ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Master's Thesis

DEEP FULLY RESIDUAL CONVOLUTIONAL NEURAL NETWORK FOR SEMANTIC IMAGE SEGMENTATION

Ali Tousi

Department of Computer Science and Engineering

Graduate School of UNIST

2018

DEEP FULLY RESIDUAL CONVOLUTIONAL NEURAL NETWORK FOR SEMANTIC IMAGE SEGMENTATION

Ali Tousi

Department of Computer Science and Engineering

Graduate School of UNIST

Deep Fully Residual Convolutional Neural Network for Semantic Image Segmentation

A thesis
submitted to the Graduate School of UNIST
in partial fulfillment of the
requirements for the degree of
Master of Science

Ali Tousi

11. 06. 2018
Approved by

Jaesik Choi

Advisor
Jaesik Choi

Deep Fully Residual Convolutional Neural Network for Semantic Image Segmentation

Ali Tousi

This certifies that the thesis of Ali Tousi is approved.

11. 06. 2018

Jaesik Choi

Advisor: Jaesik Choi

[illegible]

Committee Member: Jae-Young Sim

[illegible]

Committee Member: Jun Moon

Abstract

The goal of semantic image segmentation is to partition the pixels of an image into semantically meaningful parts and classifying those parts according to a predefined label set. Although object recognition models achieved remarkable performance recently and they even surpass human's ability to recognize objects, but semantic segmentation models are still behind. One of the reason that makes semantic segmentation relatively a hard problem is the image understanding at pixel level by considering global context as oppose to object recognition. One other challenge is transferring the knowledge of an object recognition model for the task of semantic segmentation. In this thesis, we are delineating some of the main challenges we faced approaching semantic image segmentation with machine learning algorithms.

Our main focus was how we can use deep learning algorithms for this task since they require the least amount of feature engineering and also it was shown that such models can be applied to large scale datasets and exhibit remarkable performance. More precisely, we worked on a variation of convolutional neural networks (CNN) suitable for the semantic segmentation task. We proposed a model called deep fully residual convolutional networks (DFRCN) to tackle this problem. Utilizing residual learning makes training of deep models feasible which ultimately leads to having a rich powerful visual representation. Our model also benefits from skip-connections which ease the propagation of information from the encoder module to the decoder module. This would enable our model to have less parameters in the decoder module while it also achieves better performance. We also benchmarked the effective variation of proposed model on a semantic segmentation benchmarks.

We first make a through review of current high-performance models and the problems one might face when trying to replicate such models which were mainly arise from lack of sufficient provided information. Then, we describe our own novel method which we called deep fully residual convolutional network (DFRCN). We showed that our method exhibits state of the art performance on a challenging benchmark for aerial image segmentation.

Contents

Abstract **1**

I. Introduction **1**

II. Large Scale Computer Vision **3**

2.1 Introduction . . . 3
2.2 ImageNet Classification with Deep Convolutional Neural Networks . . . 3
2.3 Large Scale Semantic Segmentation . . . 3
2.4 RefineNet (downside of dilated convolution) . . . 6
2.5 Pyramid Scene Parsing Network (PSPNet) . . . 6
2.6 Rethinking Atrous Convolution (DeepLab-v3) . . . 7

III. Implementation of state-of-the-art models **10**

3.1 Motivation . . . 10
3.2 PSPNet implementation . . . 10
3.3 Implementation analysis . . . 13
3.4 Conclusion . . . 18

IV. Aerial Image Segmentation **20**

4.1 Motivation . . . 20
4.2 Related work . . . 20
4.3 Thesis goals . . . 21
4.4 Deep Fully Residual Convolutional Neural Network (DFRCN) . . . 21
4.4.1 Model structure . . . 21
4.4.2 Preprocessing and sampling strategy . . . 22
4.4.3 The choice of loss function . . . 22
4.4.4 Implementation . . . 23
4.5 Evaluation metrics . . . 24
4.5.1 Relaxed precision/recall . . . 24
4.5.2 Mean Intersection over Union (mIoU) . . . 25

V. Experiments **26**

5.1 Introduction . . . 26
5.2 Experimental Results . . . 26
5.2.1 The choice of loss function . . . 26
5.2.2 Deep supervision . . . 28
5.2.3 Batch size . . . 28
5.2.4 Decoder feature map upsampling . . . 30

5.2.5 Output stride 31
5.2.6 Test time augmentation 33
5.2.7 Comparison with other models 34

VI. Conclusion **36**

Acknowledgements **37**

List of Figures

2.1 An example of input and output in a dataset for semantic segmentation task. Source: Pascal VOC dataset. 4
2.2 All the items on the dining table has the label of dining table. Source: (Pascal VOC dataset [1]). 4
2.3 U-Net architecture. Source: U-Net paper [2] 5
2.4 Deconvolution Network architecture. Source: Learning Deconvolution Network for Semantic Segmentation [3] 5
2.5 RefineNet model. Source: [4] 8
2.6 PSPNet model. Source: [5] 8
2.7 DeepLab-v3 model. Source: Rethinking Atrous Convolution for Semantic Image Segmentation. [6] 9

3.1 Our baseline implementation of PSPNet[5] performance on Pascal VOC validation set. First column is input image, second column is groundtruth, third column is model's prediction and fourth column is original PSPNet result. Refer to table 3.2 first row. . . . 12
3.2 Freezing batch normalization layer parameters of the baseline model. First column is input image, second column is groundtruth, third column is model's prediction and fourth column is original PSPNet result. Refer to table 3.2 second row. 13
3.3 Freezing batch normalization layer parameters of the ImageNet ResNet-101. First column is input image, second column is groundtruth, third column is model's prediction and fourth column is original PSPNet result. Refer to table 3.2 third row. 14
3.4 Our baseline multi-GPU implementation of PSPNet. First column is input image, second column is groundtruth, third column is model's prediction and fourth column is original PSPNet result. Refer to table 3.3 first row. 15
3.5 Our multi-GPU implementation of PSPNet with frozen running statistics of batch normalization layer. First column is input image, second column is groundtruth, third column is model's prediction and fourth column is original PSPNet result. Refer to table 3.3 second row. 16
3.6 Our multi-GPU implementation of PSPNet with fixed batch normalization layer. First column is input image, second column is groundtruth, third column is model's prediction and fourth column is original PSPNet result. Refer to table 3.3 third row. 17
3.7 PSPNet performance breakdown on ADE-20K dataset. Source: It is taken from PSPNet presentation slides[5]. 19

4.1 An example of road data provided by [7]. In some of the images in the dataset, aerial image is incomplete but the corresponding groundtruth has the complete annotation. . . . 23
4.2 Our novel deep fully residual convolutional neural networks (DFRCN). 24

5.1 Samples from training data of Massachusetts road dataset. (a) Input image; (b) Groundtruth. 27
5.2 Example results on the validation set of Massachusetts road dataset. (a) Using MSE loss; (b) Using dice loss; (c) Using weighted MSE and dice loss; (d) Ground truth; (e) Input image. 29
5.3 Example results on the validation set of Massachusetts road dataset. (a) Without deep supervision; (b) With deep supervision; (c) Ground truth. 30
5.4 The schematic of our decoder module. 32
5.5 The difference of transposed convolution and bilinear interpolation. Left: Input feature map. Middle: Input to transposed convolution. Right: Bilinear upsampled input feature map. 32
5.6 Checkerboard artifacts in the prediction of a decoder with transposed convolution. Left: The result of a model with a decoder that has transposed convolution as an upsampling layer, Right: The result of a model with Bilinear up-convolution (Bilinear interpolation with a convolution 33
5.7 Example results on the validation set of Massachusetts road dataset. (a) Output stride 32; (b) Output stride 8; (c) Output stride 4. 34
5.8 Test time augmentation for final prediction. 35

List of Tables

3.1 Semantic segmentation models performance and source code availability 10
3.2 Our PSPNet implementation performance Part1 11
3.3 Our PSPNet implementation performance Part2. In above table OS indicates output stride. 11
3.4 The effect of cropsize on model's performance in multi-GPU settings 13
3.5 Our PSPNet implementation performance summary 19

5.1 The effect of loss function choice on the performance of model 28
5.2 The effect of deep supervision (e.g. auxiliary loss) on the performance of model 29
5.3 The effect of batch size on the performance of model 30
5.4 Decoder upsampling choice effect on the final performance of the road extraction model 31
5.5 The effect of output stride on the performance of model 33
5.6 The effect of test time augmentation on the performance of model 34
5.7 Comparison of the road extraction models in terms of break-even point. 35

List of Abbreviations

ASPP Atrous Spatial Pyramid Pooling. 7

CNN Convolutional Neural Network. 1, 3

CRF Conditional Random Field. 21

DFRCN Deep Fully Residual Convolutional Neural Network. 1–3, 21, 24, 26, 36

DSC Dice Similiarity Coefficient. 26

FCN Fully Convolutional Network. 1, 3, 36

IOU Intersection over Union. 25, 30, 33, 34

PSPNet Pyramid Scene Parsing Network. 1, 6, 7, 10, 18, 30, 36

RBM Restricted Boltzmann Machine. 20

Chapter I

Introduction

One main end goal of Computer Vision is to enable computers to have a "high-level" understanding of the visual information. Semantic image segmentation is one of challenges computer vision needs to solve to achieve this ultimate goal. The aim of semantic segmentation is to assign a label to each pixel of an image. Although there have been much efforts to address this problem, but computer vision models have not been able to beat human performance.

Semantic segmentation has various applications in diverse domains including satellite image analysis, autonomous driving, and medical diagnosis. To be more precise, researchers are interested to automatically extract roadmaps [7] from aerial images. As for the medical imaging, segmenting the nodules in lung cancer computed tomography (CT) scans [8] or lesions in skin cancer [9] images are of high importance for diagnostic purposes. Recently, semantic segmentation emerged as a tool to investigate the interpretability of deep Convolutional Neural Networks (CNNs) [10].

Humans are great at parsing an image to it's component by just looking at them very briefly. In a glance, humans can extract a great deal of information from an image and segment virtually all the components on it. Also, humans are able to easily generalize from seeing a set of objects to recognize them in previously unseen settings. To name a few challenges computers have to overcome to solve this problem, one can point to the inherent diversity in scenes and the variability of labels that can be associated with each scene. A good vision system should be robust to variations in appearance, viewpoint, pose, illumination.

