prediction. Three deep convolutional neural networks were taken into consideration. AlexNet architecture that won ILSVRC2012, VGG nets and GoogLeNet which did exceptionally well in ILSVRC2014. Each of the networks was decapitated by discarding the final classification layer, and all fully connected layers were converted to convolutions. They append a 1×1 convolution with channel dimension 21 to predict scores of each of the PASCAL classes at each of the coarse output locations, followed by a deconvolution layer to bilinearly upsample the coarse outputs to a pixel-dense output.
Fine-tuning from classification to segmentation gave a reasonable predictions for each of the networks. Even the worst model achieved $\sim 75\%$ of state-of-the-art performance. The segmentation equipped VGG16 (FCN-VGG16) already appeared to be state-of-the-art at 52.6 mean IoU performance on the test set. Training on extra data even raised the performance to 59.4 mean IoU.
While fully convolutional classifiers can be fine-tuned to segmentation with even high score on the standard metrics, their output is unsatisfactorily coarse. The 32 pixel stride at final prediction layer limits the scale of the detail in the output. The authors addressed this problem by adding a links that combine the final prediction layer with lower layers with finer strides. Combining fine layers and coarse layers lets the model make predictions that respect global structure. Adapting the net using the multi-resolution layer combinations improves the final performance of the net to 62.7% mean IoU on the VOC2011 test set and 62.2% on the VOC2012 test set setting the state-of-the-art, while

simultaneously simplifying and speeding up learning and inference [17].

2.2.3 SegNet

SegNet [1], by University of Cambridge, was originally submitted to 2015 CVPR, but it was not officially published. Instead, it was published in 2017 TPAMI. SegNet has an encoder network and a corresponding decoder network, followed by a final pixelwise classification layer as illustrated in Fig. 2.3. The encoder consist of 13 convolutional layers which correspond to 13 convolutional layers from the VGG16 network. Each of the encoder layers has a corresponding decoder layer and hence decoder network has 13 layers. The final decoder output is fed to a multi-class soft-max classifier to produce class probabilities for each pixel independently.

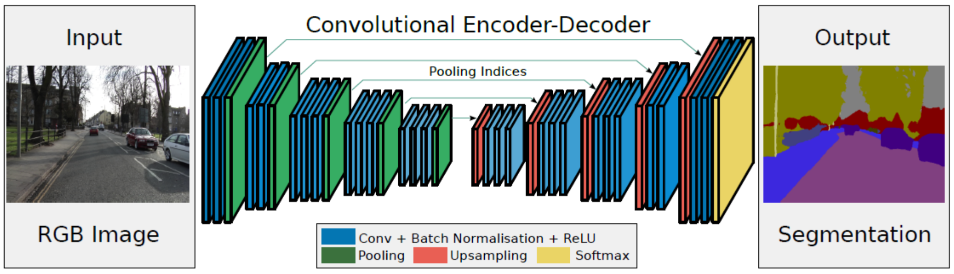


Figure 2.3: An illustration of SegNet architecture [1].

Each *encoder* in encoder network performs convolution with a filter bank to produce a set of feature maps. The feature maps are then batch normalized and an element-wise rectified-linear non-linearity ($ReLU$) is applied. Following that, max-pooling with a 2×2 window and stride 2 (non-overlapping window) is performed to sub-sample the result by a factor of 2. Max-pooling is used to achieve translation invariance over small spatial shifts in the input image. Sub-sampling results in a large input image context (spatial window) for each pixel in the feature map. While several layers of max-pooling and sub-sampling can achieve more translation invariance for robust classification correspondingly there is a loss of spatial resolution of the feature maps. The increasingly lossy (boundary detail) image representation is not beneficial for segmentation where boundary delineation is vital. The authors of SegNet decided to address this by capturing and saving boundary information in the encoder feature maps before sub-sampling is performed. Due to memory limitations the stored information was reduced to only max-pooling *indices*, i.e. the locations of the maximum feature value in each pooling window is memorized for each encoder feature map.

The appropriate *decoder* in decoder network upsamples its input feature map using memorized max-pooling indices from the corresponding encoder feature map to produce a sparse feature map. This SegNet decoding technique is illustrated in Fig. 2.4. These feature maps are then convolved with a trainable decoder filter bank to produce dense feature maps. A batch normalization step is then applied to each of these maps. The high dimensional feature representation at the output of the final decoder is fed to a trainable soft-max classifier to classify each pixel independently. The output of the classifier is a K channel image of probabilities where K is the number of classes. The predicted segmentation corresponds to the class with maximum probability at each pixel.

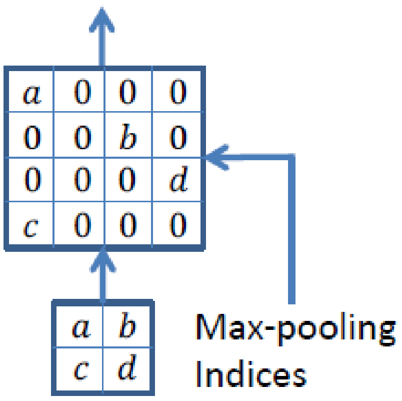


Figure 2.4: An illustration of upsampling using max-pooling indices in SegNet [1].

The performance of SegNet was quantified on two scene segmentation benchmarks. The first task is road scene segmentation (CamVid dataset [2]) which is of practical interest for various autonomous driving related problems. The second task is indoor scene segmentation (SUN RGB-D dataset [26]) which is of immediate interest to several augmented reality tasks. The model performance was benchmarked against several other well adopted deep architectures for segmentation such as FCN, DeepLab and DeconvNet. All the architectures were adopted to use the same hyper parameters and to output the same resolution output feature maps. According to the original paper results, SegNet architecture outperforms all other architectures in all the metrics (Class average, Global average, mean Intersection over Union, and Boundary F1 measure) on CamVid dataset. The experimental results on SUN RGB-D dataset shows some interesting points. All the deep architectures share very low mean IoU and boundary metrics. The global and class averages are also very small. In terms of mIoU SegNet outperformed FCN and DeconvNet but has a slightly lower mIoU than DeepLab-LargeFOV which is probably caused by the CRF post-processing used in DeepLab model. The authors predicate the overall very poor performance to a large number of classes in this segmentation task, many of which occupy a small part of the image and appear infrequently.

2.2.4 U-Net

U-Net [22] was published in a year 2015. This deep neural network architecture was designed for biomedical image segmentation. The network architecture is illustrated in Fig. 2.5. It consists of a contracting path (left side) and an expansive path (right side) which is basically encoder-decoder architecture in common notation. The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3×3 convolutions, each followed by ReLU activation and a 2×2 max pooling operation with stride 2 (no overlapping window). At each downsampling step the number of channels is doubled. Every step in expansive path consists of an upsampling of the feature map followed by a 2×2 convolution (*up-convolution*) that halves the number of channels, a concatenation with corresponding feature map from contracting path, and two 3×3 convolutions, each followed by ReLU. At the final layer a 1×1 convolution is used to map each 64-component feature vector to the desired number of classes.
The architecture of U-Net is very similar to SegNet [1] architecture. The differences are that the architectures are designed for different segmentation task. U-Net transfers the entire feature maps from encoder to decoder (grey lines in Fig. 2.5) which is then concatenated with the corresponding feature map from decoder, instead of using only max-pooling indices as it is in SegNet. Transferring the entire feature maps makes the U-Net model much larger and it needs more memory.
As for the tasks of biomedical segmentation there is very little training data available, the authors used excessive data augmentation by applying elastic deformations to the available training images. This augmentation allows the network to learn invariance to such deformations, without the need to have these transformations in the annotated image corpus. This is particularly important in biomedical segmentation, since deformation used to be the most common variation in tissue and realistic deformations can be simulated efficiently.
The experimental evaluation of the U-Net was performed on three different segmentation tasks. The first task was the segmentation of neuronal structures in electron microscopic recordings. The dataset was provided by the EM segmentation challenge what was started at ISBI 2012. The training data is a set of images from serial section transmission electron microscopy of the Drosophila first instar larva ventral nerve cord (VNC). The U-Net achieved without

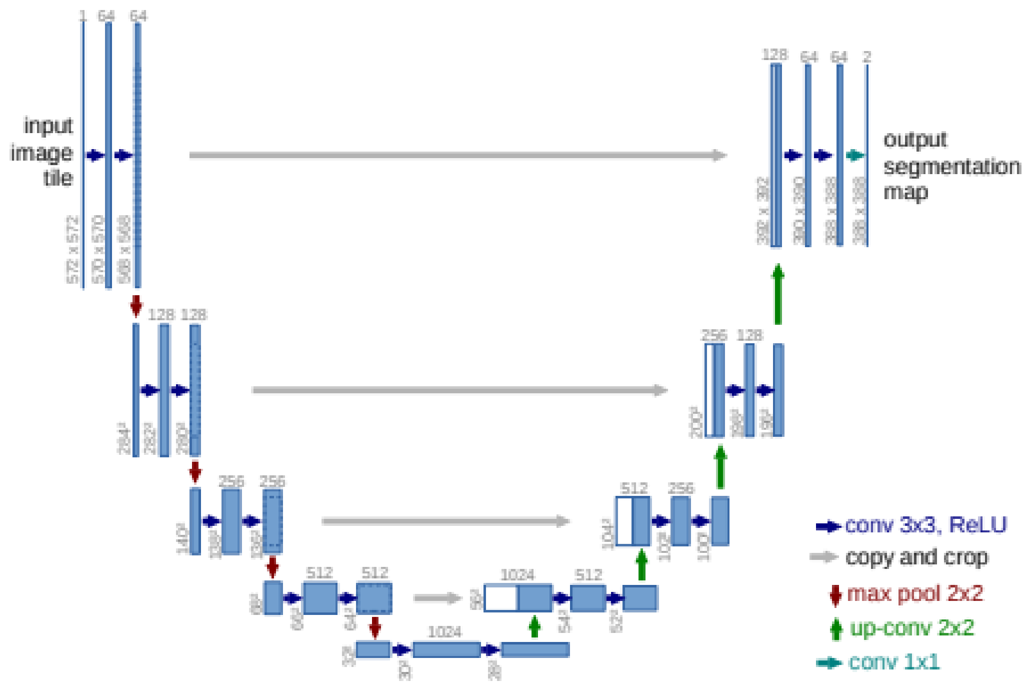


Figure 2.5: An illustration of U-Net architecture [22].

any further pre- or post-processing a warping error of 0.0003529 setting new best score. Then the U-Net was applied to a cell segmentation task in light microscopic images. This segmentation task was part of the ISBI cell tracking challenge 2014 and 2015. In this challenge U-Net achieved an average IoU of 92%, which is significantly better than the second best algorithm with 83%. The last dataset was DIC-HeLa, which are HeLa cells on flat glass recorded by differential interference contrast (DIC) microscopy. Here the architecture achieved an average IoU of 77.5% which is again significantly better than the second best algorithm with 46%.
The U-Net architecture achieved very good performance on different biomedical segmentation applications. Thanks to used data augmentation with elastic deformation, it only needs very few annotated images for learning.

