Skolkovo Institute of Science and Technology

Master's Educational Program:

Data Analysis and Management (Computer Science and Engineering)

Semantic Segmentation of Road Scenes

Author: .....
Fedor Chervinskii

Certified by .....
Victor Lempitsky
Associate Professor, PhD
Thesis Supervisor

Accepted by: .....
Clement Fortin
Dean of Education, Skoltech

Moscow

June 2016

Semantic Segmentation of Road Scenes

by

Fedor Chervinskii

Submitted to the Skolkovo Institute of Science and Technology
on May, 2016

Abstract

This work addresses the problem of semantic segmentation accuracy of road scenes seen by a single monocular camera mounted on a vehicle. Dense semantic segmentation is way of road scene parsing to an interpretable representation which is one of main challenges on the way to reliable algorithms for autonomous driving. Deep learning approach to this problem has recently shown the best performance. On the other hand, deep models are mostly hard to train, while computationally expensive. Another requirement is large amount of labeled data, but only small amount of labeled data for this task available at the moment, while much more unlabeled input samples such as videos from dashcams can be easily collected. Different approaches to this problem are compared in experiments while performing this work, such as unsupervised learning of a representation on unlabeled data, data augmentation and model simplification. We also proposed a novel approach to reuse of models, pretrained for different task, such as image classification, for pixel-wise semantic segmentation. Our approach allows to significantly speed up training of SegNet/Deconvnet and simultaneously improve performance when constrained on training data. Recently proposed architectures for pixel-level semantic segmentation such as SegNet and Deconvolutional Network has been used in experiments. For unsupervised representation learning convolutional variational autoencoder (CVAE) architecture is proposed. Improvement gained by different approaches validated quantitatively on recently published CityScapes benchmark and qualitatively on our own regional-specific data. Theoretical insights and interpretations of experimental results are discussed. As a result powerful and yet computationally effective model for semantic segmentation and method for training are proposed.

Thesis Supervisor: Victor Lempitsky
Title: Associate Professor, PhD

Acknowledgments

I dedicate this work to my parents and to my wife who supported me all the way through.

Many thanks to all the Skoltech staff who helped and guided me, especially Educational Office and Office of Student Affairs.

I am very grateful for the help and guidance of Victor Lempitsky, my research advisor at Skoltech.

Finally I would like to thank Yandex company and Anton Slesarev personally for computational resources and all other kinds of support provided.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Road scene understanding is the key component towards autonomous driving. Modern perceptive systems for vehicles include ultrasound sensors, LIDARs, accelerometers, gyroscopes. Still one of the most effective type of perception for autonomous driving is vision. A lot of important parts of roads infrastructure and environment including signs, road marking and traffic lights requires optical visual perception of vehicle to understand a road situation. Efficient visual image parsing is crucial to be able to process the perception to the driving instructions. This work is focused on the semantic segmentation of visual input as a way of visual input parsing.

The main goal of this work is to build a pipeline for training models for robust semantic segmentation of road scenes. We want to achieve state-of-the-art accuracy with the most fast and memory-effective architecture being constrained on amount of labeled data available. Another criteria is how the model generalises for data with different conditions, or taken in different geographical locations or with different camera.

Towards this goal I do the following:

1. I study the literature on road scenes analysis, particularly semantic segmentation, focusing on deep learning methods. I make an overview of data available for training such a models and benchmarks for testing. I choose two publicly available datasets for experiments and validation of our architectures.

2. I perform comparative overview and analysis of existent approaches, chosing for ex-

periments SegNet and DeconvNet architectures - those I consider most effective and powerful. I also discuss on differences, similarities, benefits and drawbacks of chosen approaches.

3. I examine chosen architectures in different scenarios on selected data. I reproduce results reported by theis authors and also make some further exploration of their properties.

4. I propose improvements of selected approaches based on observations made during reproducing experiments and explorations. These improvements allows not only to improve accuracy of models but also dramatically speed up training process.

5. Finally, I examine performance of proposed architectures and methods on selected data, compare it with existent approaches and report the results.

The remainder is organised as following:

The second chapter describes the problem domain and gives an overview of currently existing algorithms, datasets and paradigms in the field.

Chapter three describes architectures and training algorithms as well as all the tips and tricks proposed for achieving the best performance on selected domain.

Chapter four describes the results of experiments and compare these results with current benchmarks. Possible future expansions to the project are also discussed.

Chapter 2

Related Work

2.1 Road Scenes Understanding

In recent days, autonomous vehicles and intelligent road assistance became the hot topic. Not only the biggest automobile concerns are performing an active research and show first production-level applications, but also high-tech companies such as NVidia, Google and many others have already shown their prototypes of autonomous or almost autonomous vehicles. The development of robust and truly intelligent driving systems is supported by dramatic advancements in recognition algorithms gained during the last few years. The paradigm has changed - if before we would write an algorithm that process inputs to the outputs, now we train a model that would do this. From the machine learning point of view, vehicle driving algorithm is just a very complex function from sensors' inputs to driving controls. Different approaches of fitting this function could be considered. Authors of [10] suggest three paradigms:

- **Behavior reflex approach** This approach is becoming more and more popular with arise of so-called end-do-end deep architectures, such as classical work back in 1980-s [36] and very recent work of NVIDIA [6], which map raw visual input to steering commands rightaway.

- **Mediated perception approach** This approach is commonly applied in industrial ADAS (adaptive driving assistance) systems. Algorithm usually consists of several

specific detectors for each type of driving-relevant features such as lane markings, other cars, pedestrians, traffic signs. Outputs of such detectors are aggregated by numerous heuristics into control instructions.

- **Direct perception approach** The latter approach proposed in [10] falls in between the former two, it maps visual input to several meaningful affordance indicators. For example, deep neural network can predict position of the vehicle related to lanes of the road and to surrounding vehicles.

End-to-end approach to the moment shows the most impressive results in controlling vehicle on road, making it following the lane and avoiding collision when approaching another vehicle. Still it suffers from lack of competence when it comes to crossings, when decision making and path planning are necessary. The drawbacks of this approach are discussed in more details in [10]. In this work I consider the semantic segmentation of visual input as a step in one of the latter two approaches, which in my personal opinion, are more viable in sake of practical application.

2.2 Semantic Segmentation

Semantic segmentation task is formulated as to map each pixel of an image to a certain semantic class. For example, in the presented problem domain, to parse visual input an algorithm have to separate road from other moving and static objects. It could be useful to detect sign symbols, traffic lights and lane markings as this is relevant to path planning. All this could be done by aggregation of the output of several specific detection algorithms. Semantic segmentation is able to do all this simultaneously, assigning to all of the mentioned entities on the image its corresponding label. Possible class definitions used in this and in other works are presented on a Table 3.1. However, this approach to scene parsing has drawbacks, for example, pixel-wise labeling doesn't allow to distinguish entities of similar class. Of course, there are some extensions to the approach that allow to do this but in this work we don't consider such methods. Another problem is that dense per-pixel ground truth is required to be able to train such a model. Of course, dense labeling is much more

expensive, that's why datasets are for orders of magnitude smaller than those for image classification. Examples of such datasets are PASCAL Visual Object Classes Challenge [17], LabelMe [40].

2.2.1 Architectures

Before neural networks classifiers used to be constructed on top of RGB-SIFT, HOG and other hand-crafted features, using gradient boosting on decision trees, conditional random fields to predict smooth segmentation. Also some algorithms could utilize information about depth, for example depth-SIFT, pixel locations and planes normals.

Later, when deep learning architectures showed dramatic improvements in performance on classification tasks [13][31], common approach to segmentation became reuse a classification architecture to get low-resolution predictions from different levels of deep neural network and then combine and if necessary upsample them to get pixel-level prediction. Those steps of combining and upsampling often called "decoder" in literature, and subsequently, foregoing network is called "encoder". In most works such as [], VGG-16 [41] architecture is used for encoder. This architecture seems to be an experimentally-found optimal for deep representation could be reused for a different tasks. It showed best performance not only in classification (in which it is already not the state-of-the-art) but in many other domains such as image capture generation [44], transferring style [18] e.t.c.

VGG has several variations. Most popular VGG-16 architecture consists of 5 blocks of convolutions with 2x2 max-pooling and followed by 3 fully-connected layers. Each block consists of two or three convolutional layers with ReLU nonlinearities. Each convolutional layer preform convolutions with 3x3 kernel, and number of kernels increases with the depths from 64 to 512. Thus, after convolutions and max-pooling, the output has the size of an input reduced by the factor of 2^5 and the depth of 512. Usually in applications aside from classification only this first part of the network is reused. Later in this work we call this part of the model "VGG encoder".

Most recent semantic segmentation architectures share the mentioned above VGG-16 encoder architecture and differ only in decoder. Four different approaches that I explore here in details are hypercolumns, Fully Convolutional Network, SegNet and DeconvNet. In

some works, after applying deep model, (Conditional Random Fields, CRF) are used to improve smoothness and accuracy of the results. In this work I don't consider this approach neither when choosing proper architecture nor for comparison, because CRF takes a lot of time for processing. Targeting real applications I don't consider such a method viable in my case.

Hypercolumns

Hypercolumns [22] algorithm follows the natural assumption that while in vector representation of a pixel there is more semantic information, in activations of a previous layer there is more information about spatial structure, so to get accurate segmentation we take for each pixel its vector representation together with corresponding activations from "above" that pixel. So we get a "hypercolumn" for each pixel, which then taken as input for a classifier. Noteworthy, the term "hypercolumn" is taken from neuroscience and describes a column of neurons in visual cortex (V1-neurons) sensitive to edges at multiple orientations and multiple frequencies which interplays with what is desired in observed architecture.