With the big breakthrough of Convolutional Neural Networks in 2012 for object recognition [11], it sets a new standard for feature-based learning approaches. Prior to that, the state-of-the-arts object recognition models relied heavily on hand-crafted features such as Scale Invariant Feature Transformation (SIFT) [12], Histogram of Oriented Gradients (HOG) [13], Gabor and Haar wavelets [14].

After 2012, it was shown practically that deeper models can achieve better performance [15]. However, due to some problems in training e.g. vanishing gradients, fitting deeper models (e.g more than 30 layers) was not possible. In 2015, ResNet [16] model resolves many of those issue with residual connections and successfully trained a 1000 layer model and made a new state-of-the-art performance benchmark on object recognition task.

After the success of convolutional networks on object recognition, computer vision researchers tried to extend the model to the other task in computer vision. Region-based Convolutional networks (RCNN) [17] used CNN for classifying the bounding box proposals acquired by selective search algorithm [18] for object detection. Furthermore, Fully Convolutional Neural Networks (FCNs) [19] modified the convolutional neural network of AlexNet to the "fully" convolutional neural network by reinterpreting the last two fully connected layers as a convolutional layer with kernel size that spans over all the input. Most of the models proposed for semantic segmentation in the years after are merely a variant of a FCN model.

In this work, we present our novel model, Deep Fully Residual Convolutional Neural Networks (DFRCNs) for semantic segmentation. Deep models (more than 30 layers) tend to have better performance, however, they may suffer from degradation problem [16]. With our residual learning process, we successfully trained a model with more than 50 layers and achieved better performance. Furthermore, in our decoder module we used simple bilinear interpolation instead of widely used transposed convolution (a.k.a. deconvolution [20]) to upsample feature maps. We got better performance and faster model since it requires fewer parameters. We explain the model in the next chapters and show its performance on road segmentation of aerial images. We also tried to clarify some techniques regardless of model structure (e.g. batch normalization [21] parameter tuning) for getting a state-of-the-art performance on semantic segmentation benchmarks.

Chapter II

Large Scale Computer Vision

2.1 Introduction

The main focus of this thesis is utilizing deep learning algorithms for large scale semantic image segmentation. Therefore, we provided a through review of current deep learning algorithms which have been proposed for understanding visual information. Primarily, we heavily focused on a variation of deep learning models called convolutional neural networks. Currently, it is believed to be the de-facto model to tackle the large-scale machine vision problems.

2.2 ImageNet Classification with Deep Convolutional Neural Networks

Since 2010, a competition has been held annually for processing large scale visual data which is called ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [22]. In 2012, [11] proposed a method based on CNNs that surpassed the previous year winning entry by a large margin (16% error rate as opposed to 25% error rate). This sparked a huge interest in CNNs and lead the other researchers working on the other various large scale vision field such as object detection, semantic segmentation to think about adapting CNNs to their field.

2.3 Large Scale Semantic Segmentation

The aim of image semantic segmentation is to segment pixels of an image according to a pre-defined meaningful concept. This is relied heavily on the provided definition of concepts. One choice is to segments images to “objects”. However, “objects” or “things [23]” or ambiguous. As an example all the objects on the dining table in Figure 2.2 are labeled as dining table or humans in a monitor won’t be regarded as person category.

Before deep learning became a de-facto algorithm for feature extraction on many large scale vision problems including semantic segmentation, Random forest based feature extractors [24] were commonly used for semantic segmentation. One early neural network approach was patch classification [25] which was used to segment the neuronal membranes in Electron Microscopy (EM) images. At the core of that model, was a neural-network-based classifier which predicts a label for each patch given its neighbor patches. One drawback of the model was it could only handle fixed size images since it had two fully connected layer at the back-end. In 2014, [19] proposed FCNs model to resolve this issue. They reinterpreted the last fully connected layers as a convolution layer with kernel size as the input size. Furthermore, as a result of pooling layers the resulted activation map has a smaller spatial size than the input image. They introduced transposed convolution [20] (a.k.a deconvolution) for upsampling coarse fea-

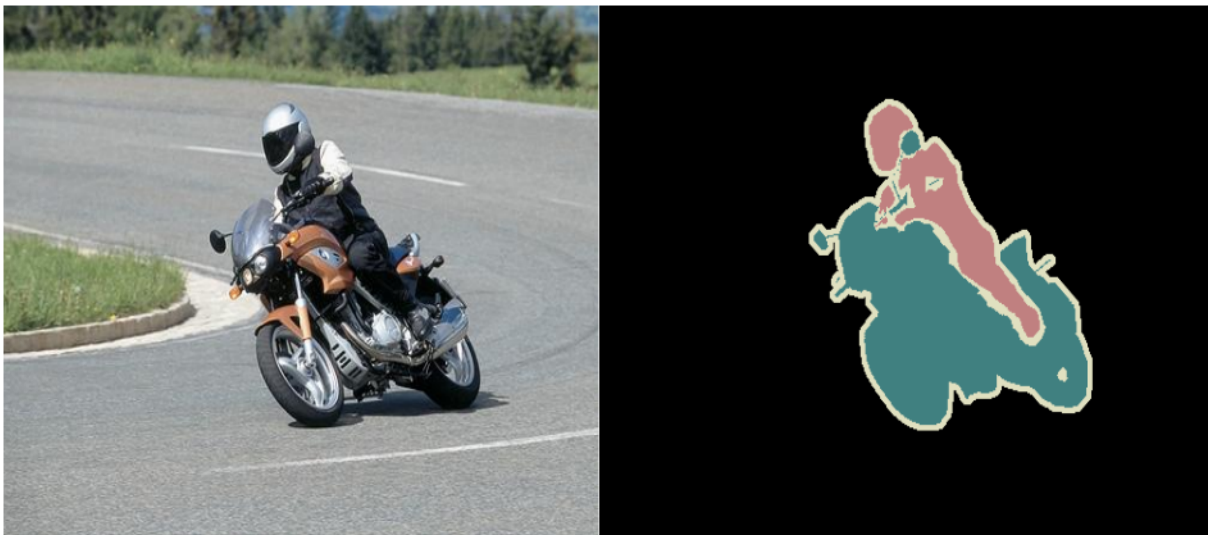
Figure 2.1: An example of input and output in a dataset for semantic segmentation task. Source: Pascal VOC dataset.

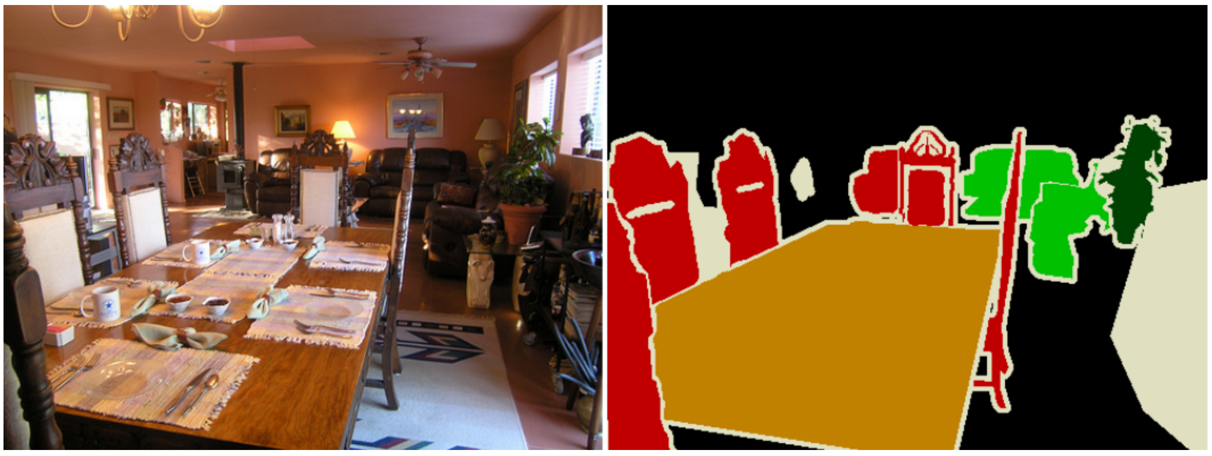
Figure 2.2: All the items on the dining table has the label of dining table. Source: (Pascal VOC dataset [1]).

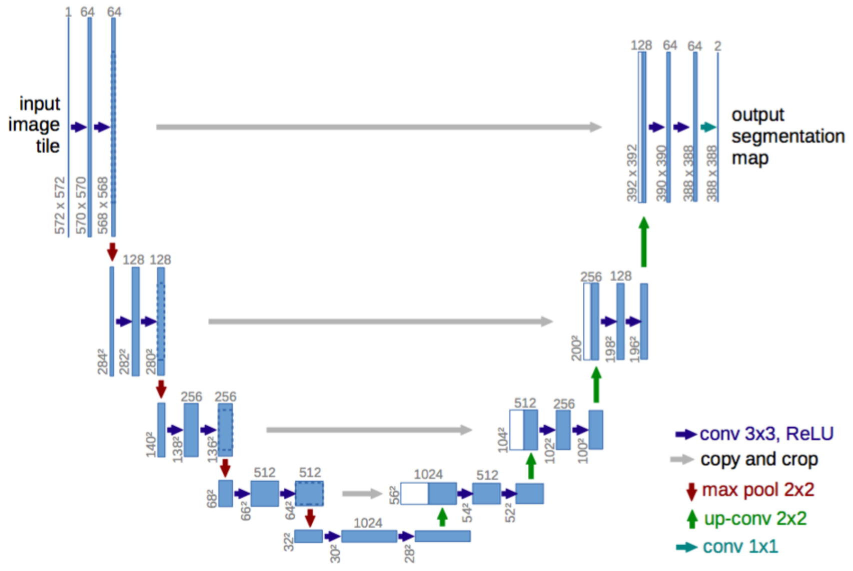

Figure 2.3: U-Net architecture. Source: U-Net paper [2]