2.2.5 DeepLab

DeepLab is a state-of-the-art semantic segmentation model designed and open-sourced by Google. The architecture of DeepLab was evolving over the years starting from DeepLabV1 in year 2014 to DeepLabV3+ in 2019. Each of the innovated versions brings some new ideas and significantly improves the performance.

Unlike the other architectures, instead of downsampling the feature maps by max-pooling, DeepLab uses an algorithm called *Atrous Convolution.* Atrous convolution allows to effectively enlarge the field of view of filters without increasing the number of parameters or the amount of computation. The algorithm will be explained in more detail in Chapter 3. Mathematically, atrous convolution $y[i]$ for a one-dimensional signals $x[i]$ with a filter $w[k]$ of length K and rate r is defined as:

$$y[i] = \sum_k x[i + r \cdot k]w[k] \tag{2.4}$$

Success of DeepLabV1 [3] in the task of semantic segmentation is due to some advancements added to the previous state-of-the-art models, specifically to the FCN model. The advancements address reducing feature resolution and reducing localization accuracy due to DCNNs invariation. Due to multiple pooling operations, there is a significant reduction in spatial resolution. DeepLabV1 remove the down-sampling operator from the last few layers of DCNN and instead up-sample the filters in subsequent convolutional layers resulting in feature maps being computed at a higher sampling rate. In order to capture fine details, DeepLabV1 implements a fully connected Conditional Random Field (CRF). The CRF potential incorporate smoothness terms that maximize label agreement between similar pixels. The model takes the images as input and passes through DCNN architecture with atrous convolution layers resulting in coarse map. This map is then up-sampled to the original size of the image, and fully connected CRF is applied to the map to improve segmentation results.
To improve the performance of DeepLabV1 architecture, DeepLabV2 [4] address the issue of existence of objects at multiple scales. The proposed solution is using *Atrous Spatial Pyramid Pooling (ASPP).* The idea is to apply multiple atrous convolutions with different rates to the input feature map, and fuse them together. As object of the same class can have a different scales in the image, ASPP helps consider the different scales which can improve accuracy.
The previous versions are able to encode multi-scale contextual information by probing the income features with filters at multiple rates (atrous convolution) and multiple effective fields of view (ASPP). The goal of DeepLabV3 [5] was to capture sharper object boundaries by gradually recovering the spatial information. The architecture of DeepLabV3 adopts the novel encoder-decoder architecture with atrous separable convolution. The general encoder-decoder structure have been successfully applied to many computer vision tasks, including object detection, human pose estimation, and also semantic segmentation. In addition to encoder-decoder network, DeepLabV3 also applies *depthwise separable convolution* to increase computational efficiency. The standard con-

volution is factorized into a depthwise convolution followed by a pointwise convolution. Specifically, the depthwise convolution performs a spatial convolution independently for each input channel. The pointwise convolution is employed to combine the output from the depthwise convolution and to change the number of channels.

DeepLabV3+ [6] extends version three by adding a simple yet effective decoder module to further refine the segmentation results especially along object boundaries. Compared to DeepLabV3 encoder part (backbone) uses Modified Aligned Xception [10] as its main feature extractor. All max-pooling operations are replaced by depthwise separable convolution with striding. The encoder is based on reducing the input image resolution by a factor of 16 (i.e. $output\_stride = 16$). Instead of simply upsample the features by 16, the decoder firstly upsamples the features by a factor of 4 and concatenates them with corresponding low-level features. After concatenation, a few 3×3 convolutions are applied and the feature map is upsampled by a factor of 4 to perform the prediction. The architecture of encoder-decoder structure with ASPP module is visualized at Fig. 2.6.

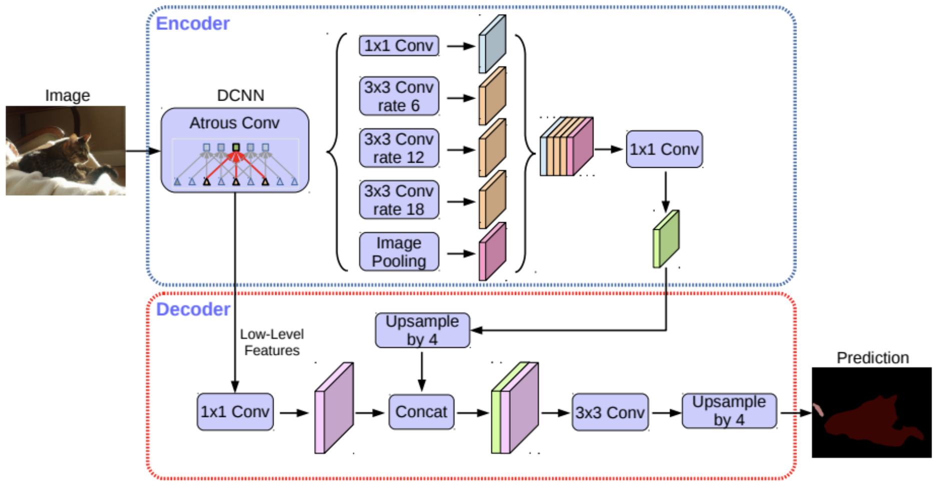


Figure 2.6: Encoder-Decoder structure with Atrous spatial pyramid pooling [6].

3 DeepLab

After a research from the chapter above it was decided that the actual state-of-the-art model on Cityscapes [9] dataset will be used. In a time of working on this thesis the best results on Cityscapes dataset were archived by DeepLabV3+ [6] model created by Google researchers. Another reasons to choose this model were open source implementation on Github and a public paper published on Arxiv.

3.1 Evolution of DeepLab

The DeepLab model was evolving over the years. In following sections it is going to be briefly described how the algorithms were improving over all four official papers written by Google Inc. authors [3–6]. The most important ideas the authors came up with were *Atrous convolution* (and its combination with another successful algorithms) and *Atrous Spatial Pyramid Pooling.*

3.1.1 DeepLab V1 + V2

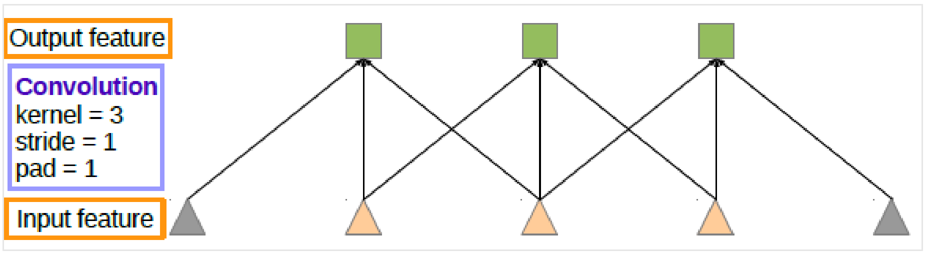
Since DeepLabV1 and DeepLabV2 architecture is very similar, they will be summed up together. Both models use *Deep Convolutional Neural Networks* with *Atrous Convolution* and *Fully Connected Conditional Random Field (CRF).* The main difference is that DeepLabV2 uses additional technology called *Atrous Spatial Pyramid Pooling (ASPP)* and implement newer deep convolutional neural network.

Atrous Convolution

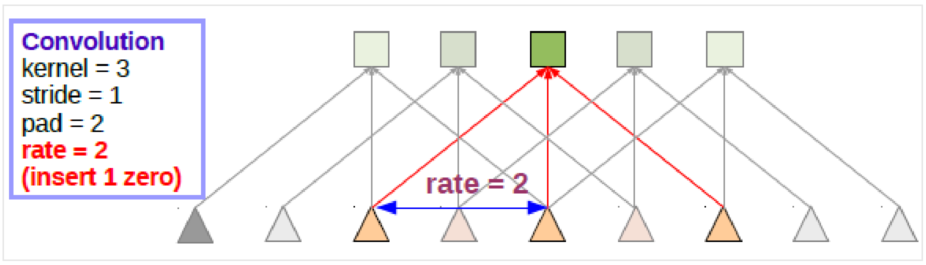
The approach illustrated at the Fig. 3.1 is known as the *hole algorithm* (*atrous algorithm*) and has been developed before for efficient computation of the undecimated wavelet transform [18]. The authors of DeepLab transferred this approach to deep convolutional neural networks and created *atrous convolution.* The implementation of this algorithm was done by modifying the standard convolution. The kernel of standard convolution is extended by inserting a zeros between trainable parameters in the kernel and therefore extending the size of the kernel. How many zeros are inserted into each "hole" is specified by new parameter called *rate.* In fact the standard convolution could be considered as a special case of atrous convolution where rate $r = 1$. The motivation to use this approach is that using atrous convolution with striding to reduce the resolution of feature maps works more efficiently and with less loose of information than using a max-pooling. Atrous convolution is computed as follows:

$$y[i] = \sum_{k} x[i + r \cdot k]w[k] \tag{3.1}$$

From the equation it is obvious that when $r = 1$, it is the standard convolution. When $r > 1$, it is the atrous convolution which is the stride to sample the input sample during convolution. Some of the papers also call this algorithm *dilated convolution*.



(a) Sparse feature extraction



(b) Dense feature extraction

Figure 3.1: The difference between (a) Standard convolution and (b) Atrous convolution [4]

Atrous Spatial Pyramid Pooling (ASPP)

Atrous spatial pyramid pooling is actually an atrous version of spatial pyramid pooling, in which the concept has been used in SPPNet [13]. In ASPP, multiple parallel atrous convolution with different rate is applied in separate branches to the input feature map, and than all output feature maps are concatenated together. This approach helps to increase accuracy by taking into account different object scales. Fig. 3.2 shows the idea of using ASPP with rates 6, 12, 18 and 24 applied to the same input feature map.