Fully Convolutional Network (FCN)

FCN architecture suggests a way of trainable upsampling of the low-resolution output of the encoder. First outcome of the original work that authors propose concept of fully-convolutional network that is independent of input size. The next key feature of this architecture is trainable upsampling via "deconvolutions". The term "deconvolution" or "deconvolutional layer" after numerous discussions in community has been decided to be called "full convolution" because the operation that is performed is still just convolution. This type of operation is also referred to as "fractionally-strided" convolution. The main difference with regular convolution is that the output has bigger size than the input. The kernel being applied to a patch $m \times m$ produce an output of size $n \times n$ where n can be larger than m. In most cases $m = 1$ and $n = 3, 5, 7$ are used for trainable upsampling, and because this operation is somehow reverse from regular convolution, it has been initially called "deconvolution" although the term deconvolution formally has the different meaning. The concept of this kind of convolutional layers is illustrated on Figure 2-2. This model still shows best

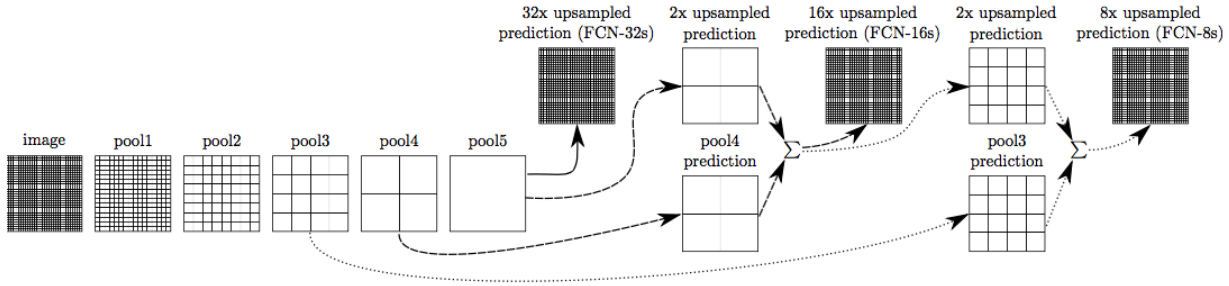


Figure 2-1: FCN architecture [33]. Only selected layers of the decoder are shown. Here all upsamplings are performed through convolutions $1 \times 1 \rightarrow n \times n$ where n is $2, 8, 16, 32$. See Figure 2-2 for details.

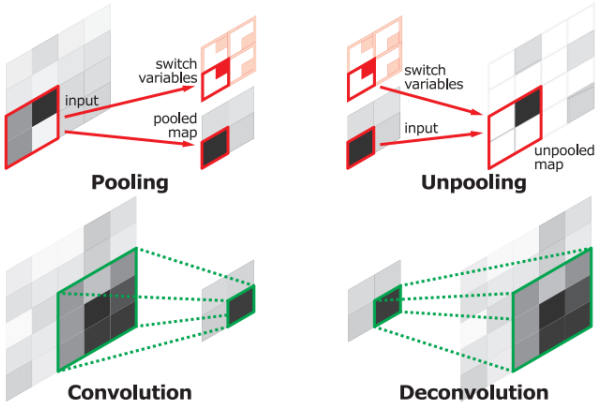


Figure 2-2: Explanation for different types of upsampling. Top right scheme shows the upsampling that is used in SegNet/DeconvNet architecture. Bottom right scheme explains the concept of "deconvolution", which is used in FCN for upsampling and in DeconvNet after unpooling.

performance when feature maps from different levels of encoding (not only the last one) are upsampled and summed up for output classification (see Figure 2-1).

SegNet

This and the next architectures don't use trainable upsampling. Instead, there is a way how we can recover spatial information from the encoder - through max-pooling indices. In this architecture, the decoder use only activations from the last layer of convolutional, and upsample them using max-pooling indices from corresponding subsampling layer. Pixels of the bigger map with the recovered indices are filled with values from smaller map and the rest is left zeroed. After each such upsampling, few layers of convolutions performed to the

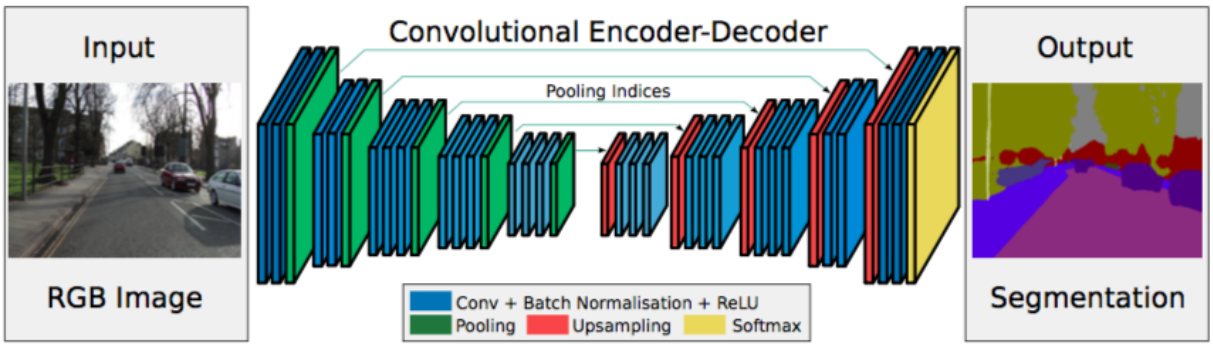


Figure 2-3: SegNet architecture proposed in [4]. Encoder structure copies convolutional part of VGG-16 network. Produced output of the encoder is upsampled and convolved several times until having an input size. On each step of unpooling indices from corresponding max-pooling of decoder are used as shown on Figure 2-2.

Method	Avg. class acc.	Size (M)	Inference time
Hypercolumns	59.2	>134.5	n/a
FCN-8s	62.2	134.5	100ms
SegNet	59.1	**29.5**	**28ms**
DeconvNet	**69.6**	277	92ms

Table 2.1: Comparison of observed architectures on PASCAL VOC 2012 dataset

map until the next upsampling. The resulting decoder structure looks just like mirrored encoder (see Figure 2-3)

Deconvolutional network

The latter approach has a small difference with the previous - encoder is going deeper to the fully connected layers, thus decoder needs to recover all spatial information only from indices. And the second and the main difference that not regular convolutions are applied in the decoder, but fractionally strided, convolving 1×1 to 3×3, as it is described in FCN subsection. That is where the name for this model comes from. This approach seems even more natural because here decoder is truly the mirrored encoder including convolutions. Later in this text I refer to this type of network as to DeconvNet.

Comparative study of aforementioned architectures on benchmarks such as PASCAL VOC, can be found in [4]. We provide aggregation of all benchmarks in Table 2.1

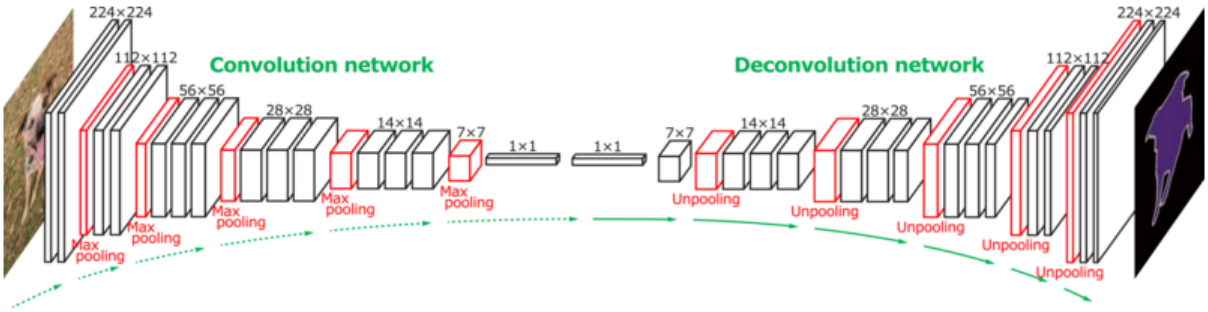


Figure 2-4: DeconvNet architecture proposed in [35]. Only difference with SegNet is additional two 1×1 convolutional layers in the middle and fractionally-strided convolutions in the decoder. Unpooling is performed in the same way as in SegNet.

2.2.2 Segmentation of Road Scenes

Segmentation of road scenes has a lot of specificity comparing to general semantic segmentation task as it is defined in different benchmarks. Firstly, it is mostly dense - that means that most of pixels need to be assigned to one of meaningful classes. For most semantic segmentation benchmarks this is not the case as most of pixels are labeled as background. Luckily though in road scenes segmentation number of classes is constrained. Although this number and notations vary for different works and datasets, classes mostly can be aggregated into several categories. In the following paragraphs I make an overview of datasets in the domain that are publicly available at the moment:

CamVid

The Cambridge-driving Labeled Video Database (CamVid) is a collection of 4 videos taken from camera mounted on vehicle traversing around Cambridge, UK [8] [7]. Videos are taken at the 30fps framerate, and pixel level semantic labels are provided for each 30th, which result in 701 fully labeled image with resolution 960×720 in total. Labels are assigning each pixel to one of the 32 semantic classes. Videos are taken in day and dusk conditions.

KITTI

The KITTI Vision Benchmark Suite (KITTI) [19] is a project of Karlsruhe Institute of Technology and Toyota Technological Institute in Chicago. A part of it is dedicated to

semantic segmentation and for the moment consists of several small sets of frames labeled independently by different researchers and made available to public.

CityScapes

[12] It consist of 3000 densely labeled frames from 12 cities in central Europe and validation set containing another 500 frames from 3 other cities. There are also extra set of 20000 coarsely labeled frames from another 23 cities. It is also worth to note, that while CamVid consists of sequential frames from videos, CityScapes is build of individual frames taken from collected videos by human. Those frames are considered interesting or important for labeling. This property of the dataset on the first sight makes it more diverse and including more possible road scenarios and hard cases, but on the other hand it brakes natural distributions and adds a selection bias.

SynthCity

This is a synthetic dataset consisting of 3000 densely labeled frames generated by rendering of a 3d-model of an imaginary city. Not yet published.