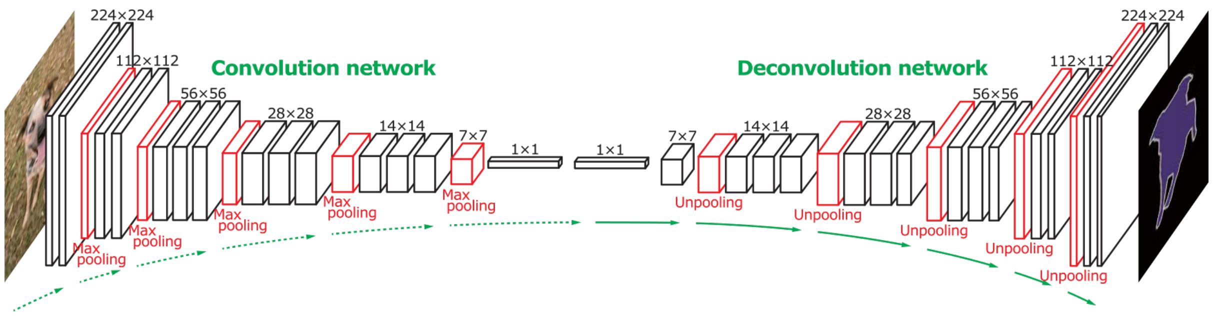

Figure 2.4: Deconvolution Network architecture. Source: Learning Deconvolution Network for Semantic Segmentation [3]

ture maps. These modification enabled them to obtain a fully convolutional network that can efficiently transfer the learned features during image classification to be used for dense prediction. FCN model became the building block of almost every dense prediction task. Still a main challenge remained: how to recover the spatial features after the sub-sampling (e.g. pooling) operation. In other words, how to aggregate both spatial features (e.g. local, pixel-level features like precise detection of edges) and global features (e.g. dynamics of a scene) was unclear. In the coming years, people have come up with two solutions: 1) Using encoder-decoder module [2][3] for training 2) Using Dilated Convolution [26][27]. For the first case, encoder module reduce the spatial size after each pooling operation which enable the network to increase the depth while decoder recovers the location-wise features from coarse output of encoder module. Furthermore, skip connections from encoder to decoder ease the training process by preserving the gradients from being vanished. Famous examples of encoder-decoder model are U-Net depicted in Figure2.3 and Deconvolutional network in Figure2.4. Dilated convolution (aka atrous convolution, hole convolution) which was first used in 1D signal processing is extended to semantic image segmentation by DeepLab-v1 [26]. The algorithm provided the benefit of computing the responses of any layer with desirable resolution. One other important feature of dilated convolution is that it can be used both post-hoc (after the image classification training has been done) or it can also be seamlessly incorporated while training.

Dilated means hole in kernel of the convolution. It will increase receptive field exponentially for capturing global context. Compare to vanilla convolution plus pooling (which are the building blocks for encoder-decoder module), dilated convolution make it possible to expand receptive field size exponentially with fewer parameters.

If the dilation rate parameter is equal to one, dilated convolution is a regular 2D convolution. If the rate is greater than one, it performs convolution with holes, convolving with the input values every rate pixels in both height and width dimensions. The other way to look at dilated convolution is a convolution which the kernel that has an upsampled filters, produced by inserting rate-1 zeros between two consecutive elements of the filters in both height and width dimensions. Both dilated network [27] and DeepLab-v1 [26] built a model based on cascaded dilated convolution. To go deeper on the dilated network, they disregarded last two pooling layers of a pretrained image classification network (VGG network [15]) and the subsequent convolutional layers between third pooling layer and fourth one have dilation rate 2 and convolution layers after fourth pooling layer have dilation rate of 4. With this settings they were able to fine-tune an image classification network for semantic segmentation task without introducing any additional parameters. Furthermore, they incorporated a separate module called context module which consist of multiple cascaded dilated convolution with different rate parameters. Through utilizing this additional module they were able to produce aggregated multi scale output and hence improve the performance.

2.4 RefineNet (downside of dilated convolution)

Although dilated convolution allows exponential growth of effective receptive field size with only linear growth in number of parameters, it is computationally expensive. That is it takes a lot of memory since they have to be applied on a large (in terms of spatial size) feature maps. RefineNet [4] proposed an spatial encoder-decoder module to address aggregation of global context while being memory efficient. The encoder of RefineNet is ResNet-101 [16] image classification network. The decoder module has RefineNet blocks which gradually fuse dense feature maps of encoder and upsampled coarse output of previous RefineNet block together. The architecture of RefineNet model can be viewed in Figure2.5.

2.5 Pyramid Scene Parsing Network (PSPNet)

In 2016, PSPNet [5] devised a model based on pyramid feature pooling [28] to capture global context. They modified ResNet-101 with all the convolution layers of third block to have dilation rate 2 and all the convolution layers of fourth block to have dilation rate 4, this would produce an output of size 1/8 of the input size. The input to the pyramid pooling module is the output of dilated ResNet-101 network. Pyramid pooling module consist of multiple pooling operation with large kernels (e.g 10, 20, 30, 60).

To be precise, the kernels of pooling layer has the size of whole, 1/2, 1/3 and 1/6 of the input feature map. The aim of such pyramid pooling is to capture both multi-scale features and also global context. They concatenate the output of dilated ResNet-101 with the output of pyramid pooling module for final prediction. The schematic of their model is provided in Figure 2.6.

While the idea of pyramid pooling seems simple it gives a noticeable boost (3%) to the final prediction [5]. However, one important component of PSPNet is training of batch normalization parameters. Training with a large batch size improve the generalizability of a model which leads to a better performance. However, due to the nature of dense prediction task (e.g. semantic segmentation and object detection), training with a large batch size (usually more than 16) requires a large amount of GPU memory. To overcome this issue, PSPNet use a modified version of caffe [29] deep learning library which enables them to synchronize batch norm parameters across multiple GPU instances. They trained their model using 8 GPU devices with 2 batch of images with crop size of 473 on each device which makes a total of 16 batches. Without using synchronized batch normalization, the performance of their model dropped almost 4% compared to using synchronized batch normalization. In chapter 3, we tried to show the effect of batch normalization for the performance of semantic segmentation model.

2.6 Rethinking Atrous Convolution (DeepLab-v3)

In 2017, DeepLab-v3 [6] model proposed a similar model as PSPNet which exhibits roughly the same performance based on Atrous Spatial Pyramid Pooling (ASPP). Similar to PSPNet, the output of encoder (either ResNet-101 or ResNet-50) is concatenated with the output of ASPP module. In ASPP module, there are one 1×1 convolution, three dilated convolution and with different large dilation rate (e.g. 6, 12, 18) and a global average pooling (image-level feature) operation followed by spatial upsampling to the desired size. However, they took a different training scheme as PSPNet. Their training can be viewed as two stages. In the first stage, in order to be able to fit a large batch size for training, they manipulated dilation rate of ResNet network to have an output stride (the spatial ratio of input image of ResNet to its output) of 16. Note that the output stride in PSPNet was kept at 8. They also used a crop size of 513 for images with batch size 16. Unlike PSPNet they didn't use sync batch normalization, however, they used a very large momentum (0.9997) for tuning batch normalization parameters (see Equation 2.2). It is worth to note that batch normalization layer has different behavior in training and testing. In training, the normalization is done based on each batch statistics (e.g. mean and variance). Also during the training, there are two parameters namely global running mean and running variance which gets updated in exponential moving average scheme with a momentum parameter (see equation 2.1).

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \varepsilon}} \tag{2.1}$$

$$\hat{x}_{new} = (1 - \text{momentum}) \times x_t + \text{momemtum} \times \hat{x} \tag{2.2}$$

In equation 2.1, $\hat{x}_i$ is the input x_i after normalization with μ and σ as the mean and variance respectively computed over the batch dimension. In equation 2.2 which shows the exponential moving statistics

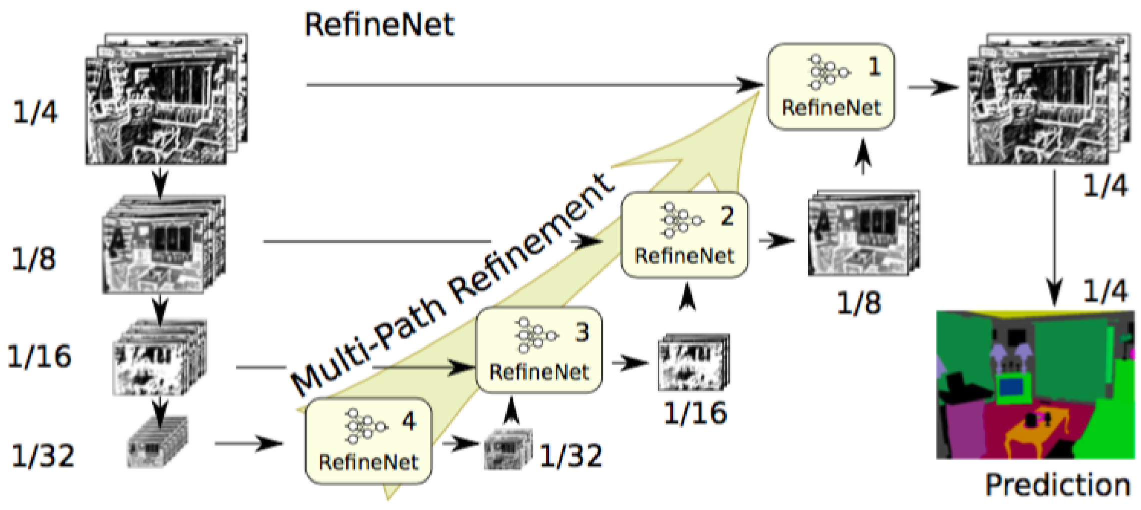

Figure 2.5: RefineNet model. Source: [4]

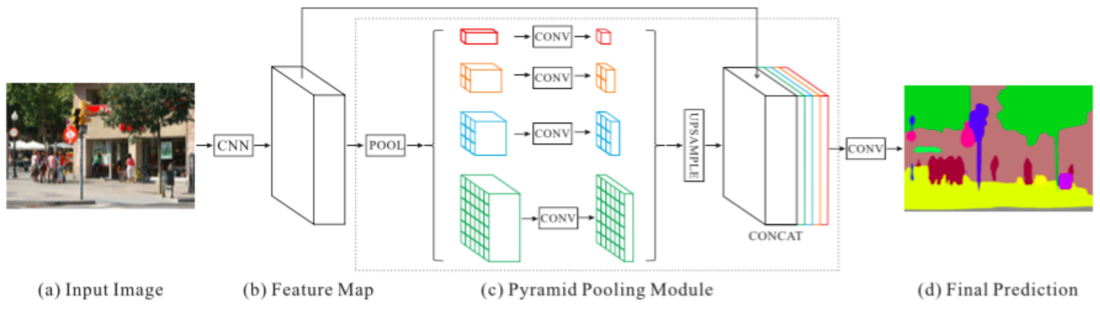

Figure 2.6: PSPNet model. Source: [5]

update rule, x_t is the new observed value and $\hat{x}$ is the computed global statistics for previous iterations.