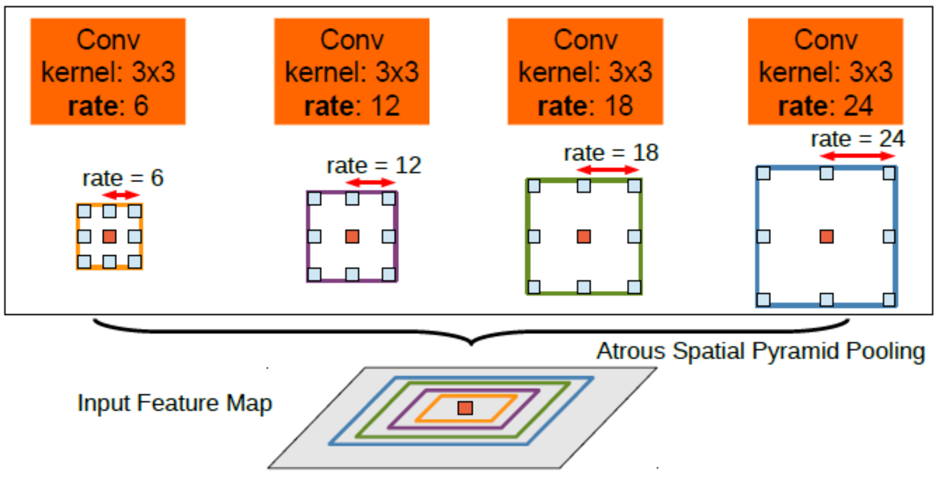


Figure 3.2: The illustration of Atrous spatial pyramid pooling [4]

Fully Connected Conditional Random Field (CRF)

A trade-off between localization accuracy and classification performance seems to be inherent in DCNNs: deeper models with multiple max-pooling layers have proven most successful in classification tasks, however the increased invariance and the large receptive fields of top-level nodes can only yield smooth responses. As illustrated in Fig. 3.3, DCNN score maps can predict the presence and rough position of objects but cannot really delineate their borders. An alternative direction based on coupling the recognition capacity of DCNNs and the fine-grained localization accuracy of fully connected CRF which were traditionally employed to smooth noisy segmentation maps was pursued in DeepLab V1 and V2 papers. As illustrated in Fig. 3.3, the score maps are typically quite smooth, therefore the integration of fully connected CRF model help to achieve better accuracy. The model employs the energy function:

$$E(x) = \sum_i \Theta_i(x_i) + \sum_{i,j} \Theta i, j(x_i, x_j), \tag{3.2}$$

where x is the label assignment for pixels, $\Theta_i(x_i) = -logP(x_i)$ is unary potential, where $P(x_i)$ is the label assignment probability at pixel i as computed by DCNN. The pairwise potential is computed as the following expression:

$$\Theta_{i,j}(x_i, x_j) = \mu(x_i, x_j)\left[\omega_1 \exp(-\frac{\|p_i - p_j\|^2}{2\sigma_\alpha^2} - \frac{\|I_i - I_j\|^2}{2\sigma_\beta^2}) + \omega_2 \exp(-\frac{\|p_i - p_j\|^2}{2\sigma_\gamma^2})\right], \tag{3.3}$$

where $\mu(x_i, x_j) = 1$ if $x_i \neq x_j$, and zero otherwise. The remaining expression uses two Gaussian kernels in different feature spaces: the first, 'bilateral' kernel depends on both pixel positions (denoted as p) and RGB color (denoted as I), and the second kernel only depends on pixel positions. The hyper parameters σ_α, σ_β and σ_γ control the scale of Gaussian kernels. Fully connected CRF as applied at the network output after bilinear interpolation.

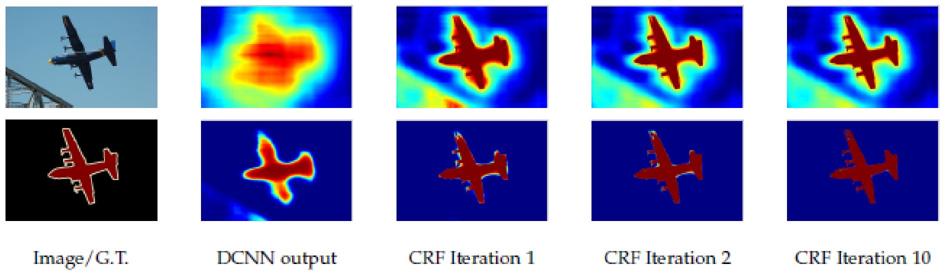


Figure 3.3: Score map (input before softmax function) and belief map (output of softmax function) for Aeroplane [4]

Architecture

The Fig. 3.4 shows the DeepLab V1 and V2 model architecture. First, the input image goes through a deep convolutional neural network such as VGG-16 or ResNet-101 with the use of atrous convolution to reduce the degree of signal downsampling (from 32x to 8x). In DeepLabV2 the ASPP module is added before the interpolation to improve the recognition and localization of objects at different scales. Then the output from a neural network is bilinearly interpolated to the original resolution and goes through the fully connected CRF to fine tune the segmentation result and improve capture of the object boundaries. DCNNs ResNet-101 and VGG-16 were both pre-trained on ImageNet [11] and than trained and tested on PASCAL VOC-2012 [12]. In the time of publishing both models DeepLabV1 resp. DeeplabV2 set the new state-of-the-art at the PASCAL VOC-2012 semantic image segmentation task, reaching 71.6 % resp. 79.7 % IoU accuracy in the test set.
However, CRF is a post-processing process which makes DeepLabV1 and DeepLabV2 become not an end-to-end learning framework. Therefore, it is not used in DeepLabV3 and DeepLabV3+ anymore.

3.1.2 DeepLab V3

The authors of the paper tried to rethink the DeepLab architecture and came up with a more enhanced DeepLabV3. The proposed model does not use a Conditional Random Fields post-processing anymore and yet outperforms DeepLabV1 and DeepLabV2.

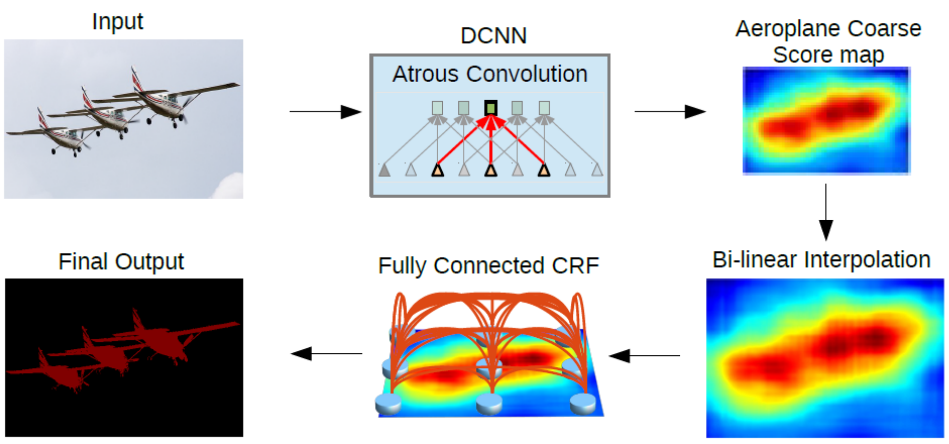


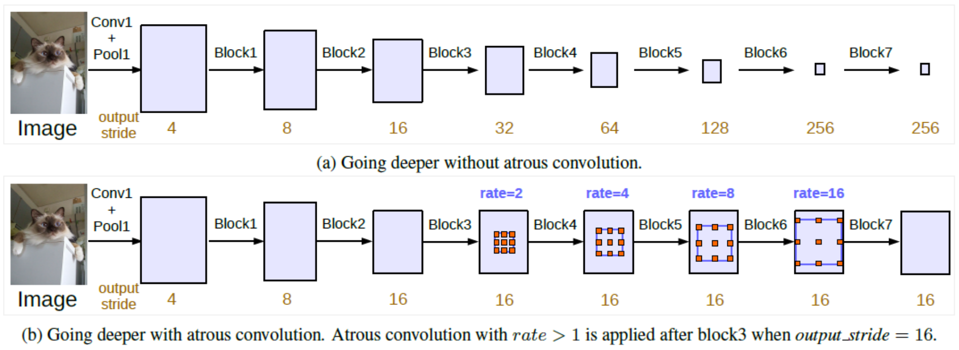
Figure 3.4: The illustration of architecture of the models

Going Deeper with Atrous Convolution

DeepLabV3 designed modules with atrous convolution laid out in cascade. There are duplicated several copies of the last ResNet block, denoted as block4 in Fig. 3.5. Each of these blocks consists of three 3×3 convolution, and the last one contains stride 2 except for the last block (this is similar to original ResNet). But it was discovered that consecutive striding is harmful for semantic segmentation since detail information is decimated. Thus, atrous convolution with rates determined by desired *output_stride* is applied, as shown in Fig. 3.5 where $output\_stride = 16$. In proposed model the authors experiment with cascaded ResNet blocks up to Block7, which has $output\_stride = 256$ if no atrous convolution is applied. Motivated by multi-grid methods a different atrous rates withing Block4 to Block7 are proposed in the model. In particular, $Multi\_grid = (r_1, r_2, r_3)$ is defined as the unit rates for the three convolutional layers within these blocks. The final atrous rate for convolutional layer is equal to the multiplication of the unit rate and the corresponding rate. For example, when $output\_stride = 16$ and $Multi\_grid = (1,\ 2,\ 4)$, the three convolutions will have $rates = 2 \cdot (1,\ 2,\ 4) = (2,\ 4,\ 8)$ in the Block4 respectively [5].

Atrous Spatial Pyramid Pooling

The ASPP from DeepLabV2 [4], where four parallel atrous convolutions with different atrous rates are applied on the top of feature map, is revisited with proposing some improvements. Newly a batch normalization is included within ASPP. The authors discovered that as sampling rate becomes larger the number of valid filter weights become smaller. In the extreme case where the rate value is close to the feature map size, the 3×3 filter, instead of capturing the whole



Conv1
+
Pool1
Image
output stride
Block1
Block2
Block3
Block4
Block5
Block6
Block7
4
8
16
32
64
128
256
256
(a) Going deeper without atrous convolution.
rate=2
rate=4
rate=8
rate=16
4
8
16
16
16
16
16
16
(b) Going deeper with atrous convolution. Atrous convolution with rate > 1 is applied after block3 when output_stride = 16.