2.3 Unsupervised learning

2.3.1 Motivation

Current datasets for dense semantic segmentation of road scenes not only lack of volume but also very specific to the regions where the data has been collected. Another thing that is affecting the data properties is weather and lighting conditions. That's why applying pretrained model on a data from different domain could be tricky and can not yield desired performance. On the other hand one can easily collect a lot of unlabeled data in different conditions in the region or on the specific route where the model is to be implemented. In this work I study the ways that we could transfer the knowledge from the labeled data to the data from different domain. To do this we need somehow to combine pretrained model with the model (representation) of our new data that captured it's specific intrinsic properties.

2.3.2 Overview of unsupervised methods

When we deal with images, we need our methods to map the target function on very high-dimensional space with dimensionality equal to the number of pixels times number of channels of input image. Most often, target domain, such as any subclass of natural images lies on a low-dimensional manifold in that space and can be efficiently represented in this manifold's subspace. This representation can be learned from the data only. Here we review unsupervised representation learning methods.

Autoencoders and Generative Models

Most of unsupervised learning methods consider an auxiliary task that doesn't require labels. One of the most common scenarios is autoencoder - when the task is to encode an input to a low-dimensional descriptor and then reconstruct the data from it's encoding with the smallest possible damage. Autoencoder can consist of several fully-connected or locally-connected (e.g. convolutional) layers, that process input to the output of the same size [24] while in the middle layers reducing dimensionality. To prevent this model from learning identity function different techniques are applied. One popular trick is Denoising Autoencoder [42]. In this method, in each iteration autoencoder is given an input image with some noise, but the output to be compared with original image. Autoencoders trained in such a manner are showed to capture meaningful features and efficiently reduce dimensionality of data.

Another type of unsupervised representation learning model is generative models. Here the task is to parametrize the distribution of the data. For example Generative Adversarial Networks [20] learn to map multivariate Gaussian to an original distribution. Practically, given a sample from random noise of low dimensionality, network produce a high-dimensional sample that lies ideally on the target manifold (image that "looks like" image from the dataset). This is achieved by training simultaneously the second network, that on every step tries to distinguish produced synthetic output from natural ones. Some extension to this method specific for images have been developed later, such as generating Laplacian Pyramid decomposition of an image [14] or using deep convolutional generator [37].

Variational Autoencoder

Recently, one specific type of autoencoder called variational autoencoder (VAE) [29][27] had become popular. We focus on this method as we chose it for unsupervised representation learning in this work. It involves Bayesian regularisation that force the decoder behave like true generative model, so it accumulates advantages of both approaches.

This method inherits the approach of earlier works such as Deep Belief Networks [34] and Boltzmann Machines [5] of learning to approximate graphical models via neural networks. Here the intention is to learn a directed graphical model of hidden factors z determining an image x in neural network (decoder). But since we don't have knowledge about z given x we train another network to approximate posterior $p(z|x)$ which we then call "encoder". We constrain prior distribution on z to be multivariate Gaussian $\mathcal{N}(0, 1)$ like it is done in generative adversarial networks.

Authors of the method showed that this model can be trained via conventional stochastic gradient descent optimization method with only exclusion. Specific technique called "reparametrization trick" need to be applied. For sample from posterior approximation to be differentiated it needs to be parametrized in the following way

$$z = \mu + \sigma \cdot \epsilon$$

Thus, hidden representation in VAE produced as two vectors, representing predicted mean and standard deviations. Then there is a special layer that sample z from predicted distributions using intrinsically generated noise vector ϵ. The same applies for the output as in terms of Bayesian optimization it's not deterministic but rather a marginal distribution $p(x|z)$.

Optimization objective in this case is variational lowerbound consisting of two terms:

$$\mathcal{L} = -D_{KL}(p(z|x)||p(z)) + log(p(x|z))$$

where the first is Kullback-Leibler distance between predicted approximate posterior distribution and reference multivariate Gaussian and the second is log-likelihood of the input

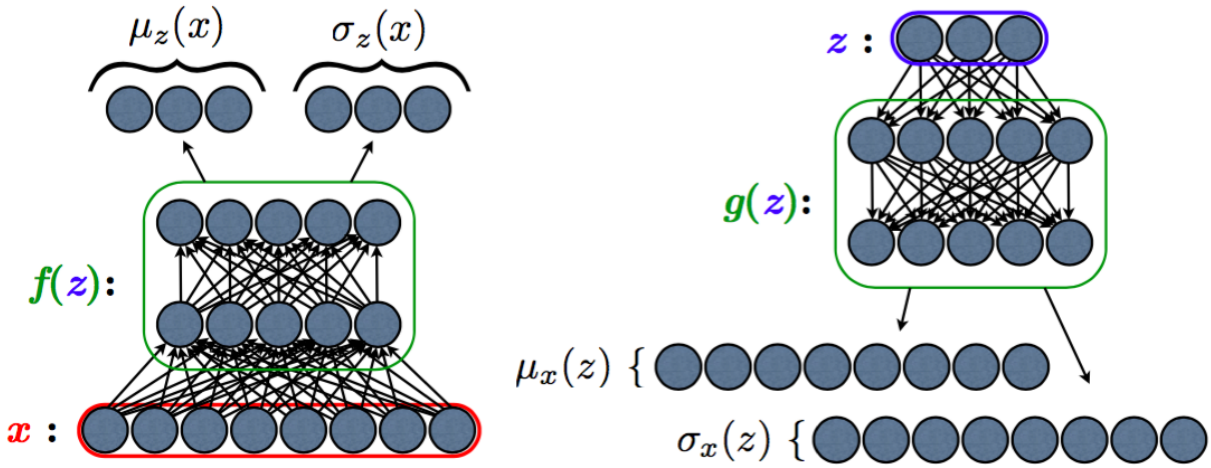


Figure 2-5: Variational Autoencoder concept. Left: encoder that produces parametrized approximation of posterior distribution $p(z|x)$. Right: decoder produces parametrized marginal distribution $p(x|z)$ for z sampled from distribution generated by encoder. *Image credits: A.Courville*

in output distribution predicted by decoder.

VAE is shown to be a flexible approach to representation learning as it allows to learn even more complicated graphical models with complex inference scheme. Later works show applicability of this approach for sequence learning (recurrent VAE) [11], even using attention model [21], semi-supervised learning [28] [32]. Also some extensions and improvements has been proposed in [9] [16] [43] [26].

Other auxiliary tasks

Aside from reconstructing data or mimic data distribution one can consider other auxiliary tasks for learning relevant representation. One bright example of such a task recently proposed in [15]. The network needs to learn to solve puzzles cut out of target images.

2.4 Knowledge Transfer

In this work we going to use different types of transfer learning. Transfer learning is the technique that allows not to train a model from scratch when our target dataset is not large enough but instead reuse features of another pretrained model, even trained for a different

task. Here we review different possible settings of transfer learning:

- **Fixed base model** In this case, we take a pretrained model, or a part of it, and train a classifier or an encoder from the output that this base model produces. For example, it became popular approach, which was proposed and explored in to train classifiers for a wide variety of tasks based on output of the last fully-connected layer of VGG-network

- **Fine-tuning** Fine-tuning is a special term for training a pretrained model on another data with initial model's weights updates. The difference with traditional learning is in the fact that initially we have some representation learned and some layers build upon initialized randomly. When training such a model it's possible to balance training of two parts by setting different learning rates [].

- **Dark Knowledge** The method of this kind has been used introduced in [23] and later called "dark knowledge transfer" refers to the knowledge that is carried by the good pretrained model in prediction probabilities. We can build another model and train it to just copy those probabilities and in this case this model could perform better than if trained optimizing regular cross entropy loss. Normally, logarithmic softmax is computed on the last layer of neural network and used only in negative log-likelihood criterion, while maximum is used as prediction. But all these values turns out to be important when transferring knowledge, that is where the "dark" come from. Later in this work we refer to "dark knowledge" as to any meaningful information that pretrained model provides, but that is not used for prediction explicitly.

2.5 Data Augmentation

Data augmentation is a series of methods that allow to increase efficient amount of training data without collecting new labels. This method showed to yield significant improvements in performance when training models for large scale image recognition [31]. These methods include image scaling, rotation, cropping and flipping, different techniques for changing color

and lighting balance. It's especially useful to perform transformations of input that we want our model to be invariant to.

For semantic segmentation no specific data augmentation methods has been observed so far. This could be because pixel-wise classification itself includes simultaneous parallel processing of many patches of receptive field size. Another technique based on this property has been used in [33], namely loss sampling. It is not truly augmentation but it also solve the problem of diversity of samples. As each of output pixels could be seen as result of processing of a receptive field size input, we can increase diversity of samples exposed to network for training by just randomly sampling output pixels and compute loss only on those pixels. While resulting patches will still have a lot of intersections, authors of [33] report that this technique is showed better results against random cropping of input for fully-convolutional architecture.

Chapter 3

Experiments

3.1 Data and Labeling Policy

For training and evaluating experiments I chose two publicly available datasets: CamVid and CityScapes. These datasets' properties are described in subsection 2.2.2. All architectures tuned to take an input of size 180×240. Thus, all images are resized (cropped) to have this size. I choose the easiest labeling policy which is to aggregate classes from CityScapes and CamVid into 14 categories, as it is suggested in [39] with little exceptions. Correspondence table is shown on Table 3.1.

Noticeable, intentionally or not, there is no any type of lane markings labels in CitiScapes dataset. That's why we set the weight of this class equal to zero while training on this dataset. The same applies for "void" class in all cases.