These global statistics then were used during testing instead of each batch (or image) for normalization. However, if the momentum parameter is chosen to be close to one, it means that each training batch statistics would have very much little effect on the global statistics which ultimately could be thought as freezing the global mean and variance. In the second stage for training deeplab, they decrease the learning rate (from 0.007 to 0.001), increase the output stride to 8 and freeze the batch normalization parameters and finetune on both training and validation set. They argued that training with output stride 16 is quite faster than output stride 8 and also more memory efficient which is useful for tuning batch normalization parameters. However it is not as accurate because it uses coarser feature maps. That performance will get better by stage two's fine tuning with using output stride 8. Figure 2.7 shows DeepLabv-3 architecture.

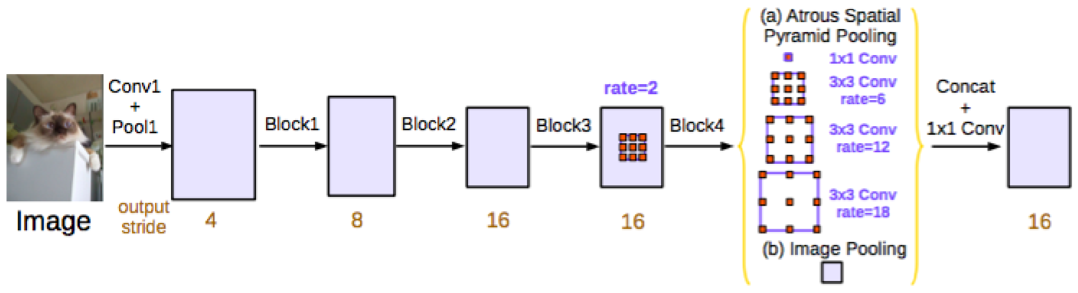

Figure 2.7: DeepLab-v3 model. Source: Rethinking Atrous Convolution for Semantic Image Segmentation. [6]

Chapter III

Implementation of state-of-the-art models

3.1 Motivation

In chapter 2, we covered some of the models that exhibits state of the art performance on the Pascal VOC leaderboard. However, most of the best performing models do not have publicly available source-codes to replicate the training process (refer to Table 3.1).

Model Name	Institution	VOC score	Train Code	Eval Code	Private Dataset
DeepLab-v3 [6]	Google	86.9	O	O	JFT-300
PSPNet [5]	CUHK, SenseTime	85.4	X	O	X
SDN [30]	CASIA	86.6	X	X	X
DIS [31]	SYSU, CHUK	86.8	X	X	IDW

Table 3.1: Semantic segmentation models performance and source code availability

As it was summarized in the table, some method like DeepLab-v3 and Deep Dual Image Segmentation (DIS) [31] model use a private dataset to train or finetune their model. Some other models like PSPNet did not share their training code and it made others unable to replicate their training procedures from their publicly available evaluation code. Some other models like Stacked Deconvolutional Network (SDN) [30] and DIS did not share their evaluation code let alone their training code. What follows is our efforts to implement one of these models based on the information in the paper. Furthermore, we gave our own detailed analysis on the main components of those models. During the implementation, we found that some of the important features of those models could be studied more.

3.2 PSPNet implementation

We start by implementing PSPNet in pytorch [32] deep learning framework. We used NVIDIA TITAN X GPU with 12GB of memory for training. The original implementation was done in modified caffe deep learning framework and run on 8 GPU devices each with 12 GB of memory. For our own initial experiment, since it was not possible to fit 16 image batches of cropsize 473×473 on a 12GB GPU device, we chose batch size 8 of cropsize 340×340 for training. To compensate the smaller batch size, we decreased the learning rate by the factor $\frac{1}{\sqrt{2}}$ (as suggested by [33]) and also increased the training iteration from 30K to 100K iterations. This leads to the performance of 71.42% on validation set which is about 10% lower than the original reported performance of PSPNet. Unlike the PSPNet, we did not do multi-scale testing which helps boost performance about 1% (refer to Figure 3.7). Furthermore, the PSPNet with ResNet-101 model as the backbone is a modified form of the original ResNet-101 model in a sense that they replace the first 7x7 kernel size convolution layer with three 3x3 kernel size convolution layer. It is noted by others that this type of modified ResNet has a better performance than the original ResNet model [34].

Inspired by DeepLab-v3 training of batch norm parameters, we froze the parameters of trained model and finetune the model for about 30K more iteration, this increased the performance about 1% compare to the baseline model. However, if we froze the batch norm parameters from the initial ResNet model (pretrained on the ImageNet dataset only), we would get about 2.2% performance boost. We suspect that training with small batch size on Pascal VOC dataset may damage the inference performance. We would like to pinpoint that ResNet-101 model batch normalization layer's global statistics (mean and variance) were computed over a "sufficiently large training batch after the training procedure" as stated by the author. This may be the reason for the better performance of a model that it's batch normalization layer parameters were frozen from the initial ResNet model pretrained on ImageNet compare to the one that was finetuned on Pascal VOC dataset for some training iterations.

No. GPUs	Batch size	Crop size	No. iterations	BN freezing	VOC score
1	8	340	100K	N (baseline)	71.42
1	8	340	100K	Y (from baseline)	72.60
1	8	340	100K	Y (from ImageNet ResNet)	73.64

Table 3.2: Our PSPNet implementation performance Part1

We also tried to increase the batch size and crop size and train on multiple GPUs. So we tried with batch size 16 on 4 GPUs (4 on each GPU) with crop size 473×473. We trained the new model for 30K iterations. We got 72.5% performance on the validation set. We also tried to use larger output stride (16 instead of 8) for the ResNet101 model. This made us enable to use 32 batches for training. However, we purposefully did not update the running mean/variance of the batch normalization layers. This got 74.94% performance on the validation set which is 2.44% boost relative to the previous model. Also we fine tuned this model by using output stride 8, however, with smaller batch size (16) and frozen batch normalization for additional 30K iterations. The resulted performance achieved was 76.41%. Our PSPNet implementation details is available at the table 3.2 and 3.3. In order to show the effect of cropsize in multi-GPU settings, we made another experiment with keeping every parameters expect cropsize same with the baseline model. Using larger cropsize will enhance the performance by 1.2% (table 3.4).

No. GPUs	Batch size	Crop size	No. iterations	BN freezing	VOC score
4	16	473	30K	N (baseline)	72.51
4	32 (OS 16)	473	30K	Y (global stats only)	74.94
4	16 (OS 8)	473	30K	Y (fixed affine transformation)	76.41

Table 3.3: Our PSPNet implementation performance Part2. In above table OS indicates output stride.

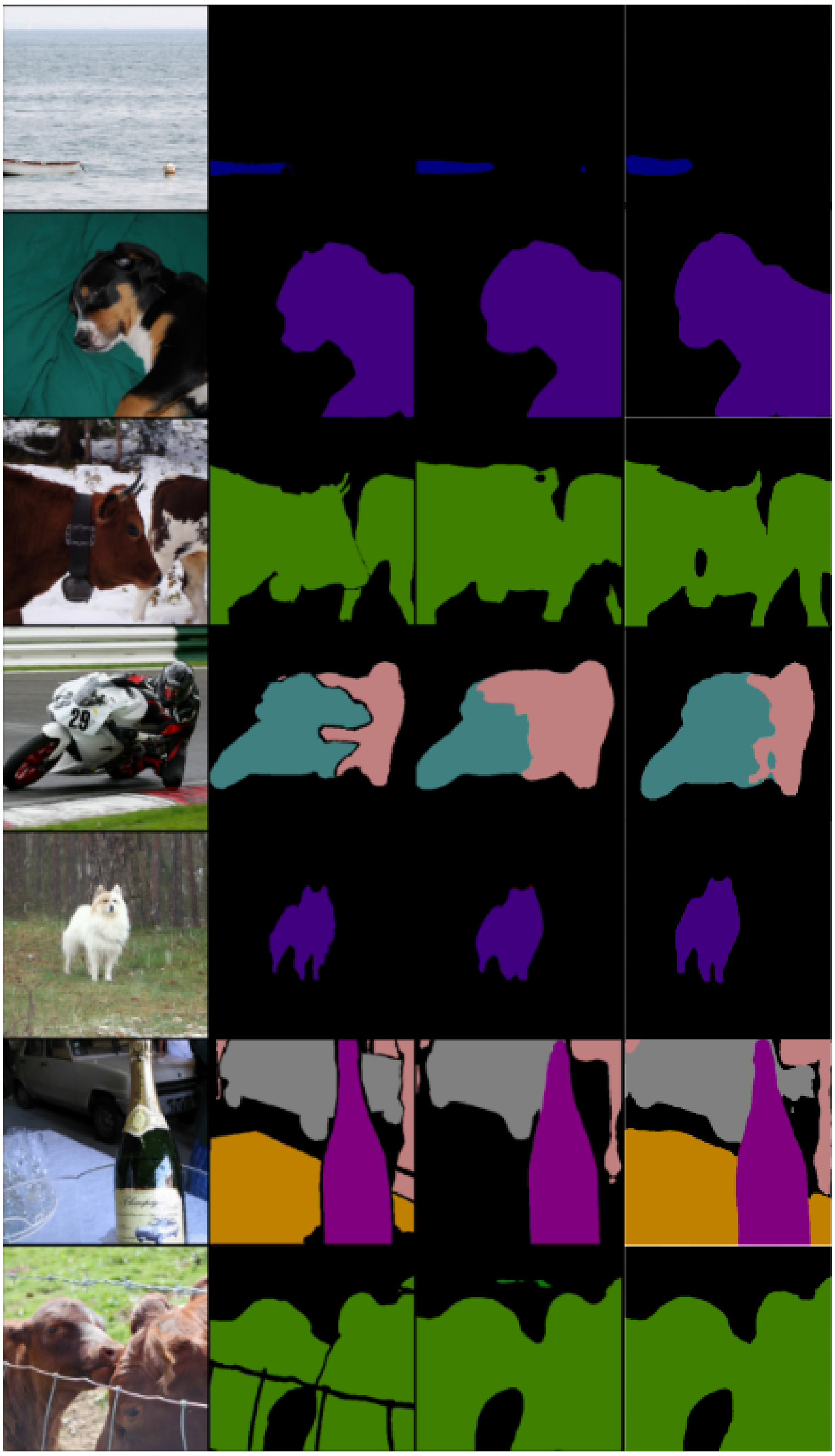

Figure 3.1: Our baseline implementation of PSPNet[5] performance on Pascal VOC validation set. First column is input image, second column is groundtruth, third column is model's prediction and fourth column is original PSPNet result. Refer to table 3.2 first row.