Figure 3.5: The comparison of a cascaded modules and their *output_stride* (a) without and (b) with atrous convolution

image context, degenerates to a simple 1×1 filter since only the center filter weight is effective. To overcome this problem and incorporate global context information to the model, the following was adopted. Global average pooling in one of the last feature maps of the model, feed the resulting image-level features to a 1×1 convolution with 256 filters (and batch normalization), and then bilinearly upsample the feature to the desired spatial dimension. In the end, the improved ASPP consists of one 1×1 convolution and three 3×3 convolutions with $rates = (6, 12, 18)$ when $output\_stride = 16$ and the image-level features, as shown in Fig. 3.6. The results of all the branches are then concatenated and pass through another 1×1 convolution (also with 256 filters and batch normalization) before the final 1×1 convolution which generates the final logits. Note that the rates are doubled when $output\_stride = 8$.

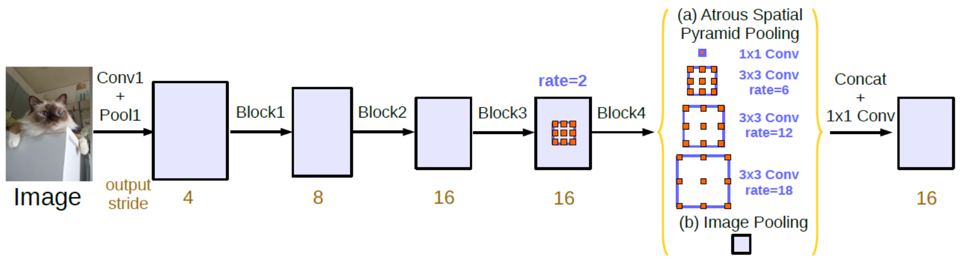


Figure 3.6: Architecture of DCNN parallel modules with atrous convolution and ASPP.

Experimental Evaluation

The ImageNet pretrained ResNet is adapted to the semantic segmentation by applying atrous convolution to extract dense features. The proposed models were evaluated on the PASCAL VOC 2012 semantic segmentation benchmark which contains 20 foreground object classes and one background class. The performance is measured in terms of pixel *Intersection over Union (IoU)* averaged across the 21 classes. As a DCNN ResNet-50 and ResNet-101 adopted with atrous convolution and ASPP were used for evaluation. By using deeper network, multi-grid method, different *output_strides*, pretraining on MS-COCO and applying a bootstrap method, DeepLabV3 model achieves the performance of 85.7% on the test set. Finally, model pretrained both on ImageNet [11] and JFT-300M dataset [27] results in a performance of 86.9% on PASCAL VOC 2012 test set.

3.1.3 DeepLab V3+

DeepLabV3+ is the latest version of DeepLab models. This model extends DeepLabV3 by a decoder module to refine the segmentation results. The model consists of a DCNN backbone, ASPP and the decoder.

Backbone

DeepLab V3+ model uses the classical encoder-decoder architecture. In the paper the encoder is also called backbone. The backbone is supposed to extract features from a high resolution image and transform it to the lower resolution feature vector. The original paper and original implementation provides two different backbones ResNet-101 [14] and Modified aligned Xception [8]. As already described in section 3.1.2, for DeepLab models the key parameter of the deep convolutional neural network is the ratio of input image to the final output resolution denoted as *output_stride*. Same as in DeepLabV3 model, the *output_stride = 16 (or 8)* is attained by implementing an atrous convolution and removing striding from last one (or two) block(s). Inspired by the success of ideas from Xception model [8], the depthwise separable convolution is adopted into the backbone together with atrous convolution.

Atrous Spatial Pyramid Pooling with Depthwise Separable Convolution

This model improves algorithms from DeepLabV3 [5] by using successful ideas from other papers. The authors combined together atrous convolution with depthwise separable convolution to create *atrous separable convolution*. And then they also implemented atrous separable convolution into atrous spatial pyramid pooling.

Atrous Convolution Atrous convolution is a powerful tool that allows to explicitly control the resolution of features computed by deep convolutional neural networks and adjust filter's field-of-view in order to capture multi-scale information, generalizes standard convolution operation. In the case of two-dimensional signals, for each location i on the output feature map y and a convolution filter w, atrous convolution is applied over the input feature map x as follows:

$$y[i] = \sum_k x[i + r \cdot k]w[k], \tag{3.4}$$

where the atrous rate r determines the stride with which input signal was sampled. Note that standard convolution is a special case in which rate r = 1. The filter's field-of-view is adaptively modified by changing the rate value.

Depthwise Separable Convolution Depthwise separable convolution (or group convolution) is a powerful operation to reduce the computation cost and number of parameters while sometimes even increasing performance. It is factorizing a standard convolution into a depthwise convolution followed by a pointwise convolution. This significantly reduces the computation complexity. The reason is that the depthwise convolution is performed independently for each input channel. After that the pointwise convolution actually combines the output from the depthwise convolution.

Atrous Separable Convolution Depthwise separable convolution decomposes a standard convolution into a depthwise convolution (applying a single filter for each input channel) and pointwise convolution (combining the outputs from depthwise convolution across channels). In DeepLabV3+ model the atrous separable convolution was introduced where atrous convolution is adopted in the depthwise convolution as shown in Fig. 3.7.

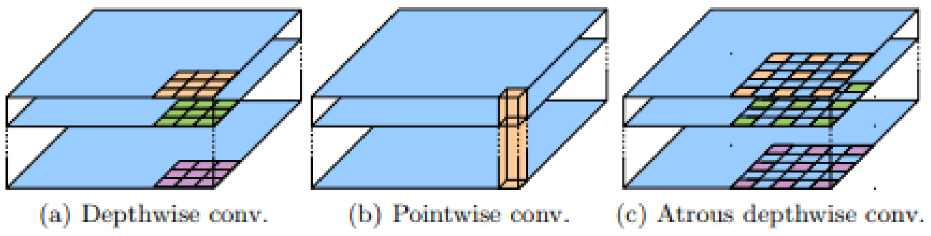


Figure 3.7: (a) Depthwise convolution, (b) a pointwise convolution and (c) atrous separable convolution (shown with *rate* = 2) [6]

Atrous Spatial Pyramid Pooling DeepLabV3 [5] employs atrous convolution to extract the features computed by deep convolutional neural networks at an arbitrary resolution. Here, the *output stride* is denoted as the ratio of input image spatial resolution to the final output resolution (before global pooling or fully connected layer). For the task of image classification, the spatial resolution of the final feature maps is usually 32 times smaller than the input image resolution and thus output stride = 32. For the task of semantic segmentation, it is better to adopt output stride = 16 (or 8) for denser feature extraction. Additionally, DeepLabV3+ augments the Atrous Spatial Pyramid Pooling module, which probes convolutional features at multiple scales by applying atrous convolution with different rates, with the image-level features. The last feature map before logits is used in the original DeepLabV3+ as the encoder output in proposed encoder-decoder structure. Note the encoder output feature map contains 256 channels and rich semantic information [6]. Fig. 3.8 shows the proposed encoder-decoder structure applying atrous spatial pyramid pooling in the encoder to extracts features from the backbone at a multiple scales.

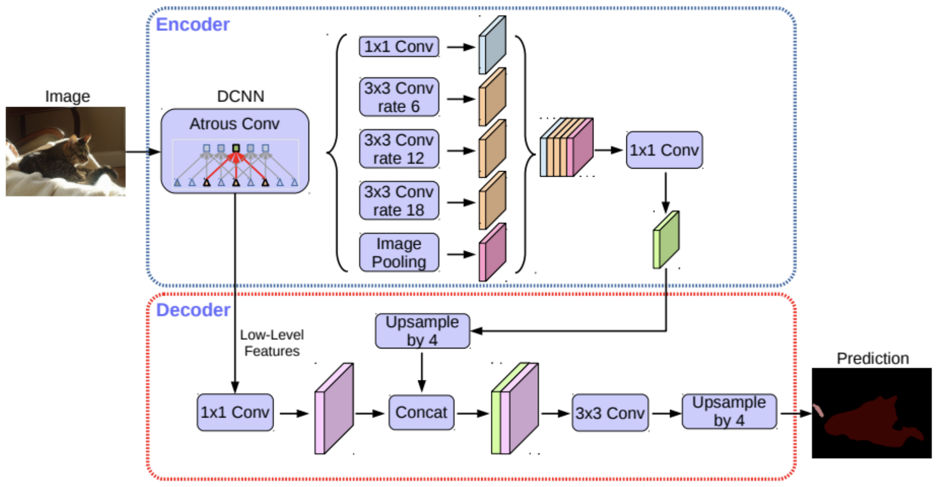


Figure 3.8: Encoder-Decoder structure with Atrous spatial pyramid pooling [6]

Decoder

In the original paper a simple yet effective decoder is proposed as illustrated in Fig. 3.8. The encoder features are first bilinearly upsampled by a factor of 4 and then concatenated with the corresponding low-level features from the network backbone that have the same spatial resolution (third Xception block before striding). Then another 1×1 convolution is applied on the low-level features to reduce the number of channels, since the corresponding low-level features usually contain a large number of channels (e.g., 256 or 512) which may outweigh the importance of the rich encoder features and make the training harder. After the concatenation a 3×3 convolution is applied to refine the features followed by another simple bilinear upsampling by a factor of 4. It was proven that using *output_stride = 16* for the encoder module strikes the best trade-off between speed and accuracy. The performance is marginally improved when using output stride = 8 for the encoder module at the cost of extra computation complexity [6].

3.2 Re-Implementation of DeepLabV3+

Even tough there is an open source original implementation of DeepLab V3+ in TensorFlow published on Github, it was decided to make complete re-implementation of the model. First reason to do that was that the official implementation done by Google is not very well documented and poorly readable. Another reason was that the original repository does not provide the whole implementation of the architecture of the layers in the model. The representation of the model is made only by a frozen graph which means no change to the architecture of the model can be made. And last but not least is that the model will be used as out-of-the-box solution, therefore the complete control over the model and understanding is necessary. Due to these reasons it was decided to create a custom re-implementation. Programming language *Python* and the machine learning framework *Keras* with *TensorFlow* backend was chosen to be used for the re-implementation.