3.2 Architectures

During my work I perform series of experiments where I benchmark different architectures on the CityScapes dataset. Proposed deep architectures are mainly based on Segnet/DeconvNet approach. This choice is based mainly on practical reasons. Because my goal is to build end-to-end learning architecture, I don't benchmark R-CNN, CRF and other time-consuming methods. I decided not to include hypercolumns in experiments for the sake of time and supposing that the three are all built upon the idea of hypercolumns and should perform

Ours	CamVid	CityScapes
void	void, animal	unlabeled, ego vehicle, ground, dynamic, static, rectification border, rider, rail track, guard rail
sky	sky	sky
road	road, shoulder	road, parking
sidewalk	sidewalk	sidewalk
building	building, archway, bridge, tunnel, wall	buiding, wall, bridge, tunnel
pole	column/pole, traffic cone, parking block	pole, pole group
fence	fence	fence
vegetation	tree, vegetation misc.	vegetation, terrain
traffic sign	sign/symbol	traffic sign
traffic light	traffic light	traffic light
car	motorcycle/scooter, car (sedan/wagon), SUV/pickup truck, truck/bus, train	car, truck, bus, on rails, motorcycle, caravan, trailer, license plate
bicycle	bicyclist	bicycle
pedestrian	pedestrian, child, rolling cart/luggage/pram	person
road markings	lane markings drivable, non-drivable	-

Table 3.1: Labels correspondence

better. Although the reported performance on PASCAL VOC is not the worst, according to the original paper [22] to reach this a lot of tricks such as R-CNN and grid of detectors has been used. In main part of the work I also don't use FCN architecture. Benchmarks, available at the official site of the CityScapes dataset [2] shows that SegNet works significantly faster than all other approaches including FCN. This is the consequence of usage max-pooling indices connections instead of heavy skip connections and 4 parallel flows used in FCN-8s architecture. Although SegNet performance is showed to be not among the best, in this work I show how different training techniques can yield better performance of the SegNet model. In the benchmarks there are no DeconvNet at the moment, but recently [39] showed that it serves very well on road scenes.

Thus, SegNet and DeconvNet architectures have been chosen for experiments. For us these architectures seems also equivalent. It's easy to show that despite border effects, a result of convolving an image with convolutional and backwards convolutional kernel is pretty similar. To be more precise, backwards convolution without upsampling (as used in DeconvNet) is equivalent to a regular convolution with center-mirrored kernel (suppose 3×3 kernel $\{W_{i,j} \,|\, i = \{-1, 0, 1\}, j = \{-1, 0, 1\}\}$):

$$\textbf{conv} : Y_{i,j} = \sum_{k,l=-1,0,1} X_{i+k,j+l} W_{k,l}$$

$$\textbf{deconv} : Y_{i,j} = \sum_{k,l=-1,0,1} X_{i+k,j+l} W_{-k,-l}$$

Based on this observation, it's interesting to see how architectures based on these different types of convolutions differ in training.

For simplicity we remove from DeconvNet two fully-connected layers. As we show later, these layers are redundant, and don't carry much useful information. Without these layers DeconvNet architecture becomes very similar to SegNet with only difference in type of convolutions in decoder. Details of all architectures that are used in experiments are shown in Figure 3-2

3.3 Initialization

3.3.1 Pretrained Encoder

In most cases, when a deep learning architecture for a certain task inherit its structure from one of AlexNet, VGG, GoogLeNet e.t.c. as most successful architectures in large scale image classification domain, it is suggested to initialize the model with weights of an original network trained for classification. This usually speeds up training and lead to better global optimum, as pretrained filter maps of first convolutional layers of such networks has shown to gather relevant semantic information for wide variety of visual recognition tasks. Despite this, authors of [4] report that initialization of SegNet with weights of VGG-16 pretrained on ILSVRC (from [41]) doesn't show any significantly better performance than one trained from randomly initialized weights. On the other hand, in [35] very similar DeconvNet is trained from pretrained weights, but no any comparison against random initialization is reported.

In this work I suggest a qualitative theoretical explanation of this effect and propose a workaround supported with experimental results. Suppose the model at the very start of the training, encoder is initialized with pretrained weights while decoder consists of randomly initialized convolutional filters. Decoder filters has to learn to reconstruct semantic maps based on positions and values of non-zero values after upsampling, which takes some iterations to converge. Until that, big errors are back-propagated through the bottleneck to the encoder and change its filters. That means that while learning to understand meanings of non-zero values placed by encoder maps, decoder is simultaneously changing corresponding filters during back-propagation, which leads to different meanings of the same maps. Such a play of cat and mouse takes many iterations to converge (if manageable), and finally leads to completely different encoder filters at the end. This seems for us to be a reason why pretrained encoder doesn't significantly change neither learning dynamics nor the final accuracy of SegNet/Deconvnet model.

One way to overcome this behavior is to have full encoder, or a part of it, that produces indices, fixed. According to the above hypothesis, SegNet architecture with fixed pretrained encoder should converge much faster to some extremum, which could be not optimal because the encoder is not optimized to the task. We explore this hypothesis in Section 3.4 further

in details.

3.3.2 Unsupervised Representation Learning

Based on motivation described in 2.3.1 I examined an approach of unsupervised pretraining of part of the model on the data from target domain. For this I used extra part of CityScapes dataset which consist of 20k images taken from different cities, but in very similar conditions. As all considered architectures share the same encoder structure, and the encoder is expected to capture structural and semantic information from input pictures - it is natural to pretrain the encoder on the similar data in unsupervised manner, hoping that it will learn relevant representation that will possibly boost the performance of semantic segmentation build on top of it, or at least will speed up supervised training on labeled data for the target task. To achieve this I propose a Convolutional Variational Autoencoder architecture based on VGG-16 CNN encoder.

This autoencoder is naturally build upon the same encoder inherited from convolutional part of VGG-16. We inject two convolutional layers with 1×1 kernels to split predictions into μ and $\log \sigma$ predictions as described in Section 2.3.2. Convolutions with unit kernel is fully convolutional equivalent to fully-connected layer, when the same weights are applied to each pixel of input. Following the terminology introduced in [33] this is a "fully-convolutional" version of a fully connected layer. Outputs of these two layers are taken in vector form by gaussian sampling layer, which samples z from $\mathcal{N}(\mu, \sigma)$, based on input parameters. Then output vector reshaped back to bottleneck size and propagated through the decoder.

After several attempts I took a decoder very similar to what is used in SegNet architecture, only upsampling is performed in different manner. We found out that transferring max-pooling indices while learning autoencoder is generally not a good idea because these indices carry a lot of information about original picture and an autoencoder learns to reconstruct it based only on this information, not using an information flow through the bottleneck. In this case, filters of the encoder stays random, and decoder learns to decode an image from positions of maximums of encoder maps. To avoid this and learn a meaningful representation of a data we break this connections and use convolutional upsampling instead. Usage of fractionally-strided convolutions in the generative part of the network is

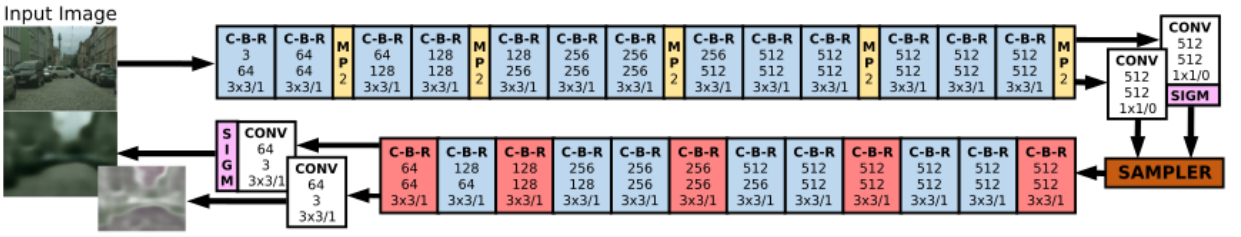


Figure 3-1: Proposed CVAE architecture: encoder structure copies VGG-16 convolutional part, then reparametrization and Gaussian sampling layers are inserted. Sample from the posterior distribution is decoded via sequence of regular convolutonal and fractionally-strided convolutional layers. Output is generated in parametrized form as well.

also inspired by the concept of deep convolutional adversarial network [37] that has shown to be able to generate smooth large images. Another difference with SegNet decoder is the last layer. As suggested in [29] we produce an output in reparametrized way as well, so for the last layer I implement the same procedure as for bottleneck despite there is no need for sampling layer. We only need μ_{out}, σ_{out} for computing likelihood of an original image in the posterior distribution. Schematics of proposed architecture are shown on Figure 3-1.

3.4 Reuse of max-pooling indices

SegNet model while being the most attractive model in terms of memory and computational efficiency, still needs further exploration of underlying principles and of representation it learns. During this work we performed qualitative experiment of building autoencoder based on SegNet. This experiment showed, that an information stored in max-pooling indices of VGG-16 is enough to reconstruct an image with a very good accuracy. This even doesn't depend on encoder convolution kernels. Even for randomly initialized filters, decoder learns to reconstruct an image. That raises the question about how the information flow is split between max-pooling indices and bottleneck values in this type of architectures.