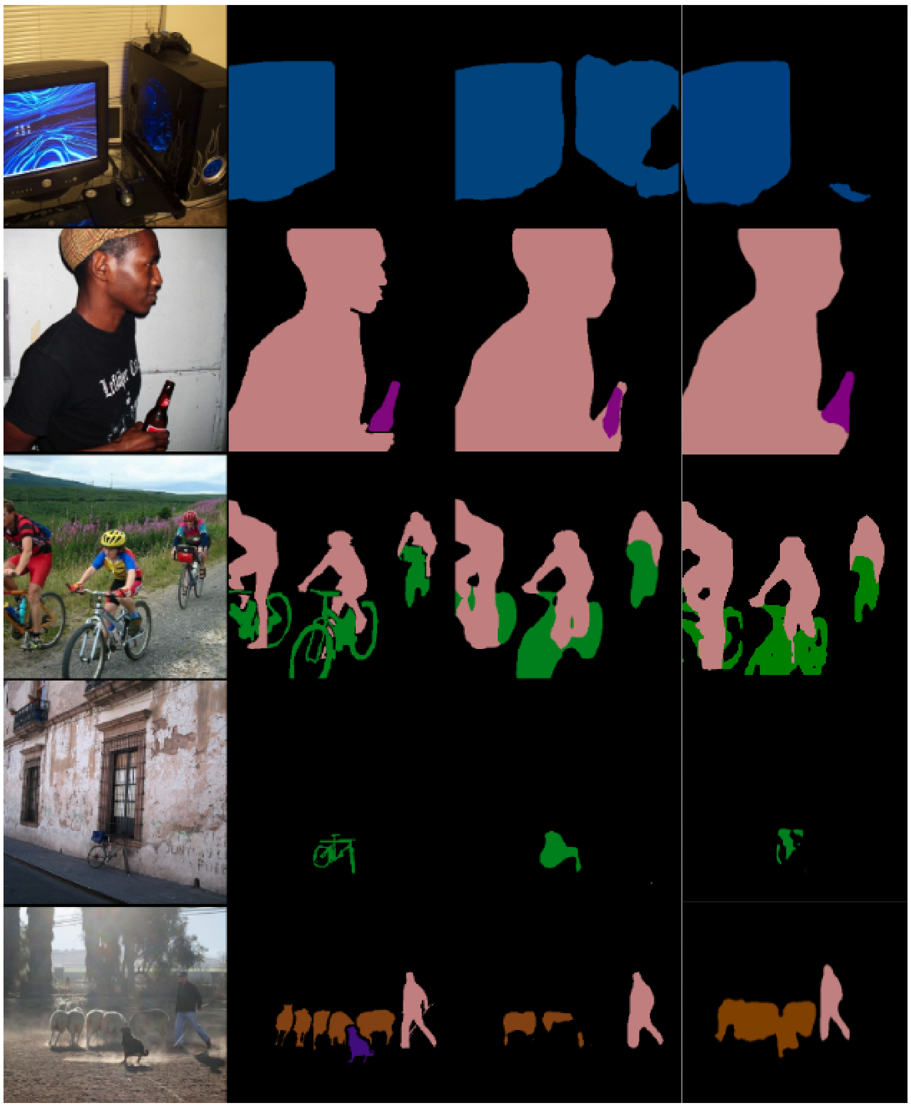

Figure 3.2: Freezing batch normalization layer parameters of the baseline model. First column is input image, second column is groundtruth, third column is model's prediction and fourth column is original PSPNet result. Refer to table 3.2 second row.

No. GPUs	Batch size	Crop size	No. iterations	BN freezing	VOC score
4	16	340	30K	N	71.34
4	16	473	30K	N	72.51

Table 3.4: The effect of cropsize on model's performance in multi-GPU settings

3.3 Implementation analysis

Normalization plays a critical role in order to have a good performance on a large scale vision benchmark [35][36]. Since batch normalization proposed, it enhanced the object classification accuracy models. To effectively train batch normalization parameters, however, the size of mini-batch should be large enough or it may not perform well because of behavior difference in training time and inference time.

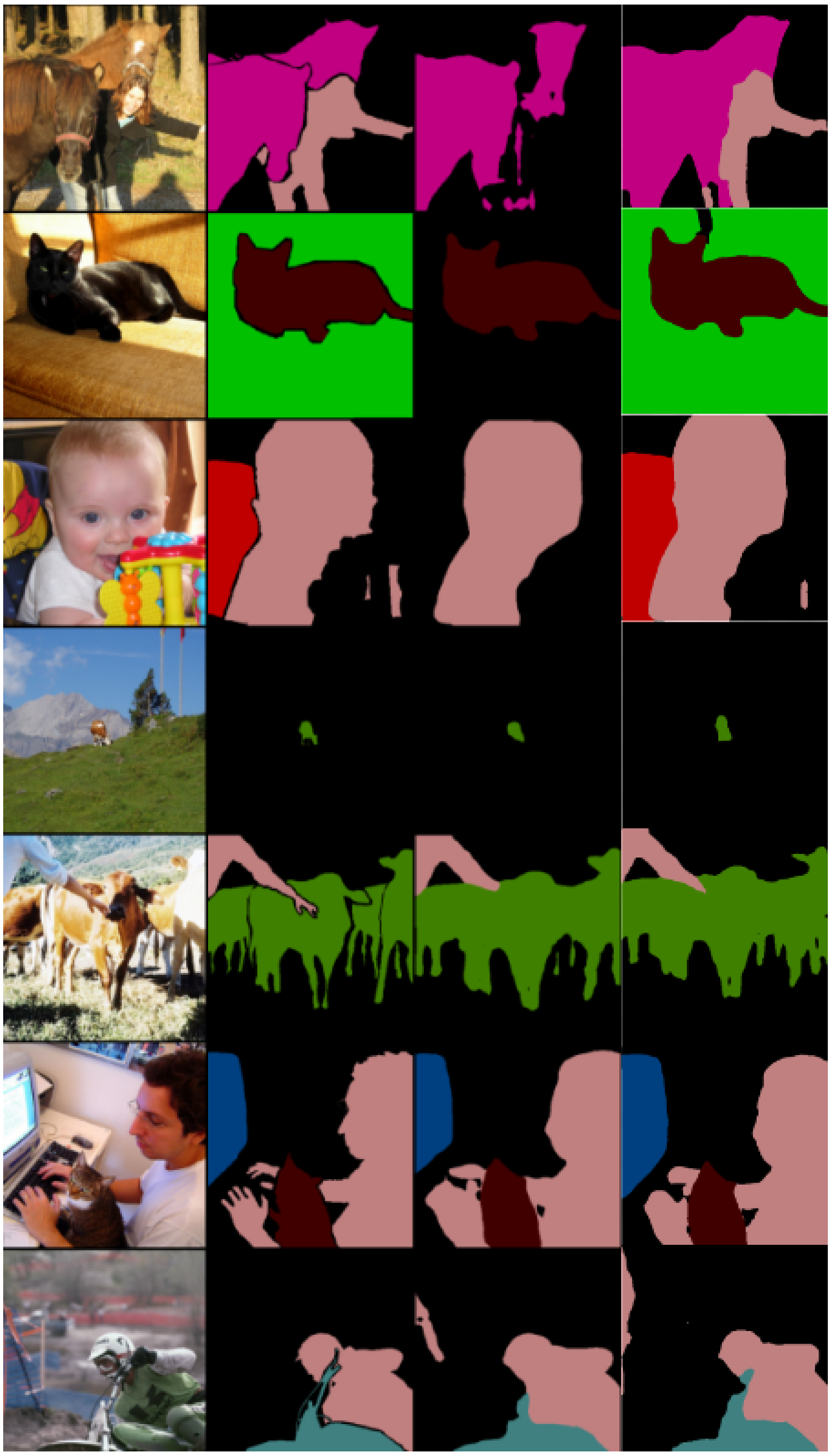
Figure 3.3: Freezing batch normalization layer parameters of the ImageNet ResNet-101. First column is input image, second column is groundtruth, third column is model's prediction and fourth column is original PSPNet result. Refer to table 3.2 third row.