3.2.1 Keras + TensorFlow

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. This deep learning library allows easy and fast prototyping, supports convolutional neural networks and runs seamlessly on CPU and GPU [7].

3.2.2 Backbone

The original implementation offer to use 2 different backbones: ResNet-101 and Xception. Due to time limitation it was chosen to re-implement only one backbone. According to the official results from the original paper and Cityscapes leader board, the model with Xception backbone reached slightly

better performance, therefore it was chosen to re-implement model only with Xception backbone.
Xception backbone architecture was implemented same as in the original implementation - Modified Aligned Xception. This architecture was created by MSRA team [10] by modifying original Xception model by:

- Adding more layers.
- Max pooling operations replaced by depthwise separable convolution with striding.
- Adding extra batch normalization and ReLU activation after each 3×3 depthwise convolution.

Depthwise Separable Convolution

Depthwise separable convolution was implemented as shown in Fig. 3.9. First there is a depthwise convolution applied to each channel separately followed by batch normalization and ReLU activation. Then the dependency between channels is captured by a pointwise convolution (again follow by a batch normalization and ReLU activation). Applying a pointwise convolution also allows to change the dimension.

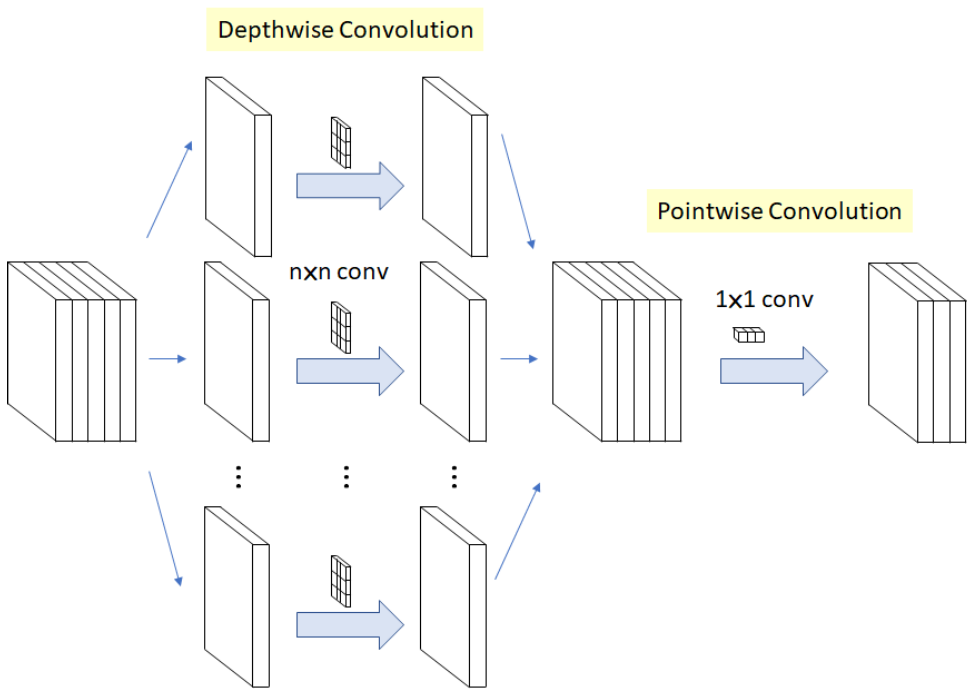


Figure 3.9: Depthwise separable convolution. In concrete, here $n = 3$. [8]

Compared with conventional convolution, a 3×3 depthwise separable convolution is not performed across all channels which means, that the number of connections is lower making the model lighter. Less connections also results in lower number of trainable parameters which reduces the computation complexity and makes a training easier.

Xception Architecture

The actual architecture is shown at Fig. 3.10. The modifications of the original Xception are denoted by orange color. In short, the Xception architecture is a linear stack of blocks with depthwise separable convolution layer and residual connections organized into 3 blocks - Entry Flow, Middle flow and Exit flow. Each of the blocks consists of three 3×3 depthwise separable convolutions and residual connection. In Entry flow and in the first block of Exit flow the last convolution layer is with striding to reduce the resolution of output feature map by a factor of 2. In case of blocks with striding, to achieve same resolution of output from residual connection and separable convolutions, the residuum goes through a pointwise convolution with striding. At the end of the block, the results from separable convolutions and residual connection are then merged together by a *Addiction* layer. At this setting the $output\_stride = 16$, but according to the experimental evaluations from the original paper the model attains slightly better performance when during evaluation $output\_stride = 8$. Therefore in the implementation there is a possibility to set stride = 1 in the third block in Entry flow.

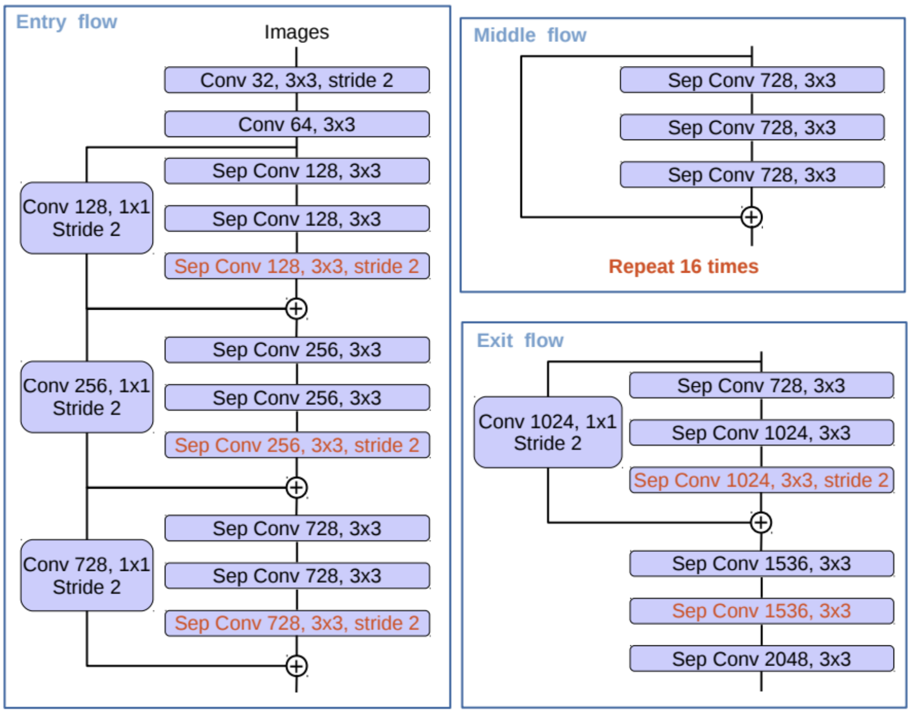


Figure 3.10: Visualization of architecture of Modified Aligned Xception [10]

3.2.3 Atrous Spatial Pyramid Pooling

In addition, DeepLabV3+ model augments Atrous Spatial Pyramid Pooling used in DeepLabV3 [5] module, which probes convolutional features at multiple scales by applying atrous convolution with different rates. The last feature map before classification layer from backbone is used. In concrete, input feature map has a resolution 16 (or 8) times smaller than the input image and has 2048 channels when Xception backbone is used. To this feature in ASPP module is applied:

- 1×1 pointwise convolution,
- 3×3 atrous convolution with rate 6 (or 12),
- 3×3 atrous convolution with rate 12 (or 24),
- 3×3 atrous convolution with rate 18 (or 36),
- Global Average Pooling.

Where all convolution layers are with padding *valid* and followed by a batch normalization and ReLU activation. Global Average Pooling is applied at each channel separately and then the output is upsampled to the same resolution as output feature maps from convolutions. Note that each of the layers applied in ASPP has 256 channels. These output feature maps are therefore concatenated by *Concatenate* layer and transformed by 1×1 pointwise convolution with 256 filters, batch normalization and ReLU activation. As a difference from the original implementation on the top of this pointwise convolution layer is added a *Dropout* with rate $= 0.1$ to help the generalization of ASPP module and to reduce overfitting. The output feature map from ASPP module is bilinearly upsampled by a factor of 4 and is one of the inputs of decoder module.

3.2.4 Decoder

The decoder part of the architecture is supposed to decode the encoded low level features and perform a classification. In the original paper the authors proposed two encoders and compared the results of model with each of them. The first one could be considered as a naive decoder design. This decoder only bilinearly upsamples the last feature map by *output_stride*. This version with naive decoder is actually very similar to DeepLabV3 [5] architecture with only difference of upsampling a feature map and then classifying unlike during training of DeepLabV3 the ground truths were downsampled. The approach with upsampling feature maps instead of downsampling ground truth attained the performance of 77.21% on PASCAL VOC 2012 [12] *val* set.
To improve over this naive baseline they employed a decoder module at top of encoder output as shown in Fig. 3.8. In the decoder module, they considered three places for different design choices, namely (1) the 1×1 convolution used to reduce channels of the low-level feature map from encoder module, (2) the 3×3 convolution used to obtain sharper segmentation results, and (3) which encoder low-level features should be used [6].
In the re-implementation it was decided to implement the second proposed decoder which is still simple yet more effective and significantly improves the performance. The inputs to the decoder module are (1) a low-level feature from encoder (2) and the output from ASPP module. The low-level feature from encoder must be one of the feature maps from the encoder block with corresponding resolution to the output from ASPP. In concrete, in Xception backbone it is block2 from Entry flow, because the resolution of input feature maps to convolution layers in this block are 4 times smaller than the resolution of the input image. Which of these feature map to use is a designer choice. For the baseline model in the re-implementation the input to third convolution layer is used. As shown in Fig. 3.8 the skipped low-level feature first goes through a 1×1 feature map to reduce a number of channels to 48 followed by a batch

normalization and ReLU activation. Then the feature map is concatenated with the output from ASPP module. After that, a 3×3 convolution is applied to refine the segmentation results. On the top of the module a classification layer is added. Classification is done by a 1×1 convolution layer with channels equal to the number of classes in the dataset. In the end, the results are upsampled by a factor of 4 to obtain the feature map of the same resolution as the input image.
Since the DeepLabV3+ model is fully convolutional (even a classification layer), any resolution of the input image can be segmented because no layer is resolution dependent.