Intuition behind the split is described in [38]. Representation is about "what" and transformation parameters such as pooling indices are about "where". While for classification tasks information about location on the image is often irrelevant, for segmentation task it's contrariwise crucial. That's why upsampling in all segmentation architectures matters a lot. Based on [38] reuse of max-pooling indices seems to be reasonable because these indices keep

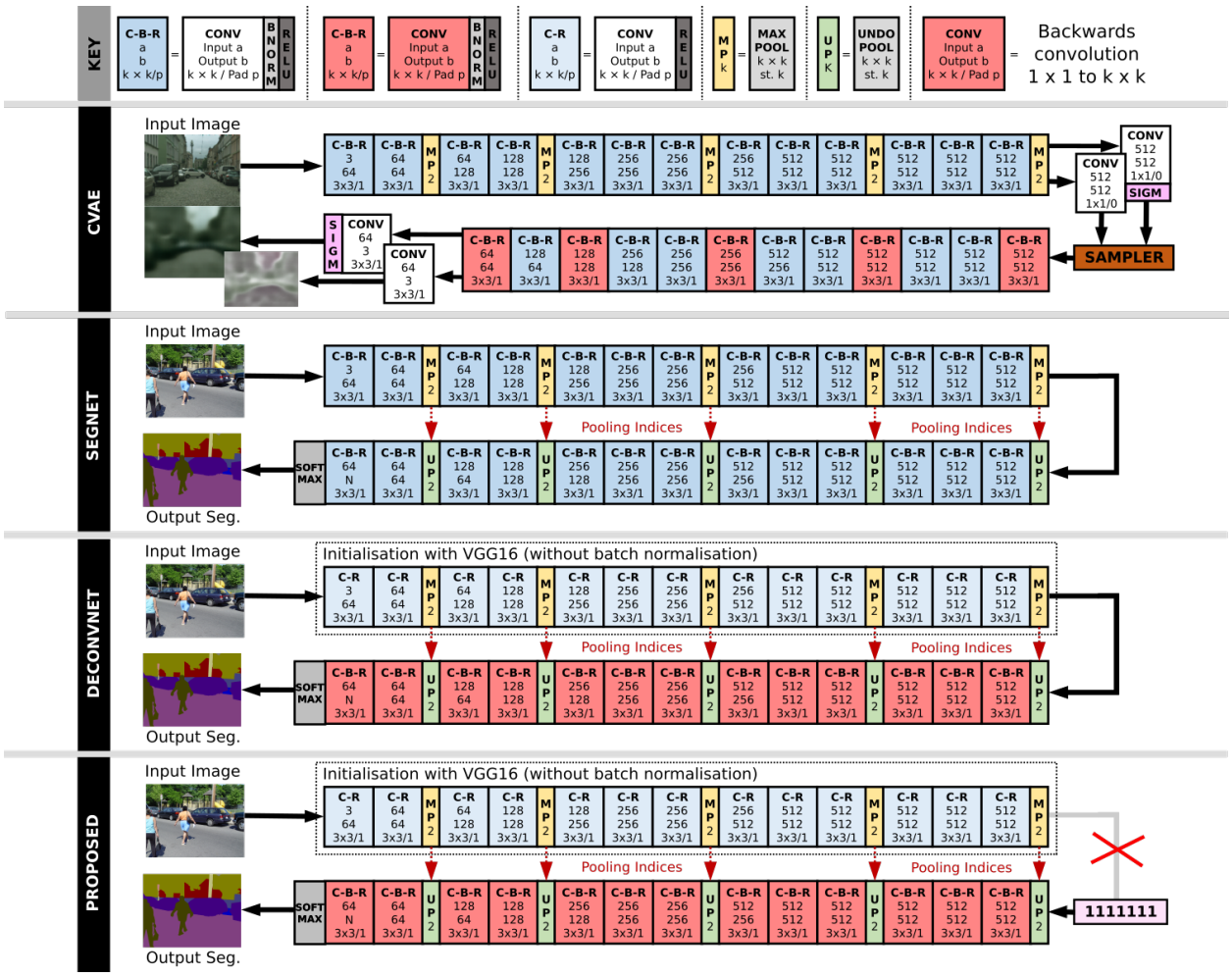


Figure 3-2: Overview of existent and proposed architectures

information about location of features on original image. To explore these properties deeper we suggest a hypothesis that we don't have to learn to produce these indices but we can reuse another network pretrained for a different task, which max-pooling indices still carry information about image structure.

3.5 Proposed approach: decoder from indices

After a series of experiments described in the next chapter we find that even not only all necessary spatial information about an image, but also all necessary semantic information is carried through max-pooling indices. This especially true when we are using an encoder trained for image classification. That's why we propose a novel approach to reuse of pre-trained representation - we train a decoder to reconstruct semantically segmented image based only on max-pooling indices produced by a pretrained convolutional network. Proposed architecture is shown in the bottom of Figure 3-2. In our approach we propagate input images through fixed pretrained VGG convolutional encoder as is, saving only max-pooling indices. Then, we initialize a tensor of the same size with output of the last max-pooling layer of VGG, and fill this tensor with ones. Then we propagate this tensor through the decoder, using stored max-pooling indices from VGG encoder for upsampling. During training we back propagate errors and update weights of the decoder only.

3.6 Data Augmentation

In this work we propose data augmentation technique that is very similar to what is used in image classification, but for best of our knowledge has not been used before for semantic segmentation tasks. Our approach includes cropping of image on different scales and then resizing to a required input size. Because we need to conserve consistency of image and dense labeling we perform these transformations simultaneously on both image and label image, which is just a byte matrix with integer class indices for each pixel. When resizing label in case when new size is not multiple of the current and there is a need for interpolation - nearest neighbors interpolation is used.

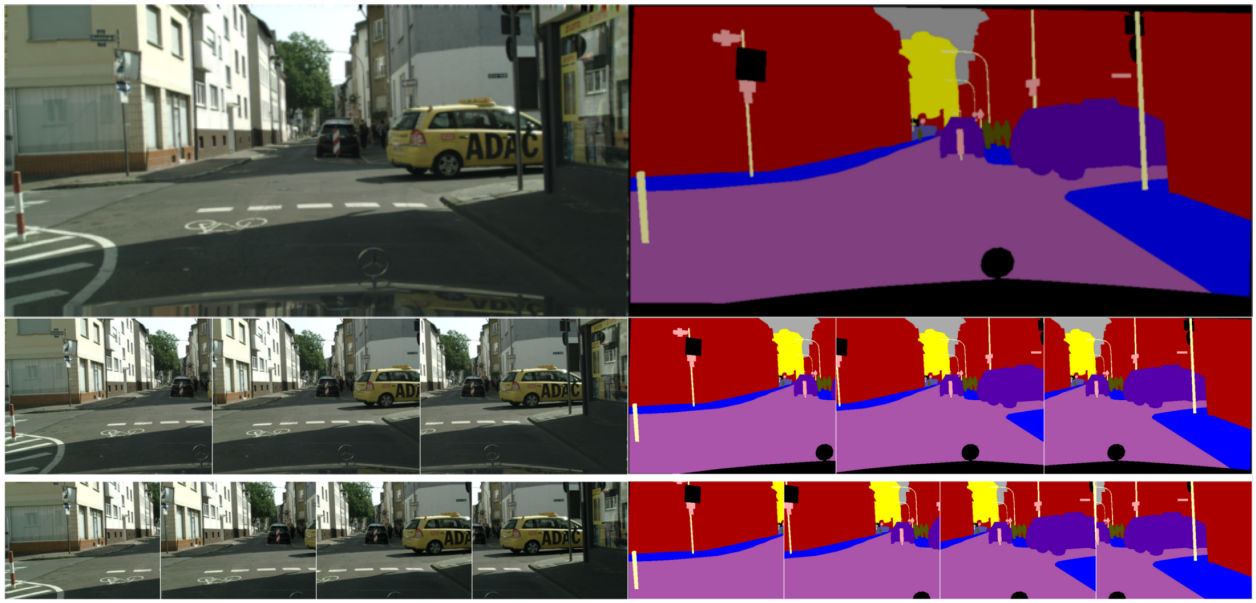


Figure 3-3: Sample of image and corresponding labeling from the validation set of CityScapes dataset (top row) and crops of it that we used in training process when augmenting (two bottom rows). Original images has a size of 1024×512 px. We crop with aspect ratio 4:3 and then scale to 240×180 px.

In most of our experiments we examine the simplest setting when the grid of crops is fixed and on each iteration of training we randomly sample from the grid. Example of grid and samples that we used is shown on Figure 3-3. In addition, we propose a "harder" version of this method, when the crop sizes are fixed, but position of the patch to crop is taken randomly. We report the performance gain of this method in experiments. It seems for us to be important that we do not sample random receptive fields of the same image, but instead sample an image as a whole, completely changing statistics, that should be important for segmentation model to learn. It also can be seen as by sliding the window of cropping we simulate additional steering of imaginary car and by sampling on smaller scale we simulate moving forward thereby increasing efficient size of a dataset without additional expensive labeling.

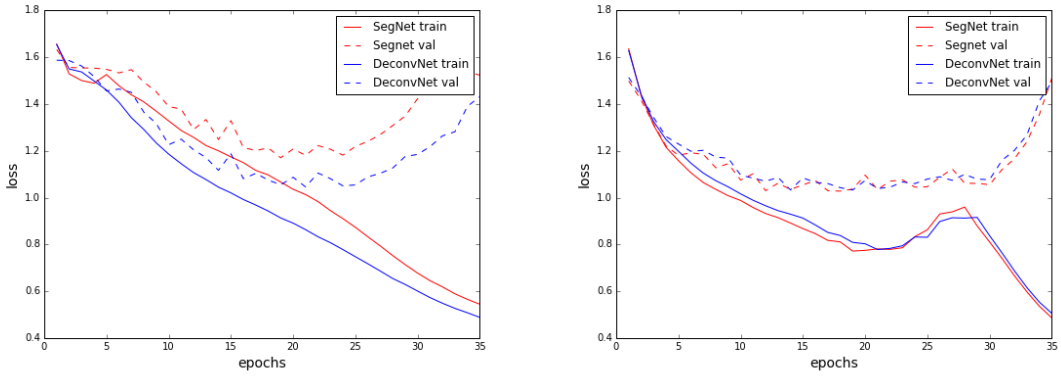
Chapter 4

Experimental Results and Discussions

We perform comparative study of all proposed architectures and approaches on the same data. In the following experiments we use CityScapes training set for training and CityScapes validation set for validation as described in 3.1. For validating we observe mean cross-entropy as well as mean accuracy and Intersection-over-Union (IoU) metrics.

All models and experiments are implemented using Torch7 [3] framework and written in Lua programming language. Training is performed on GPGPUs using NVidia CUDA and CUDNN libs. Weights of VGG-16 model, pretrained on ILSVRC dataset are taken from Caffe model Zoo [1]. All models are trained with one image per batch, with fixed learning rate with Adam optimizer [30]. Despite the fact that the batch size is one image, spatial batch normalization [25] after each convolutional and deconvolutional layer is inserted (unless other specified). In our case, mean and standard deviations are computed over output maps for each channel.