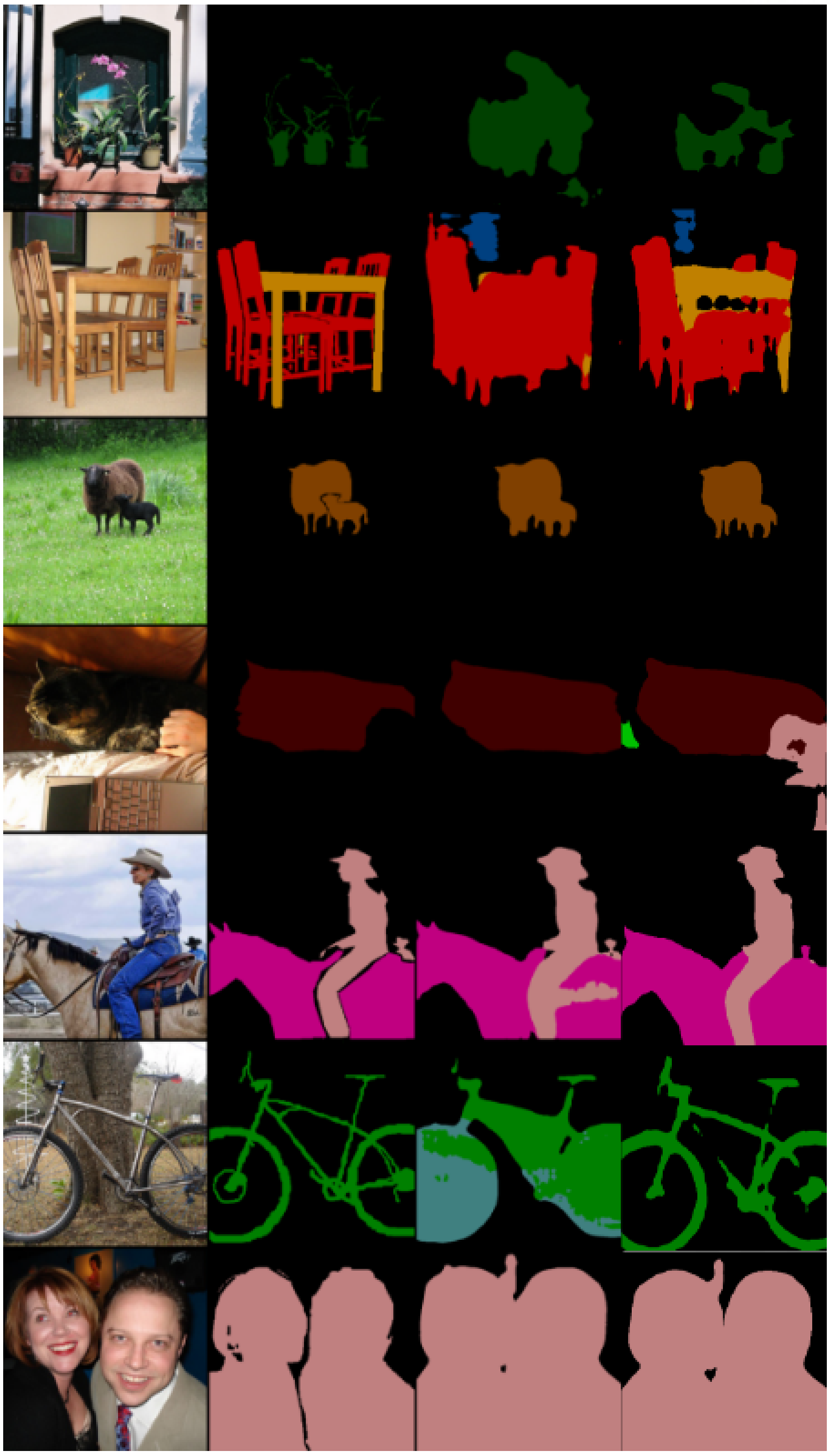

Figure 3.4: Our baseline multi-GPU implementation of PSPNet. First column is input image, second column is groundtruth, third column is model's prediction and fourth column is original PSPNet result. Refer to table 3.3 first row.

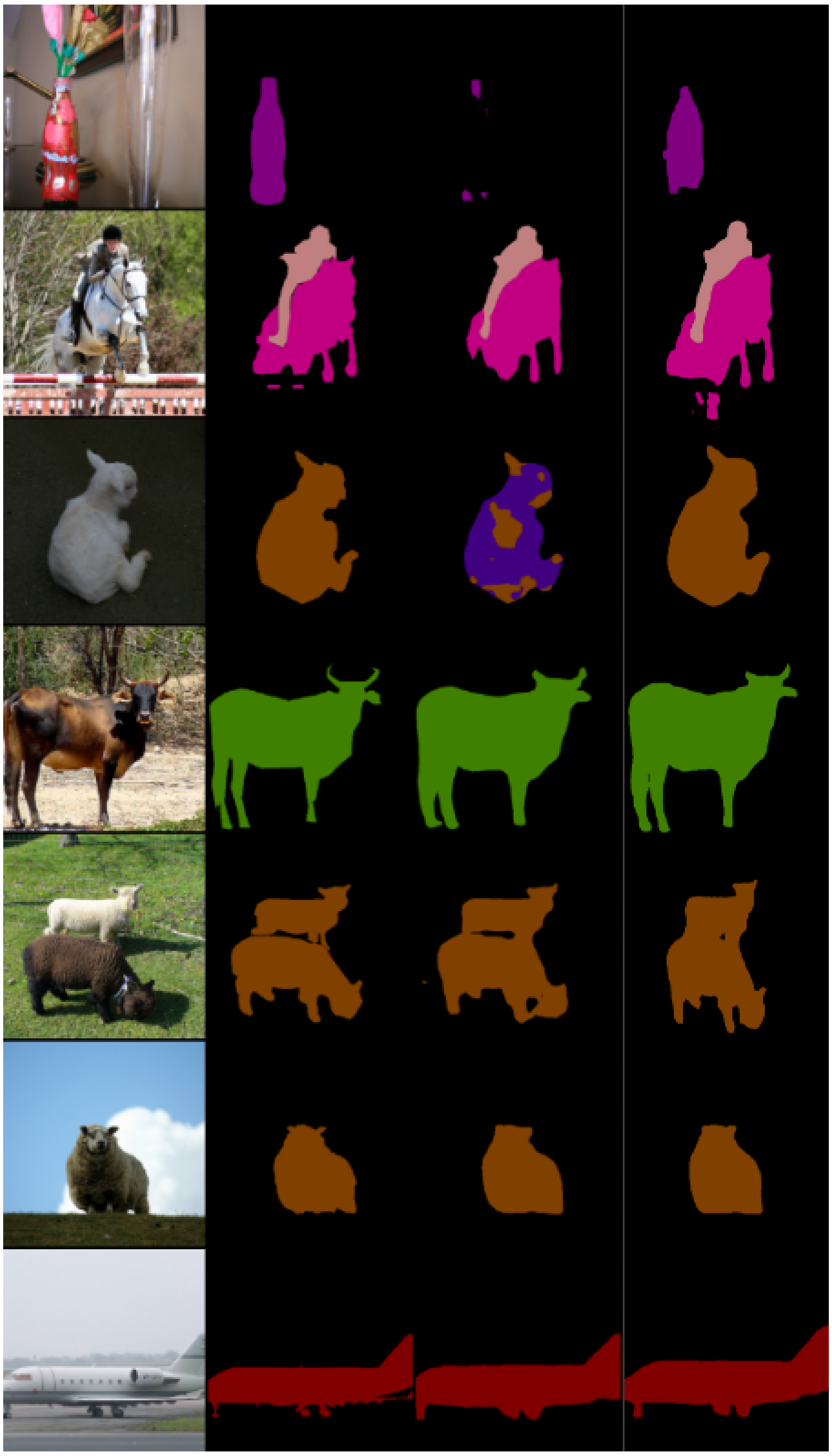
Figure 3.5: Our multi-GPU implementation of PSPNet with frozen running statistics of batch normalization layer. First column is input image, second column is groundtruth, third column is model's prediction and fourth column is original PSPNet result. Refer to table 3.3 second row.

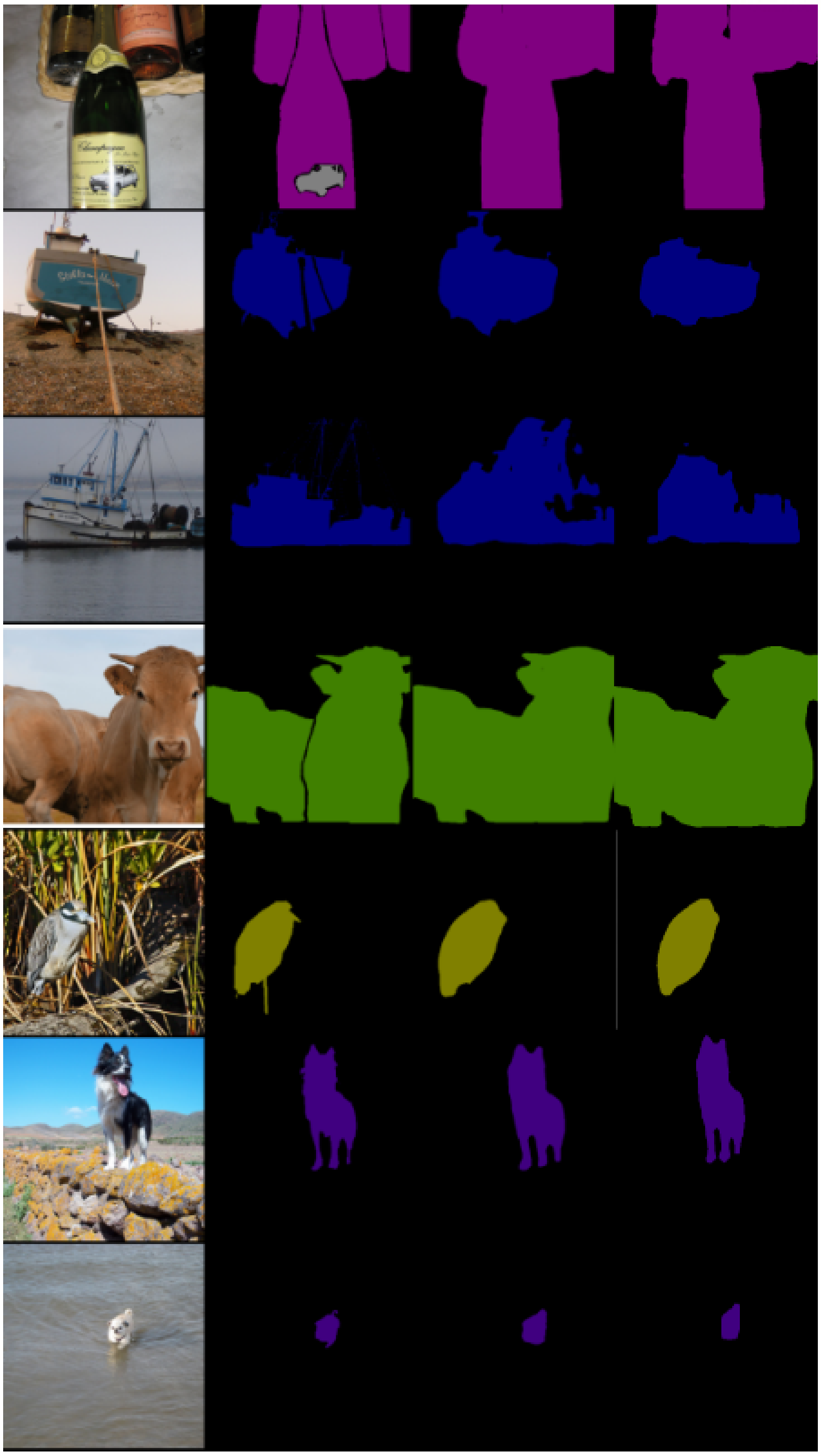

Figure 3.6: Our multi-GPU implementation of PSPNet with fixed batch normalization layer. First column is input image, second column is groundtruth, third column is model's prediction and fourth column is original PSPNet result. Refer to table 3.3 third row.

Unfortunately, for object detection and semantic segmentation since output has the same size as input, it is hard to fit large mini-batch on limited GPU memory. As we observed with implementation of multi-GPU PSPNet model and it was noted by [35] also, the number of min-batches on each GPU is crucial to the overall performance of the model. The change in mean/variance statistics computed by batch normalization with different number of mini-batch in each GPU will introduce different levels of random variation. PSPNet tried to resolve this issue by synchronizing the batch normalization across multi GPUs. Precisely, in each iteration, the mean of each mini-batch on all GPU workers were broadcasted to the master GPU and it then computes the global mean. After that the global mean were broadcasted to each worker to compute the variance with regard to global mean. After each worker computed the variance, all get broadcasted to master to compute the global variance. After the global variance was computed by the master, each worker can normalize their mini-batch with regard to global mean/variance rather than the mean/variance of mini-batch fitted on that specific GPU worker. In this way it would work as if a large mini-batch is used on a single worker. We would like to note that if no synchronous batch normalization is used, the performance of PSPNet drops by almost 4% (we found this via personal communication with the authors). In our experiments we noticed that crop size is also very important and can contribute to the overall performance by 1% (from 340 to 473).