3.3 Fine-tuning and Transfer Learning

Fine-tuning is a deep learning method in which a developed model for some task is reused as a starting point model on a similar task. It is a very popular method which enable having a well performing models even with a little training data. In practice the usual process is to fine-tune existing network trained on a large dataset like ImageNet by continue training on a smaller dataset relevant to the given task. Provided that the dataset is not drastically different in context to the original dataset (e.g. ImageNet), the pre-trained model should have learned the features that are relevant to given classification problem.
Transfer learning is a machine learning method where a model developed for one task is reused as a starting point for a model on a second task. It is a popular approach in deep learning where pretrained models are used as the starting point saving the time and resources required to develop neural network models on related problem. As a transfer learning could be also considered a part of creating DeepLab architecture where classification network (Xception backbone) pretrained on ImageNet to perform a classification was reused as a starting point for the semantic segmentation task.
In practice, both fine-tuning and transfer learning of deep learning architectures is usually done by truncating the last classification layer (usually softmax layer) of the pretrained network and replace it with the new classification layer that is relevant to the new problem. For example, pre-trained network on ImageNet comes with a softmax layer with 1000 classes. The task of semantic segmentation on Cityscapes dataset requires the classification into 19 classes, the new classification softmax layer of the network will have 19 channels instead of 1000.

4 Evaluation

The DeepLab V3+ model was trained end-to-end without piecewise pretraining of each component. The model was first pretrained on ImageNet [11] to extract dense feature maps by atrous convolution. Then the model was fine-tuned for semantic segmentation using the Cityscapes dataset [9]. The evaluation was performed on the validation set of Cityscapes dataset.
The same training protocol as in original paper [6] was followed, i.e. the same learning rate schedule, crop size 513×513, fine-tuning batch normalization parameters, and using the augmentations during the training.

4.1 Datasets for Semantic Segmentation

For the task of semantic segmentation are overall being used algorithms of supervised learning. The key ingredient for supervised learning models is to have access to enough labeled data for the training. Luckily for the researchers there were created some large public datasets available free for the non-commercial use. There are many computer vision datasets created for different computer vision tasks such as image classification, object detection and semantic segmentation.

4.1.1 ImageNet

The digital era has brought with it an enormous explosion of data. There are a uncountable numbers of photos on Flickr, videos on YouTube and even larger number for images in the Google Image Search database. By exploiting these data, more sophisticated and robust algorithms could be trained, resulting in better applications for users. Jia Deng et al [11] introduced a database called *ImageNet*, a large-scale ontology of images, which is solving the problem of organization of such huge amount of image data.
ImageNet [11] is ongoing research effort to provide researchers an easily accessible image database. The image database is organized according to the WordNet hierarchy. Each meaningful concept in WordNet, possibly described by multiple words or word phrases, is called a *synset*. There are around 80 000 noun synsets in WordNet. The ImageNet database aim to to provide on average $500 - 1000$ images to illustrate each synset. Images of each concept are quality-controlled and human-annotated. In its completion, the database will

offer tens of millions of cleanly sorted images for the most of the concepts in the WordNet hierarchy.
Hierarchy ImageNet organizes the different classes of images in a densely populated semantic hierarchy. The main asset of WordNet lies in its semantic structure, i.e. ontology concepts. Similarly to WordNet, sysnets of images in ImageNet are interlinked by several types of relations, Although one can map any dataset with category labels into a semantic hierarchy using WordNet, the density of ImageNet is unmatched by others. For example, it is very unlikely that any computer vision dataset offers images of 147 dog categories.
Accuracy The goal is to provide a clean dataset at all levels of the WordNet hierarchy. But achieving a high precision for all depth of the ImageNet tree is very challenging because the lower in the hierarchy a synset is, the harder is to classify, e.g. Siamese cat versus Burmese cat.
Diversity ImageNet constructed with the goal that objects in images should have variable appearances, positions, view points, poses as well as background clutter and occlusions. In an attempt to tackle the difficult problem of quantifying image diversity, average image of each synset is computed and lossless JPG file size is measured which reflects the amount of information in an image. The idea is that a synset containing diverse images will result in a blurrier average image, the extreme being a gray image, whereas a synset with a little diversity will result in a more structured, sharper average image. Therefore, a smaller JPG file size is expected to be smaller of the average image of a more diverse synset.
The first construction of ImageNet database involved collecting candidate images for each synset. The average accuracy of image search results from the internet was around 10%. A large set of candidate images was collected (over 10k images for each synset). To collect a highly accurate dataset, cleaning of the data must have been applied. They relied on humans to verify each candidate image collected in previous step for a given synset. This was achieved by using Amazon Mechanical Turk (AMT), an online platform where users are a set of candidate images and the definition of the target synset and asked to verify whether each image contains object of the synset.
The creation of ImageNet database was crucial for the growth of computer vision, especially for the growth of deep learning. ImageNet became the central resource for a broad of range of vision related research. In fact, almost every deep convolutional neural networks used for any computer vision task (image classification, detection, segmentation, etc.) is first pretrained on ImageNet.

4.1.2 Cityscapes

The resurrection of deep learning had a major impact on the current state-of-the-art in machine learning and computer vision. A major contributing factor for the deep neural networks success is the availability of large-scale dataset. Therefore, in a year 2016 a team from *Daimler AG R&D*, *Max Planck Institute for Informatics* and *TU Darmstadt Visual Inference Group* decided to take another step in visual understanding of complex urban street and created Cityscapes Dataset [9]. Despite the existing gap to human performance, scene understanding approaches have started to become essential components of advanced real-world systems. A particularly popular and challenging application involves self-driving cars, which make extreme demands on system performance and reliability. Research progress was heavily linked to the existence of datasets such as *KITTI Vision Benchmark Suite*, *CamVid* and *Leuven* datasets. These urban scene datasets are often much smaller than Cityscapes Dataset and does not fully capture the variability and complexity of real-world inner-city traffic scenes. Cityscapes Dataset is specifically tailored for autonomous driving in an urban environment involving a very wide range of highly complex inner-city street scenes that were recorder in 50 different cities. Cityscapes significantly exceeds previous efforts in terms of size, annotation richness, and, more importantly, regarding scene complexity and variability.
Data recording and annotation methodology was carefully designed to capture high variability of outdoor street scenes. Several hundreds of thousands of frames were acquired from a moving vehicle during span of several months, covering spring, summer and fall in 50 cities. For comparability and compatibility with existing datasets they provided low dynamic-range (LDR) 8 bit RGB images that are obtained by applying a logarithmic compression curve. Such tone mapping is common in automotive vision, since they can be computed efficiently and independently for each pixel. 5000 images were manually selected from 27 cities for dense pixel-level annotation, aiming for high diversity of foreground objects, background and overall scene layout.
The dataset provides coarse and fine annotations at pixel level including instance-level labels for humans and vehicles. These 5000 fine pixel-level annotation consist of layered polygons guaranteeing highest quality levels. For 20 000 coarse pixel-level annotations, accuracy on object boundaries was trade off for annotation speed. The authors defined 30 visual classes for annotation, which are grouped into eight categories: flat, construction, nature, vehicle, sky, object, human and void. Classes that are too rare are excluded from the benchmark, leaving 19 classes for evaluation.
Densely annotated images were split into separate training, validation and test

sets. The coarsely annotated images serve as additional training data only. It was chosen not to split the data randomly, rather in a way that ensures each split to be representative of the variability of different street scenarios. Specifically, each of three splits is comprised of the data recorder with the following properties in equal shares:

- In large, medium, and small cities.
- In the geographic west, center, and east.
- In the geographic north, center, and south.
- At the beginning, middle, and end of the year.

Following this scheme resulted in a unique split consisting of 2975 training and 500 validation images with publicly available annotations, as well as 1525 test images with annotations withheld for benchmark purposes.
Finally, defined 30 classes divided into 8 groups are:

- **Flat** - road, sidewalk, parking*, rail track*.
- **Human** - person, rider.
- **Vehicle** - car, truck, bus, on rails, motorcycle, bicycle, caravan*, trailer*.
- **Construction** - building, wall, fence, guard rail*, bridge*, tunnel*.
- **Object** - pole, pole group*, traffic sign, traffic light.
- **Nature** - vegetation, terrain.
- **Sky** - sky.
- **Void** - ground*, dynamic*, static*.

Note that labels marked with * are not included in any evaluation and treated as void (or in the case of *license plate* as the vehicle mounted on).

4.1.3 KPIT traces

The dataset called *KPIT traces* is an internal non-public dataset that belongs to *Volkswagen*. The dataset consists of 5134 images taken from the fish eye rear camera in the car and 117 calibration images. The calibration images includes the chessboard in many different positions which allows to calculate the actual calibration of the camera. The annotation includes 5 classes: *freespace, vegetation, vehicle, pedestrian* and *other*. The classes are divided into 2 groups:

Obstacle classes and *Freespace classes.* Currently the machine learning modules are designed only to recognize obstacles and freespace (therefore dividing into 2 groups). In the future, the car should be also able to recognize what type of obstacle is behind the car to react accordingly (therefore more labels in the annotation).

4.2 Evaluation metrics

To quantify the performance of the model, the well designed metrics are necessary. There many ways how to measure the prediction results. The most essential for semantic segmentation would be an *Pixel Accuracy* and *Intersection Over Union (IoU)* (also known as *Jaccard Index*) and *F1 score*(also known as *Dice Coefficient*).
The metrics were calculated per each class and then the mean value represents the quantified result of the model. During evaluation, for each class at each image a binary bitmap was created setting a pixel to 1 if the pixel was assigned to the given class and zero otherwise for both - predictions and ground truths. Out of these bitmaps, a rate of *TP, TN, FP* and *FN* were calculated.

- **TP** (True positive) - number of pixels that were classified to the class and should have been.
- **TN** (True negative) - number of pixels that were not classified to the class and should have been.
- **FP** (False positive) - number of pixels that were classified to the class and should have not been.
- **FN** (False negative) - number of pixels that were not classified to the class and should have not been.