In first series of experiments we compare chosen SegNet and DeconvNet architectures between each other. Each model is trained in two options - random initialization of both encoder and decoder and random initialization of decoder along with encoder initialized with weights of model trained by the VGG team in ILSVRC-2014 [41]. It needs to be mentioned that conditions are not fairly similar as we use batch normalization in the encoder when training from scratch but don't include it when build upon pretrained encoder. It is possible that incorporating batch normalization in the second case would lead to different results but we believe that batch normalization is a technique important for initial stage of training.



(a) (b)

Figure 4-1: Learning curves of SegNet and DeconvNet trained on CityScapes dataset **(a)** from scratch **(b)** from VGG weights initialization of encoder

When fine-tuning from pretrained filters, weights' values are already optimally normalized and centered so using batch normalization from this state should not influence training dynamics.

Figure 4-1 illustrates training dynamics of these combinations. As one can see, both models smoothly converge and very soon start to overfit to the data. One can also notice that proposed architectures almost don't differ in learning trends. That gives us an experimental proof of a theoretic observation made in Section 3.2 that two proposed methods of upsampling using indices of max-pooling and different kinds of convolutions are very similar, not to say equivalent. Also we observe that initialization of encoder weights with pretrained VGG weights slightly improves the convergence speed, but finally doesn't have significant influence on performance. On the Figure 4-2 some filters of three levels of encoders trained in different conditions are visualized. I took three convolutional layers: `conv_1_1`, `conv_3_1` and `conv_5_1`. These are three first convolutional layers of the first, middle and the last blocks of convolutions in VGG-16 encoder. Weights of these layers have dimensionalities $64 \times 3 \times 3 \times 3$, $256 \times 128 \times 3 \times 3$ and $512 \times 512 \times 3 \times 3$ accordingly. I visualize only 3 first components (for the first layer that is all) of 64 first filters of each layer. In the top row are filters from DeconvNet trained from scratch, in the second from DeconvNet trained from VGG initialization and the last row is original VGG weights for reference. This image displays how the same encoder structure is utilized by the model in different conditions.

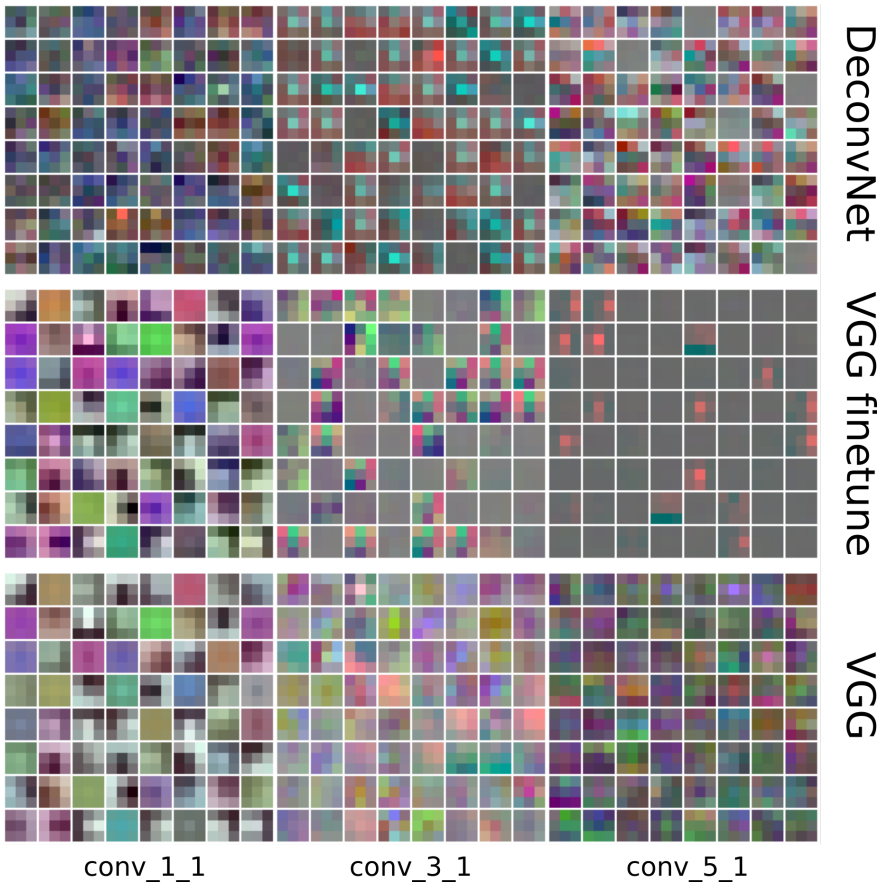


Figure 4-2: Visualization of first 3 components of first 64 filters of some convolutional layers from models trained in different conditions. **Top:** DeconvNet trained from scratch. **Middle:** DeconvNet trained from VGG initialization. **Bottom:** Original VGG weights.

In first case, filters are learned only by back-propagating error of segmentation on the decoder output. In these settings encoder kernels train specifically to optimize values that they sum up to in bottleneck. Training process doesn't take into account the other way how kernels influence the output - max-pooling indices. This process can be seen as greedy optimization - we optimize encoder kernels w.r.t. their income to output and in second order we optimize decoder kernels to reconstruct output based on both values and their upsampling positions defined by encoder kernels. Hypothetically, this process is barely guaranteed to converge to a global optimum.

Situation when the same optimization starts from good initial approximation, which is VGG weights, is shown in the middle row of the figure. Noticeable, maps become more sparse on deeper levels. It seems like weights and filters that are not relevant to semantic categories of this task are neglected by the model. This also should lead to much less information in indices produced by these sparse filters. Interestingly, filters of the first convolutional layer stayed almost as initialized.

The reference model VGG-16 seems to have smooth dense filters that means that it uses its capacity optimally for the task it has been trained. Our next hypothesis is that this model produce meaningful max-pooling indices for all the filters. On the other side, if we have a function that produced meaningful indices determined only by an input from the start, then optimization process of our model should speed up significantly and should converge to a better local optimum.

We examine this hypothesis by implementing the following training process: on each iteration we first propagate input image forward through an original VGG encoder and save 5 groups of max-pooling indices that it produced. Then we propagate the same input through our SegNet/Deconvnet model, using stored max-pooling indices for upsampling in the decoder instead of indices produced by encoder of the model being trained. The encoder of our model is either randomly initialized or initialized with VGG weights as before.

The results are shown on Figure 4-3. We compare learning curves for SegNet and DeconvNet trained using max-pooling indices produced by VGG encoder with original SegNet and find that our method shows dramatic increase in both convergence speed and final performance. Both models converge to minimum validation loss in 4 epochs, which is 5 times faster

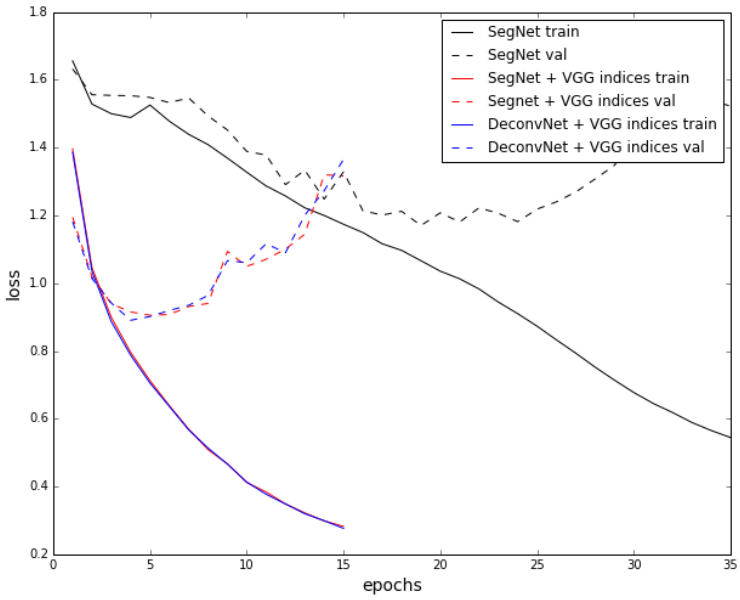


Figure 4-3: Comparison of learning dynamics of SegNet in initial settings and proposed SegNet/DeconvNet training using upsampling based on pretrained VGG encoder max-pooling maps.

than initial SegNet model. There is still a disadvantage of proposed technique that now forward time increased by the factor of 1.5 because we need to perform additional propagation through VGG encoder.

For proposed combination we can try to use a pretrained representation. As a side experiment we train a variational autoencoder in the way that is described in 3.3.2. An example of CVAE reconstruction is shown on Figure 4-4. The model has shown to learn a representation of spacial structure of road scenes. We used the encoder part of this model for initialization of SegNet model, but this did not bring any improvement, the performance in this setting was even worse that if training from scratch. This result could be interpreted as another evidence of importance of max-pooling indices in SegNet architecture as spacial structure information carriers.