We would like to mention other approaches than syncing batch normalization. As Deeplab-v3, they tend to freeze the batch norm global mean/variance and not update it for the semantic segmentation task. We also observed that freezing the batch normalization mean/variance will indeed improve the performance. However, frozen batch normalization seems like a sub-optimal solution since it requires more amount of training time as with the case of DeepLab-v3 model.

3.4 Conclusion

To summarize what we observed is having large cropsize and big batch size are important to have a better model. We also noticed that training on multi GPU (data parallel scheme) does not affect the final performance. The factor that is important in multi GPU settings is the number of batches per GPU. Furthermore, we noticed that normalization namely batch normalization is crucial to the overall performance. In order to have a better model there is need to have a large batch size. As cross GPU synchronous batch normalization was used for training the original PSPNet model to simulate a large batch size, we noticed that freezing batch normalization can improve the performance by around 4% as well (Table 3.5).

No. GPUs	Crop size	Batch size	Batch Normalization freezing	performance
1	340	8	N	71.42
4	340	16	N	71.34
4	473	16	N	72.51
4	473	16	Y	76.41

Table 3.5: Our PSPNet implementation performance summary

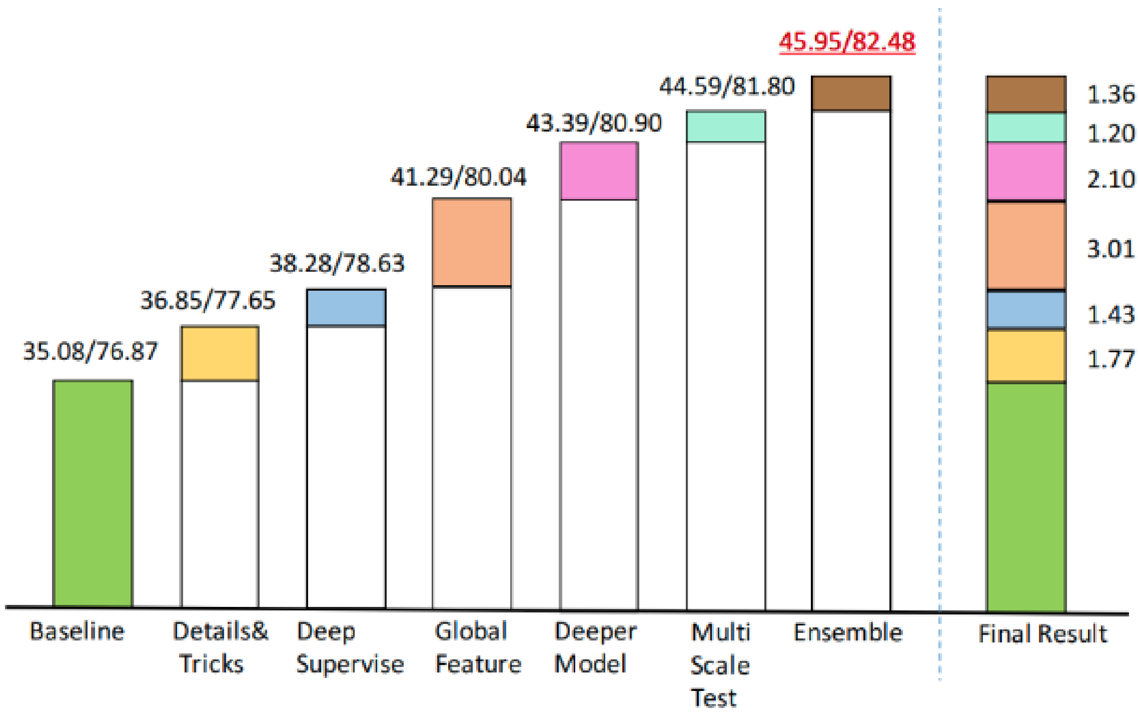

Figure 3.7: PSPNet performance breakdown on ADE-20K dataset. Source: It is taken from PSPNet presentation slides[5].

Chapter IV

Aerial Image Segmentation

4.1 Motivation

When a natural disaster strikes an area, maps and accessibility information become an invaluable source for crisis response. In case of an earthquake in a less accessible zone, road maps will ease the process of helping the people in need. Also in aftermath of such events, aerial images can be a source for approximating the damage. Take a wild fire as an example, in order to get a sense of what proportion of forest were caught by fire, damage experts compare the before and after aerial images. In this thesis, we would like to address the challenge of automatically segmenting road and street networks from aerial images. Some specific applications of this problem is road navigation maps, urban planning, autonomous vehicles and geographic information monitoring. We devised a novel approach based on deep learning to tackle this problem.

The problem consists of extracting roads from satellite images. For this problem, a set of satellite/aerial images acquired and annotated by [7], are provided. Ground-truth images where each pixel is labeled as (road, background) are also provided. The goal is to train a classifier to segment roads in these images, i.e. assign a label (road=1, background=0) to each pixel. Most of the methods proposed to solve this problems can be categorized in two groups of road area extraction and road centerline extraction. Road area extraction [7][37][38] will outputs a map that labels each pixel with road or background class. The goal of road centerline extraction [39] is to provide skeletons of a road. We would like to mention that there are also methods which extract both road areas and centerline, simultaneously [40]. We sat upon ourselves to tackle road area road extraction problem since getting road centerline from road area maps are somewhat trivial [41].

In this thesis we mainly focused on using deep learning algorithms for solving road extraction since it requires less amount of feature engineering, furthermore, the model with minor changes can effectively be used in other aerial image analysis domains like object detection and scene recognition. However, there are challenges when applying deep learning algorithms to get a robust accurate segmentation model for road extraction such as dataset class imbalance, boundary effect and low resolution of images present in the dataset which were discussed in more details in the this chapter.

4.2 Related work

One of the early deep learning algorithms suggested for aerial road extraction was made by Mnih and Hinton [7]. Their proposed model was based on Restricted Boltzmann Machines (RBMs) for patch-based semantic segmentation task. Input aerial image was partitioned into 64×64 patches. They applied pre-processing on the raw images and performing some post-processing algorithms on the output of their model. Their preprocessing consisted of applying Principal Component Analysis (PCA) to reduce data dimension. Furthermore, their post-processing was based on applying Conditional Random Field (CRF) algorithms to capture low-level pixel dependencies like road connectivity. They evaluated the performance on a large aerial image dataset that covered around 500 km^2.

Saito et al [37] devised a convolutional neural network which simultaneously segmented pixels of aerial images to buildings and roads. They trained a CNN that had a three-channel output layer for prediction of three categories (roads, buildings and background). Their model achieved a better performance than [7] while it didn't need further performance boosting techniques such as noise models and structured prediction (e.g. CRF post processing step) which were employed by Mnih and Hinton. However, unlike this model, our model is fully convolutional which enable us to use arbitrary size images for testing. We also make use of residual learning which made training deep networks feasible hence improve the accuracy.

4.3 Thesis goals

In this thesis, we proposed a novel deep fully residual convolutional neural networks for road extraction. We employed dilated convolution in our deep networks in order to control the feature extraction (e.g. encoder) part of our network. Furthermore, we effectively transferred the features learned by a deep network during general image classification task to the problem of aerial image road extraction by using a joint scheme of a naive sampling strategy and also introducing auxiliary deep supervision loss. Instead of time consuming post-processing CRF method, we incorporated a simple yet effective test time augmentation technique to enhance the network result. In this chapter, we described our model in details and reviewed its important components.

4.4 Deep Fully Residual Convolutional Neural Network (DFRCN)

4.4.1 Model structure

Our approach is based on patch-based semantic segmentation framework that is commonly used in deep learning algorithms for semantic segmentation problem. Our model employs residual learning which was proposed by [16]. Specifically, we took the ResNet-50 layers as the backbone model of our encoder module. We replaced the vanilla convolution layers of all ResNet blocks with dilated convolution. Replacing the vanilla convolution enabled us to control the output stride which is the ratio of input to the output of our encoder (e.g. ResNet-50 network). We speculated that using output stride bigger than 4 can hurt the convergence for the two following reasons:

1. The ratio of road pixels to the non-road was very small, i.e. that road parts are usually a thin lines in the whole images. If a large output stride is chosen recovering this thin road areas from a small output (more than $\frac{1}{4}$ of the input) would be extremely hard since many of the spatial features are aggregated in order to have robust depth features.

2. Using a large output stride would requires employing more parameters in the decoder module and hence more number of parameters in total which ultimately requires more data or complex regularization technique. We tried to keep the model simple which make it possible to apply our model to other aerial image segmentation problems such as extracting building, vehicles, etc.

Our decoder module follows a U-Net [2] structure. We concatenated the upsampled features with the encoder module features of second and first block plus first convolution layer of the ResNet-50 model. In upsampling the features, we used three consecutive convolution layers with kernel size of 3 followed by a bilinear interpolation layer. In our decoder module we refrain from using transposed convolution [20] cause it would increase the computation time since it performs convolution on the upsampled feature maps. It can also introduce checkerboard artifacts [42] that may heavily decrease the performance in road extraction.

4.4.2 Preprocessing and sampling strategy

The aerial image dataset [7] consists of 1171 satellite images of Massachusetts, each of them of 1500×1500 pixels. The data was partitioned into 1108 training images, 14 validation images, and 49 test images. The acquired dataset is quite noisy. There were many samples that satellite image were not complete (see Figure4.1). Furthermore, a tiny portion of each image had been labeled as road, which makes the dataset quite imbalanced. We devised a simple approach to solve this issue. We only sampled batches that have more than 10% of it labeled as road and also use a large crop size to cover a larger area (256 rather than 64 in [7]). We also augment our dataset with random flipping and random 90 degree rotation. For each image in original dataset there will be 8 (4 possible rotation and left-right flipping) images in the augmented dataset. Before feeding the images to the model, channel-wise dataset mean values are subtracted from each sample to normalize the data.