Pixel Accuracy
Pixel Accuracy is the percent of pixels in the image that were classified correctly. It is perhaps the easiest metric to understand conceptually. Unfortunately, it is definitely not the best one. There is an issue when evaluating semantic segmentation using pixel accuracy. This issue is called *class imbalance* and may occur if there are a very few pixels of one class in the image. For instance, lets consider a simple problem of detecting an object, which is present only in 5% of pixels in ground truth and the model make a prediction of no object detected in the image. The pixel accuracy would be equal to 95% in this case which looks really good, but the actual result is really bad. Despite this issue,

this metric has some information value, therefore it make sense to calculate it. The calculation was done according to the formula:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

Intersection Over Union (IoU)
The Intersection Over Union (IoU), also known as Jaccard index, is one the most common metrics used widely to evaluate semantic segmentation. It is very straight forward metrics which is proven to be extremely effective. As expressed by the formula 4.2 (and visualized in Fig. 4.1) the IoU is the area of overlap between predicted segmentation and the ground truth divided by the area of union between the predicted segmentation and the ground truth. The metric ranges from 0 to 1 (0-100%) with signifying 0 as no overlap and 1 as a perfect overlap.

$$IoU = \frac{||A \cap B||}{||A \cup B||} \tag{4.2}$$

Using a binary bitmaps, the metric can be rephrased in terms of true/false positives/negatives as:

$$IoU = \frac{TP}{TP + FP + FN} \tag{4.3}$$

For multi-class segmentation, the mean IoU of the image is calculated by taking the IoU of each class and averaging them.
To compare this metric with pixel accuracy, lets consider the scenario with class imbalance. The calculation have to be done separately for each class and then the result is average of IoU of each class. First calculate the IoU of the object class. The object was not detected, thus the area of intersection is 0 and then the IoU is 0. For the second class (background) the area of intersection is 95 and the area of union is 100, thus the IoU is 0.95. In this scenario, the mean IoU of the image is 47.5%. Without a doubt that is a much better indicator of the success of the segmentation.
F1 score
F1 score, also known as *Dice Coefficient*, is doubled the area of overlap between the prediction and ground truth divided by the total number of pixels in both images. This metric is very similar to IoU and the results of these metrics are positively correlated. This metric is basically the function of *Precision* and *Recall.* It seeks the balance between precision (how precise was the prediction) and recall (how many actual positives model captured).

$$F1 = \frac{2||A \cap B||}{||A|| + ||B||} \tag{4.4}$$

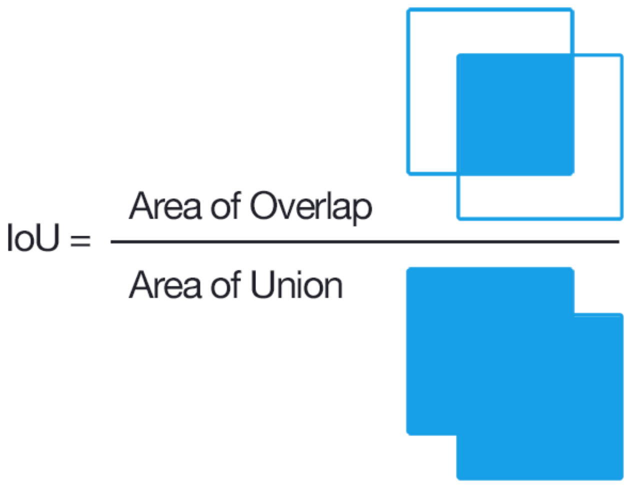


Figure 4.1: IoU calculation visualized [28]

Using a binary bitmaps, the metric can be rephrased in terms of true/false positives/negatives as:

$$F1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{4.5}$$

4.3 Augmentations

Limited amount of available training data is the major obstacle of every deep learning model. There are many ways to address complications associated with limited amount of data in machine learning. *Image augmentations* is one very useful technique in building convolutional neural networks that can increase the size of the training set without acquiring new images. The idea is to duplicate available images with some variation so the model can learn from more generalized examples. The process must be executed carefully to avoid a model learning non-sense data that will not appear in validation and test sets. The augmentations used for training of DeepLabV3+ model were *horizontal flip, random scale* and *color space transformations.*

Horizontal Flip

A horizontal flip is one of the basic methods to create augmentations. A horizontal flip usually does not change the meaning of the image (unlike the vertical flip). The car or the street in Cityscapes images still make a perfect sense even if it is horizontally flipped. This augmentation only creates and problem with the texts, because horizontally flipped text is not readable. But since analysis of the text in the image in not the goal of this model, the only thing that had to be taken care of was to correctly flip also the ground truth during the training.

Random Scale
Training the model with lower resolution images than the original resolution creates a possibility to make a crops and therefore increase the number of training images. The Cityscapes dataset images have a resolution 2048×1024. The DeepLabV3+ model was trained on 513×513 crops.
Color Space Transformations
Image data is encoded into 3 stacked matrices, each of size $height \times width$. These matrices represent pixel values for an individual RGB color value. The effectiveness of color space transformations is fairly intuitive to conceptualize. The color space transformations done during the training were performed as increasing/decreasing the brightness or the contrast of random crops in the images.

4.4 Evaluation on Cityscapes Dataset

The implemented DeepLabV3+ model with Xception backbone was evaluated on the Cityscapes dataset. Cityscapes is a large-scale dataset containing high quality pixel-level annotations of 5000 images (2975, 500, and 1525 for the training, validation, and test sets respectively). The dataset also includes a 20,000 coarsely annotated images which can be used for the training. However, because the goal of the re-implemented model is not to achieve the best results on Cityscapes benchmark, it was decided to not use them.
The Tab. 4.1 shows attained result of evaluation on the Cityscapes validation set quantified using IoU and F1 metrics. The overall result of the model is obtained by averaging the result of each class. As discussed in Sec. 4.2, the accuracy might not be the most appropriate metric to evaluate semantic segmentation. However, the information of what percentage of the pixels was correctly classified could be interesting, hence during evaluation accuracy was also calculated. The model achieved an accuracy of 89.53%.
The Tab. 4.2 shows the IoU result on Cityscapes validation set mentioned in the original paper and the result of the re-implemented model. The original implementation achieved a better class performance ($\sim$ 5%). It is caused by the authors using also 20,000 coarsely annotated images from Cityscapes dataset and pretraining on JFT-300M [27] dataset. Since the aim of this thesis is to obtain a reasonable result and not to reproduce the original results, the performance of 73.55% IoU is satisfying enough. Interesting point is that the re-implemented model achieved a comparable results in terms of category IoU (only 1% less). This result shows that model more often confuses a similar objects which belongs to the same category (human/rider, building/wall, mo-

	IoU	**F1 score**
road	94.5248%	80.5162%
sidewalk	82.0718%	72.4319%
building	86.6678%	73.0096%
wall	65.7022%	74.5499%
fence	62.3170%	60.1392%
pole	61.2422%	56.2688%
traffic light	65.4054%	78.0368%
traffic sign	53.1085%	60.8097%
vegetation	93.1282%	78.3386%
terrain	58.9070%	81.0753%
sky	89.5487%	76.7635%
person	77.5181%	60.7598%
rider	65.9915%	87.1013%
car	85.6959%	84.4297%
truck	77.9868%	90.3239%
bus	69.1686%	86.0706%
train	70.9835%	91.6874%
motorcycle	69.8926%	92.7908%
bicycle	67.5234%	68.5998%
average	**73.5465%**	**76.5107%**

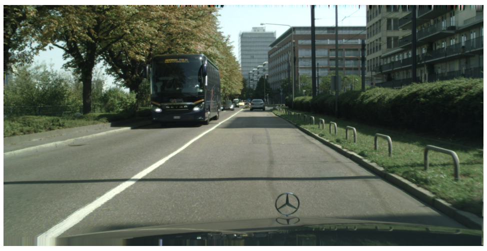
Table 4.1: Cityscapes validation set results for each class.

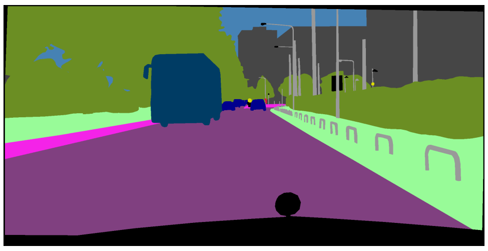
torcycle/bicycle etc.).

	class IoU	category IoU
Original	78.79%	87.99%
Re-Implemented	73.55%	86.83%

Table 4.2: Comparison of the original result with result of the re-implemented model.

Fig. 4.2 shows the comparison of segmentation results 4.2c with the original image 4.2a and the ground truth 4.2b. As one can see, the model is struggling with the recognition of a car hood in the bottom of the image and with distinguishing between the *terrain* (light green) and *vegetation* (dark green). The classes *terrain* and *vegetation* are from the same class category - *nature*, which means that confusing them does not affect the category results. The result visually looks quite well, perhaps a little noisy, but all the objects in the image were recognized.


(a) Original image [9].


(b) Ground truth [9].

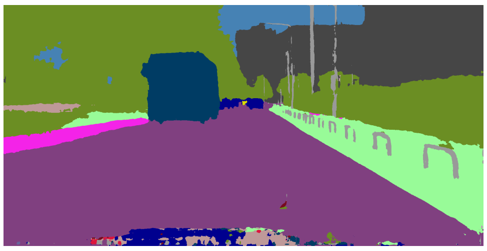
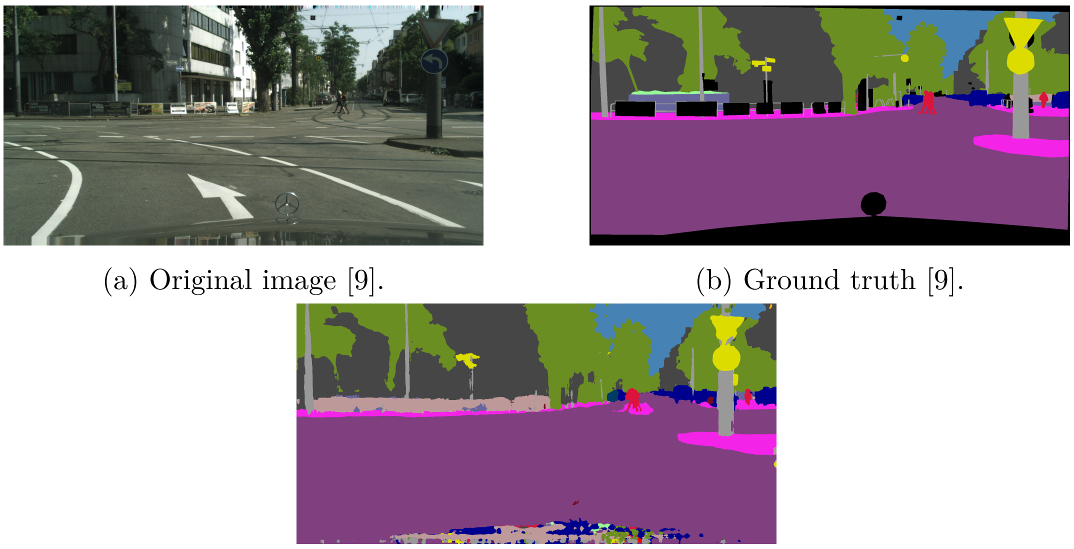

(c) Segmentation prediction.