Finally we examine how important is the information flow through the bottleneck in SegNet/DeconvNet architectures, we examine models in two very similar settings of SegNet/DeconvNet architectures. In the first setting we use only fixed pretrained VGG as

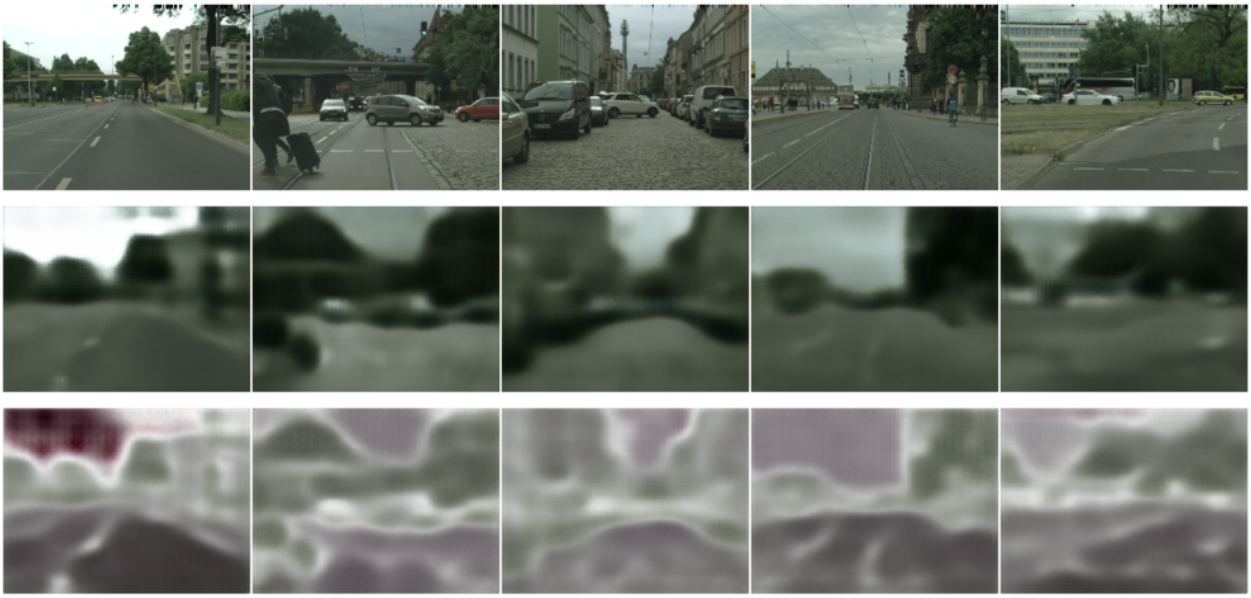
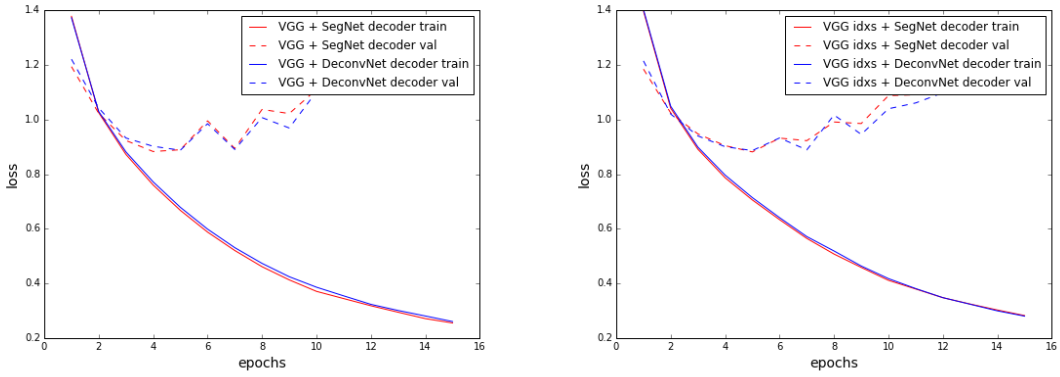


Figure 4-4: Convolutional Variational Autoencoder output. Top row: input, second row: output μ, third row: output σ.

encoder, the decoder reconstruct a dense output starting from output VGG features and based on max-pooling indices produced by it as well Figure 4-5 (a). In the second setting, we restrict model for only using indices produced by VGG, taking as input tensor of ones as shown on the bottom scheme of Figure 3-2 and the result is shown on Figure 4-5 (b).

As one can see from the results, out models still suffer from overfitting. On possible cause of it is our usage of batch size of one image and the models found out how to remember each. Another reason is that we have relatively small dataset. The first problem could be solved by random sampling of loss as described in Section 2.5. But this technique does not increase effective amount of data. We intent to push the performance of the models further by increasing the amount of data by using data augmentation by fixed-grid scaling and cropping as described in Section 3.6. Since we showed before theoretically and experimentally that our implementations of SegNet and DeconvNet are equivalent for training, in this and following experiments we examine only SegNet architecture. The results of training models on augmented data are shown on Figure 4-6. The plot shows that proposed data augmentation strategy leads to a much longer convergence of all models. Still SegNet trained from scratch converges to a higher loss than to SegNet decoder build upon VGG encoder. Also two ways of using pretrained encoder are compared: with use of VGG output values and with use of



(a) (b)

Figure 4-5: Learning curves of SegNet and DeconvNet with fixed (without fine-tuning) VGG encoder **(a)** with bottleneck values propagation **(b)** without bottleneck, only indices are used for decoding.

max-pooling indices only, as in previous experiment. Curves show that decoder from indices performs slightly worse than when using values, but still much better than SegNet trained from scratch.

Finally we report the metrics that shows our models. During the training we performed natural frequency class balancing, so the target metric to be optimized is average class accuracy. We also tried to train without class balancing, which resulted in better global accuracy but yielded worse results for most of the classes, which is reported in the last row of Table 4.2. In the Table 4.1 models described above are compared between each other. Class accuracies and three metrics for different models are shown. Metrics are: average class accuracy, global average accuracy (percentage of correctly classified pixels) and mean intersection over union (I/U), also called Jaccard index, commonly used in semantic segmentation since PASCAL VOC 2012 challenge. Four settings that we report results for here are conventional SegNet model trained from scratch as in [4], SegNet decoder trained on the output of fixed VGG convolutional encoder, SegNet decoder trained only on max-pooling indices from VGG encoder and the last one is SegNet decoder trained on the output of VGG with harder augmentation technique used. The table shows that models that reuse VGG indices perform significantly better than SegNet as is. Also we proved that without propagation of bottleneck values, SegNet decoder still performs comparably with those model

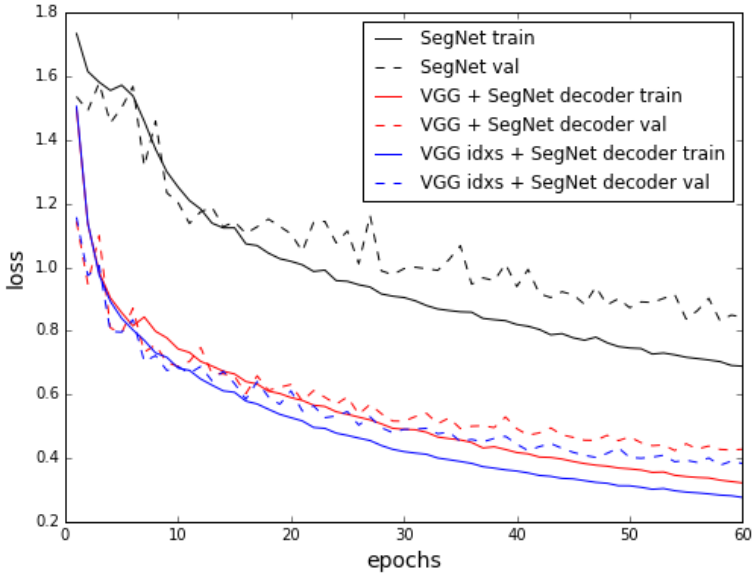


Figure 4-6: Learning curves of SegNet trained in different settings when using data augmentation strategy proposed. Black curve denotes SegNet in initial settings, blue curve is SegNet decoder with fixed VGG encoder and red curve is proposed approach of decoding from VGG indices.

that use these values. The last model shows the best performance overall which proves the efficiency of proposed augmentation methods.

In the Table 4.2 we compare our results with original SegNet performance on CamVid dataset. Original SegNet has been trained on the dataset of size 3.5k which is comparable to CityScapes (2975 train images) and tested on CamVid dataset for the same categories (excluding traffic lights, they merged into Sign-Symbol class). We should note that Cityscapes dataset is much more challenging that CamVid as it contains much more difficult scenarios, more crowd scenes with hard lighting conditions. In addition, validation and training frames are taken in different geographical locations. To compare SegNet model in different settings trained in our experiments with original SegNet model we test them on the whole CamVid dataset and compute the same metrics. We do not fine-tune our models to the dataset so additionally we check the consistency of two datasets and robustness of models trained. The table shows that while we trained on different dataset, models perform comparably or better for some classes on testing data. While SegNet trained from scratch on CityScapes perform

Method	*Building*	*Tree*	*Sky*	*Car*	*Sign-Symbol*	*Traffic light*	*Road*	*Pedestrian*	*Fence*	*Column-Pole*	*Side-walk*	*Bicyclist*	*Class avg.*	*Global avg.*	*Mean I/U*
SegNet ours	77.5	86.4	81.9	88.5	72.4	31.4	93.8	37.2	**23.0**	90.6	83.0	**49.5**	67.9	63.4	35.6
VGG + SegNet decoder	88.1	91.2	81.9	**91.0**	82.5	42.3	**94.5**	61.3	21.1	94.4	86.3	45.0	73.3	79.6	**46.8**
VGG + SegNet decoder from indices	86.4	90.3	82.1	89.7	78.1	37.1	94.3	60.8	21.1	**94.6**	**86.5**	47.2	72.3	76.4	45.3
VGG + SegNet decoder (hard augm.)	**89.3**	**90.5**	**82.4**	89.1	**83.6**	**45.6**	94.4	**64.3**	18.9	92.2	85.0	46.1	**73.5**	**86.2**	41.0

Table 4.1: Performance metrics of SegNet architecture trained in different settings. First row: our implementation of SegNet, trained from random initialization. Second row: SegNet decoder trained on top of fixed pretrained convolutional VGG encoder. Third row: SegNet decoder trained to decode from indices only. Fourth row: VGG encoder and SegNet decoder trained with hard augmentation. All models are trained and tested on CityScapes dataset.

Method	*Building*	*Tree*	*Sky*	*Car*	*Sign-Symbol*	*Road*	*Pedestrian*	*Fence*	*Column-Pole*	*Side-walk*	*Bicyclist*	*Class avg.*	*Global avg.*	*Mean I/U*
SegNet (CamVid) [4]	73.9	**90.6**	90.1	86.4	69.8	94.5	**86.8**	**67.9**	74.0	**94.7**	**52.9**	80.1	**86.7**	**60.4**
SegNet ours	75.1	74.8	95.0	86.9	81.1	90.8	75.1	26.9	91.3	71.0	28.9	72.5	56.1	20.9
SegNet (VGG indices)	82.0	82.1	94.6	89.9	83.5	94.9	80.6	38.2	93.1	91.1	27.4	78.0	69.6	32.3
VGG + SegNet decoder	88.8	85.0	96.8	93.7	89.2	96.2	78.8	35.0	94.0	91.7	29.6	79.9	79.5	37.6
VGG + SegNet decoder from indices	85.3	84.1	96.1	91.8	84.3	95.7	81.9	43.6	**94.8**	90.9	29.3	79.8	73.7	34.4
VGG + SegNet dec. (hard augm.)	89.3	84.9	**97.5**	93.7	**90.6**	**95.8**	81.9	33.7	94.6	91.6	29.2	**80.3**	79.7	38.2
VGG + SegNet dec. (hard augm. no bal.)	**90.8**	85.3	97.2	**94.9**	72.1	95.5	71.0	20.7	90.9	90.6	21.6	75.5	83.1	39.4

Table 4.2: Performance metrics of described models in comparison with the results from [4]. No fine-tuning to the dataset has been done, whole CamVid is used for testing.

much worse than the original one, other models that reuse pretrained encoder are showing competitive accuracy for most of the classes. Most difficult for our models are the classes "Fence" and "Bicyclist" which is partially the consequence of datasets' inconsistency. For example, there is no "Bicyclist" class in CityScapes dataset, but each bicyclist is labeled as two separate instances of "bicycle" and "rider" classes. According to our labeling policy, "rider" class assigned to void and models are not penalised for labeling it, so they naturally learn to label "rider" as "pedestrian" in most of cases. That's why on CamVid dataset it fails on "Bicyclist" class. Still with hard augmentation technique, SegNet decoder trained on top of VGG encoder outperforms original SegNet model trained on larger amount of data and tuned for CamVid dataset by mean class accuracy.

4.1 Conclusions

In this work we examined for the moment the two most promising architectures for the semantic segmentation on road scenes data. We showed in the experiments supported by

theoretical observations, that these architectures are almost equivalent. We experimented on the efficiency of those models and showed that training encoder is unnecessary for the task. Using pretrained encoder allows to train faster, reduce overfitting and increase resulting performance.

We explored possibilities of unsupervised representation learning specifically for the road scenes data domain with possible application to reuse of pretrained encoder in segmentation model. As a negative result, we showed that encoder pretrained in such a way doesn't significantly influence the model performance. Nevertheless we suppose that further exploration of unsupervised representation learning for this data domain is needed, because it could be useful not only for semantic segmentation, but for other road parsing and recognition tasks, even possibly for direct control.

We showed, how the knowledge transfer and inference can be performed not through the output values, but through max-pooling maps of the convolutional network. As before max-pooling indices reuse has been observed only in scope of improving upsampling in networks with necessarily dense output, now we found out that full information about an input is carried through these indices and this information is enough for image reconstruction as well as for image segmentation. This idea has been not yet discussed in literature and can possibly lead to some new methods in image recognition.

We also demonstrate a power of data augmentation for the task of pixel-wise semantic segmentation and proposed the way how the augmentation should be performed for the road scenes segmentation task.

Finally, we offer the pipeline for training a robust and precise model for semantic segmentation of road scenes that includes data augmentation and reuse of pretrained model. The code of experiments and the final pipeline is going to be available online.

4.2 Future work

In the experiments we showed, that max-pooling indices of a deep convolutional neural network provide a lot of information about input image, and this information can be successfully used without any additional knowledge for semantic segmentation of images. We consider

that there will be a lot of extensions of proposed approach, such as reduce of dimensionality in indices space. We can work out an efficient binary image representation (descriptor) through its max-pooling indices that can be used for image retrieval and search of similar images. Another type of regularization such as max-pooling indices dropout could be used. Efficient implementation of segmentation inference through binary values of pooling indices seems to be a promising way to speed up this type of models for deploy in real applications.

There is still an open question which pretrained model is better to use for encoder part and how to reduce a number of parameters in the model. Dark knowledge transfer seems to be one way of doing it. We showed that not values are important but any model that will produce the same pooling indices as VGG will work comparably.

Datasets consistency is another subject for study. We learned that transferring labels between datasets could be challenging and could bring unexpected drops in performance, which should be taken in consideration when it comes for validation.

Bibliography

[1] Caffe model ZOO. https://github.com/BVLC/caffe/wiki/Model-Zoo.

[2] CityScapes Dataset Benchmarks. https://www.cityscapes-dataset.com/benchmarks/. Accessed: 2016-05-07.

[3] Torch7: a scientific computing framework for LuaJIT. http://torch.ch.

[4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015.

[5] Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009.

[6] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to End Learning for Self-Driving Cars. *ArXiv e-prints*, April 2016.

[7] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, xx(x):xx–xx, 2008.

[8] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV (1)*, pages 44–57, 2008.

[9] Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *CoRR*, abs/1509.00519, 2015.

[10] Chenyi Chen, Ari Seff, Alain L. Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. *CoRR*, abs/1505.00256, 2015.

[11] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *CoRR*, abs/1506.02216, 2015.

[12] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[13] Y. Le Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson. Advances in neural information processing systems 2. chapter Handwritten Digit Recognition with a Back-propagation Network, pages 396–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[14] Emily L. Denton, Soumith Chintala, Arthur Szlam, and Robert Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. *CoRR*, abs/1506.05751, 2015.

[15] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. *CoRR*, abs/1505.05192, 2015.

[16] S. M. A. Eslami, N. Heess, T. Weber, Y. Tassa, K. Kavukcuoglu, and G. E. Hinton. Attend, Infer, Repeat: Fast Scene Understanding with Generative Models. *ArXiv e-prints*, March 2016.

[17] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.

[18] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[19] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[20] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

[21] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. DRAW: A recurrent neural network for image generation. *CoRR*, abs/1502.04623, 2015.

[22] Bharath Hariharan, Pablo Andrés Arbeláez, Ross B. Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. *CoRR*, abs/1411.5752, 2014.

[23] G. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network. *ArXiv e-prints*, March 2015.

[24] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

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

[26] D. Jimenez Rezende, S. Mohamed, I. Danihelka, K. Gregor, and D. Wierstra. One-Shot Generalization in Deep Generative Models. *ArXiv e-prints*, March 2016.

[27] D. Jimenez Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *ArXiv e-prints*, January 2014.

[28] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. Semi-Supervised Learning with Deep Generative Models. *ArXiv e-prints*, June 2014.

[29] D. P Kingma and M. Welling. Auto-Encoding Variational Bayes. *ArXiv e-prints*, December 2013.

[30] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[32] T. D. Kulkarni, W. Whitney, P. Kohli, and J. B. Tenenbaum. Deep Convolutional Inverse Graphics Network. *ArXiv e-prints*, March 2015.

[33] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[34] A. Mnih and K. Gregor. Neural Variational Inference and Learning in Belief Networks. *ArXiv e-prints*, January 2014.

[35] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, 2015.

[36] Dean A. Pomerleau. Alvinn, an autonomous land vehicle in a neural network. *technical report*, Research Showcase @ CMU, 1980.

[37] A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *ArXiv e-prints*, November 2015.

[38] Marc'Aurelio Ranzato, Fu-Jie Huang, Y-Lan Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR'07)*. IEEE Press, 2007.

[39] G. Ros, S. Stent, P. F. Alcantarilla, and T. Watanabe. Training Constrained Deconvolutional Networks for Road Scene Semantic Segmentation. *ArXiv e-prints*, April 2016.

[40] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Labelme: A database and web-based tool for image annotation. *Int. J. Comput. Vision*, 77(1-3):157–173, May 2008.

[41] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

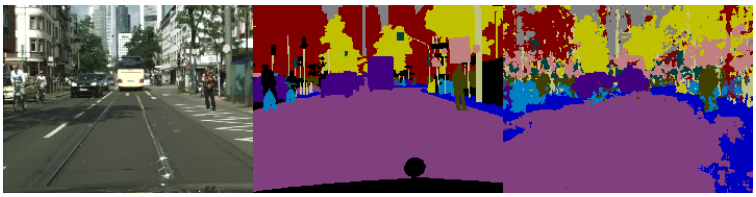
[42] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010.

[43] X. Wang and A. Gupta. Generative Image Modeling using Style and Structure Adversarial Networks. *ArXiv e-prints*, March 2016.

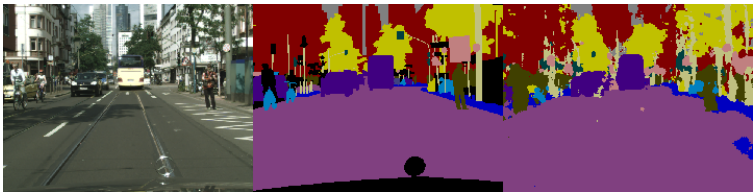
[44] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015.

Appendix A

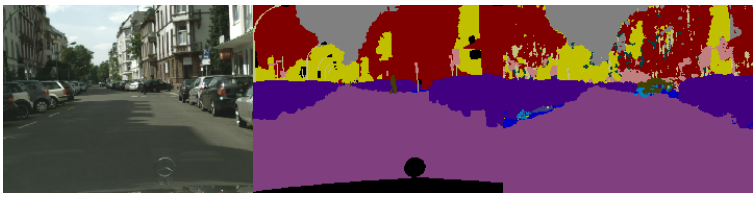
Examples



(a) SegNet



(b) VGG + SegNet decoder



(c) SegNet

(d) VGG + SegNet decoder

(e) SegNet

(f) VGG + SegNet decoder

Figure A-1: Examples of output of trained models for tree different inputs and two different training settings: SegNet from scratch and SegNet decoder built upon VGG features and indices. Left: input, middle: ground truth, right: output. All images are taken from CityScapes validation set.