Figure 4.2: The comparison of the segmentation with its ground truth and original image.

Fig. 4.3c shows the segmentation with wrongly classified objects. In the ground truth 4.3b the advertising tables were marked as *void* but the model predicted them as a *fence*. Very interesting is that the *fence* was also predicted in the car hood. The reason is, that as we can see in the original image, the advertisement is reflected in the car hood.


(a) Original image [9].

(b) Ground truth [9].

(c) Segmentation prediction.

Figure 4.3: The comparison of the segmentation with its ground truth and original image.

4.5 Evaluation on KPIT Traces

The aim of this thesis was to create a model, which performs a semantic segmentation of the images from fish eye camera with a reasonable results. The *KPIT* dataset of 5134 images from fish eye camera was randomly divided into training and validation set (80%/20% respectively). Then the re-implemented DeepLabV3+ model pretrained on ImageNet [11] and Cityscapes [9] was fine-tuned using the KPIT training set.

The two different approaches was employed to fine-tune the model. The first one consisted of replacing a classification layer with the new one and freezing weights in all layers except for the classification layer. The results of this experiment are denoted in Tabs. 4.3,4.4,4.5 as *FW* (frozen weights). The second approach was to replace a classification with the new one and fine-tune the whole model. Results from second experiment are in Tabs. 4.3,4.4,4.5 as *NFW* (not frozen weights).

Because of that the dataset is non-public and this is the first effort to perform the semantic segmentation, there are no results available for comparison and benchmarking. Therefore, the results were mostly evaluated visually with developers and managers involved in the project dealing with image processing of rear camera. However, to set a baseline for future work and to compare the results of the experiments the evaluation was also quantified using the IoU metric. The result of evaluation is shown in Tab. 4.3.

	class IoU	category IoU
Frozen weights	58.32%	63.19%
Not frozen weights	59.26%	64.77%

Table 4.3: IoU results on validation set of KPIT dataset.

The Tabs. 4.4,4.5 shows the results of the experiments for each class and for each category of the dataset. According to the IoU performance, the second approach with fine-tuning the whole model without freezing any layer was proven to achieve a better performance, in concrete 59.26% IoU.

	FW	NFW
freespace	55.26%	56.38%
vegetation	58.15%	56.55%
vehicles	72.14%	73.26%
other	42.47%	48.44%
pedestrian	65.56%	61.67%

Table 4.4: KPIT validation set results for each class.

	FW	NFW
Freespace	81.12%	82.96%
Obstacle	45.26%	46.58%

Table 4.5: KPIT validation set results for each group.

4.6 Future Work

Due to the amount of work and limited time, it was not possible to realize more experiments. Executing another experiments will be part of a future work with the goal to improve performance of the model on the task of semantic segmentation of fish eye camera images. This topic is very perspective in automotive. The semantic segmentation has proven to help the autonomous vehicles to understand the context of the environment around the vehicle. And using fish eye lens to increase the angle of view of the camera, instead of using more cameras, seems to be the new trend in automotive.
During the previous work, a tool calculating the calibration parameters of the camera was developed. The plan is to try two approaches using these calibration parameters: (1) to un-distort the fish eye camera images and perform the segmentation using a standard DeepLabV3+ model, and (2) to distort the Cityscapes dataset images to create more training data. Both approaches could possibly significantly improve the segmentation results. From the automotive point of view, especially the second approach resulting in a good performance could create a very promising model that might significantly support a series development.
During the time of writing this thesis a subcontractor of Volkswagen works on creating a new dataset. According to the assignment, the dataset should be covering a lot of different scenarios under many different conditions. The dataset should include enough images covering all reasonable obstacles behind the car, all possible weather conditions etc. Very interesting is that it should also cover issues like broken glass on the top of the lens or the camera having limited angle of view caused by some dirt and many more.
Both datasets (KPIT and the one in progress) consists of a real world data. The images are noisy which makes the segmentation even more difficult. Having a model that gives a reasonable segmentation results on this real world images would significantly help to the progress of self-driving vehicles.

5 Conclusion

The goal of this thesis was to implement a deep learning model performing a semantic segmentation. First, it was necessary to find a suitable deep neural network architecture capable of semantic segmentation. Next step was to choose a suitable dataset with automotive images eligible for the training of convolutional neural networks and train the chosen model on this dataset. After achieving a satisfying performance the model was fine-tuned with a new dataset to perform the semantic segmentation of the automotive images taken by the fish eye camera.
After a research of available network architectures, it was chosen to implement a state-of-the-art semantic segmentation model on Cityscapes benchmark - DeepLabV3+. The model employs the encoder-decoder structure where a backbone is used to encode the rich contextual information and a simple yet effective decoder module is adopted to recover the object boundaries. In addition to a standard convolutional neural networks algorithms, DeepLabV3+ uses atrous convolution to reduce the resolution of feature maps in encoder without loosing the contextual information and atrous spatial pyramid pooling to detect the objects at different scales.
As the most suitable dataset for the given problem was chosen the Cityscapes dataset. The dataset consists of automotive images taken from the in-car camera capturing the space in front of the car while driving. The dataset provides 5000 fine annotated images divided into 19 classes.
The DeepLabV3+ architecture was re-implemented using Keras and TensorFlow frameworks. The model was evaluated using Cityscapes validation set achieving a performance 73.55% IoU. The performance is lower than the original implementation due to not pre-training the model on JFT-300M dataset and not using a coarsely annotated images from Cityscapes dataset.
Finally, the implemented DeepLabV3+ model was fine-tuned using a KPIT dataset to perform a semantic segmentation of automotive images taken by the car rear camera with fish eye lens. In the evaluation the model achieved a performance of 59.26% IoU.
The semantic segmentation is the key ingredient for the car to obtain the contextual information of its environment. Having a model which performs a semantic segmentation with a reasonable results might help the car to analyze the situation around and react to it in the right way. This model would help to progress a development of a self-driving vehicles.

Bibliography

[1] Badrinarayanan, V. – Kendall, A. – Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2017.

[2] Brostow, G. J. – Fauqueur, J. – Cipolla, R. Semantic Object Classes in Video: A High-Definition Ground Truth Database. *Pattern Recognition Letters*. 2008.

[3] Chen, L.-C. et al. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*. 2014.

[4] Chen, L. et al. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *CoRR*. 2016, abs/1606.00915. Dostupné z: `http://arxiv.org/abs/1606.00915`.

[5] Chen, L. et al. Rethinking Atrous Convolution for Semantic Image Segmentation. *CoRR*. 2017, abs/1706.05587. Dostupné z: `http://arxiv.org/abs/1706.05587`.

[6] Chen, L. et al. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. *CoRR*. 2018, abs/1802.02611. Dostupné z: `http://arxiv.org/abs/1802.02611`.

[7] Chollet, F. – others. Keras. `https://keras.io`, 2015.

[8] Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. *CoRR*. 2016, abs/1610.02357. Dostupné z: `http://arxiv.org/abs/1610.02357`.

[9] Cordts, M. et al. The Cityscapes Dataset for Semantic Urban Scene Understanding. *CoRR*. 2016, abs/1604.01685. Dostupné z: `http://arxiv.org/abs/1604.01685`.

[10] Dai, J. et al. Deformable Convolutional Networks. *CoRR*. 2017, abs/1703.06211. Dostupné z: `http://arxiv.org/abs/1703.06211`.

[11] Deng, J. et al. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[12] Everingham, M. et al. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html, 2010.

[13] He, K. et al. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *CoRR*. 2014, abs/1406.4729. Dostupné z: `http://arxiv.org/abs/1406.4729`.

[14] He, K. et al. Deep Residual Learning for Image Recognition. *CoRR*. 2015, abs/1512.03385. Dostupné z: `http://arxiv.org/abs/1512.03385`.

[15] Krizhevsky, A. – Sutskever, I. – Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F. et al. (Ed.) *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012. s. 1097–1105. Dostupné z: `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

[16] Lecun, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 1998, 86, 11, s. 2278–2324.

[17] Long, J. – Shelhamer, E. – Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *CoRR*. 2014, abs/1411.4038. Dostupné z: `http://arxiv.org/abs/1411.4038`.

[18] Mallat, S. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, Inc., 3rd edition, 2008. ISBN 0123743702.

[19] Mcculloch, W. – Pitts, W. A Logical Calculus of Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*. 1943, 5, s. 127–147.

[20] Minsky, M. – Papert, S. A. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 2017. ISBN 0262534770.

[21] Otsu, N. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*. 1979, 9, 1, s. 62–66.

[22] Ronneberger, O. – Fischer, P. – Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR*. 2015, abs/1505.04597. Dostupné z: `http://arxiv.org/abs/1505.04597`.

[23] Rosenblatt, F. The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. *Psychological Review*. 1958, s. 65–386.

[24] Rumelhart, D. E. – Hinton, G. E. – Williams, R. J. Learning representations by back-propagating errors. . 1986.

[25] Sahu, B. *The Evolution of Deeplab for Semantic Segmentation* [online]. 2019. Dostupné z: `https://towardsdatascience.com/the-evolution-of-deeplab-for-semantic-segmentation-95082b025571`.

[26] Song, S. – Lichtenberg, S. P. – Xiao, J. SUN RGB-D: A RGB-D scene understanding benchmark suite. *Computer Science, 1988-2019.* 2015, s. 567–576.

[27] Sun, C. et al. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. *CoRR.* 2017, abs/1707.02968. Dostupné z: `http://arxiv.org/abs/1707.02968`.

[28] Tiu, E. *Metrics to Evaluate your Semantic Segmentation Model* [online]. 2019. Dostupné z: `https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2`.

[29] Verbeek, J. – Triggs, B. Scene Segmentation with Conditional Random Fields Learned from Partially Labeled Images. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, s. 1553–1560, Red Hook, NY, USA, 2007. Curran Associates Inc. ISBN 9781605603520.

[30] Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE.* 1990, 78, 10, s. 1550–1560.

[31] Widrow, B. Adaline: Smarter than sweet, 1963.