4.4.3 The choice of loss function

As for the loss function we tested for combination of of binary cross entropy, minimum square error and weighted dice coefficient loss which are the common choices for binary classification. Binary cross entropy loss is defined as follows:

$$J_{BCE}(W) = \frac{-1}{N} \sum_{n=1}^{N} [y_n log \hat{y}_n + (1 - y_n) log(1 - \hat{y}_n)] \tag{4.1}$$

where y_n is the target value of sample n and $\hat{y}_n = g(w.x_n)$ and g is the logistic function. In many semantic segmentation dataset, there is one or more classes that are underrepresented which makes the whole dataset imbalanced. In such cases the class of interest occupies only a very small portion of the image. Dice loss is successfully applied to many medical image segmentation problems [43][44]. The dice coefficient loss is computed as follows:

$$J_{Dice}(W) = 1 - \frac{(2 * \sum_{n=1}^{N} \hat{y}_n * y_n) + \lambda}{\sum_{n=1}^{N} y_n + \sum_{n=1}^{N} \hat{y}_n + \lambda} \tag{4.2}$$

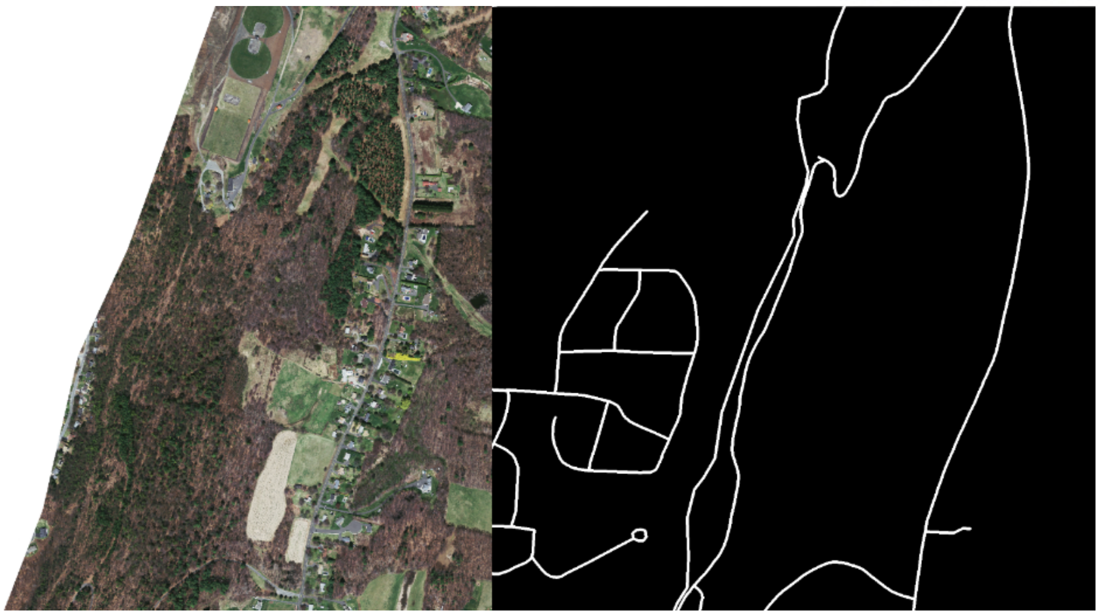
Figure 4.1: An example of road data provided by [7]. In some of the images in the dataset, aerial image is incomplete but the corresponding groundtruth has the complete annotation.

Where λ is small value that will be added to the numerator and denominator. If both of the output and target are empty, it will make the dice to be 1. If either of output or target are empty (e.g. all image pixels are non-road), then higher choice for lambda would make the loss smaller (e.g. close to zero). Also we used minimum square error (MSE) as loss function as well which is computed as below:

$$J_{MSE}(W) = \frac{1}{N}\sum_{n=1}^{N} \|(\hat{y}_n - y_n)\|^2 \tag{4.3}$$

We trained our model with each of the mentioned loss functions and also we experimented with a weighted sum of both dice loss and minimum square loss.

4.4.4 Implementation

We implemented our model using tensorflow [45]. We trained our network using Momentum optimizer [46] algorithms with momentum parameter set to 0.9. As for the learning rate policy we used “poly” learning rate policy, where the initial learning rate is multiplied by $(1 - \frac{currentiteration}{maxiterations})^{power}$ with power set to be 0.9. Since the global structure (like whether the image is taken from urban area or rural area) of aerial image plays an important role for segmentation, we used a large cropsize to train our patch segmentation model. Furthermore, since we are using dilated convolution with large rate, in order to have valid values (e.g. convolution on the feature maps rather on the zero-padded regions), we have to use a large crop size. We took crop size of 256×256 taken randomly from an image. We use extensive studies on the components of the model such as number of min-batches, using shallower network, the choice of loss function, etc. Figure 4.2 gives an overview of our model.

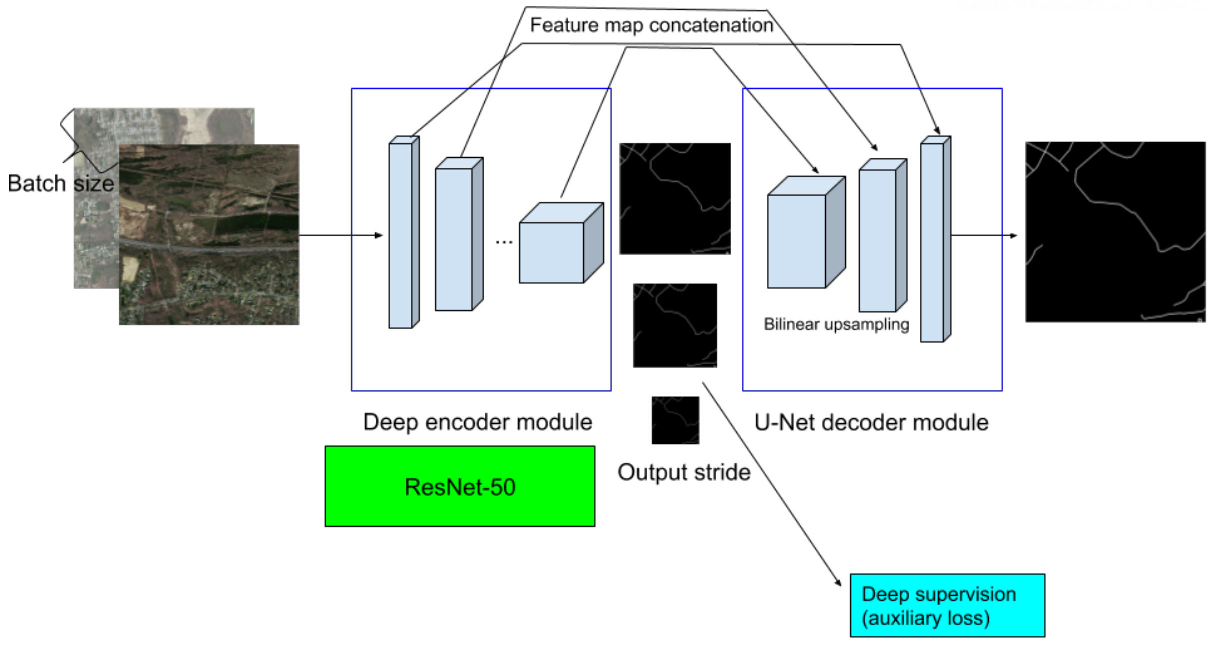

Figure 4.2: Our novel deep fully residual convolutional neural networks (DFRCN).

4.5 Evaluation metrics

4.5.1 Relaxed precision/recall

The common evaluation metric for binary classification methods are precision and recall. It is also known as correctness and completeness in road prediction as well. Precisely, precision is the fraction of correctly predicted road pixels over all predicted road pixels. Recall is the fraction of all the road pixels in the ground truth labeled map that are correctly predicted by the model. Mnih and Hinton [7] introduced relaxed precision and recall scores since correctly labeling all the road pixels were difficult. The relaxed precision is the fraction of number of pixels predicted by the model as road within a proximity of ρ pixels from pixels labeled as road in the groundtruth label map. The relaxed recall is the fraction of number of pixels labeled as road that are in a proximity of ρ pixels from pixels that are predicted by the model as road. In all our experiments that involves computing relaxed precision/recall we chose ρ to be 3 which is consistent choice with previous studies [47] [37].

4.5.2 Mean Intersection over Union (mIoU)

We also report the mean Intersection over Union (IOU) also know as Jaccard Index. It measures the intersection over the union of the labelled segments for each class labels and reports the average over classes. In other words, it is defined as follows: $IOU = \frac{TP}{TP+FP+FN}$ where TP, FP, FN stands for true positive, false positive and false negative respectively. Note that in computing mIOU we didn't use relaxed measurements as proposed by [7].

Chapter V

Experiments

5.1 Introduction

For testing the components of our model deep fully residual convolutional neural networks (DFRCN), we conducted experiments on road extraction from aerial images on Massachusetts road dataset.

5.2 Experimental Results

We experimented with different variations of the components of our model. What follows is the summary of experiments with both quantitative and qualitative result. Some sample images from dataset can be viewed in Figure 5.1

5.2.1 The choice of loss function

We mentioned in chapter 3 that we had three options for training our model: Binary cross entropy (BCE), Dice coefficient loss and Minimum square error (MSE). Our motivation for using dice loss comes from the intrinsic dataset imbalance that is observed in many aerial image segmentation tasks such as road extraction. For example, in road extraction a tiny portion of an aerial image is occupied by road and most of the pixels of the image are labeled as background. This class imbalance usually lead to learning process to stuck at a local minima of the loss function which produce a model that its prediction are biased towards the background class. In such cases, the class of interest (e.g. roads in road extraction task) is often missing (all the pixels are predicted as background) or partially detected. To remedy this problem of class imbalance, some used sample re-weighting in the loss function where foreground regions are given more importance than background ones. However, this is heavily sensitive to the ratio that is being used for re-weighting.

In this work we used dice loss which comes from Dice Similiarity Coefficient (DSC). It measures the amount of agreement between two image regions. It is widely used as a metric to evaluate semantic segmentation models in medical image analysis. It is defined as the below formula:

$$DSC = \frac{2 * |GT \cap PR|}{|GT| + |PR|} \tag{5.1}$$

In above formula $|GT|$ and $|PR|$ is the number of pixels labeled as class of interest in ground truth image and model's prediction respectively. Also $|GT \cap PR|$ is the number of pixels that has the label of class of interest in both ground truth and prediction in the exact same location. Unfortunately, the DSC can not be used as a loss function since it is not differentiable. However, a continuous version of the dice score (known as dice loss, refer to Equation 4.2) have been suggested that allow differentiation

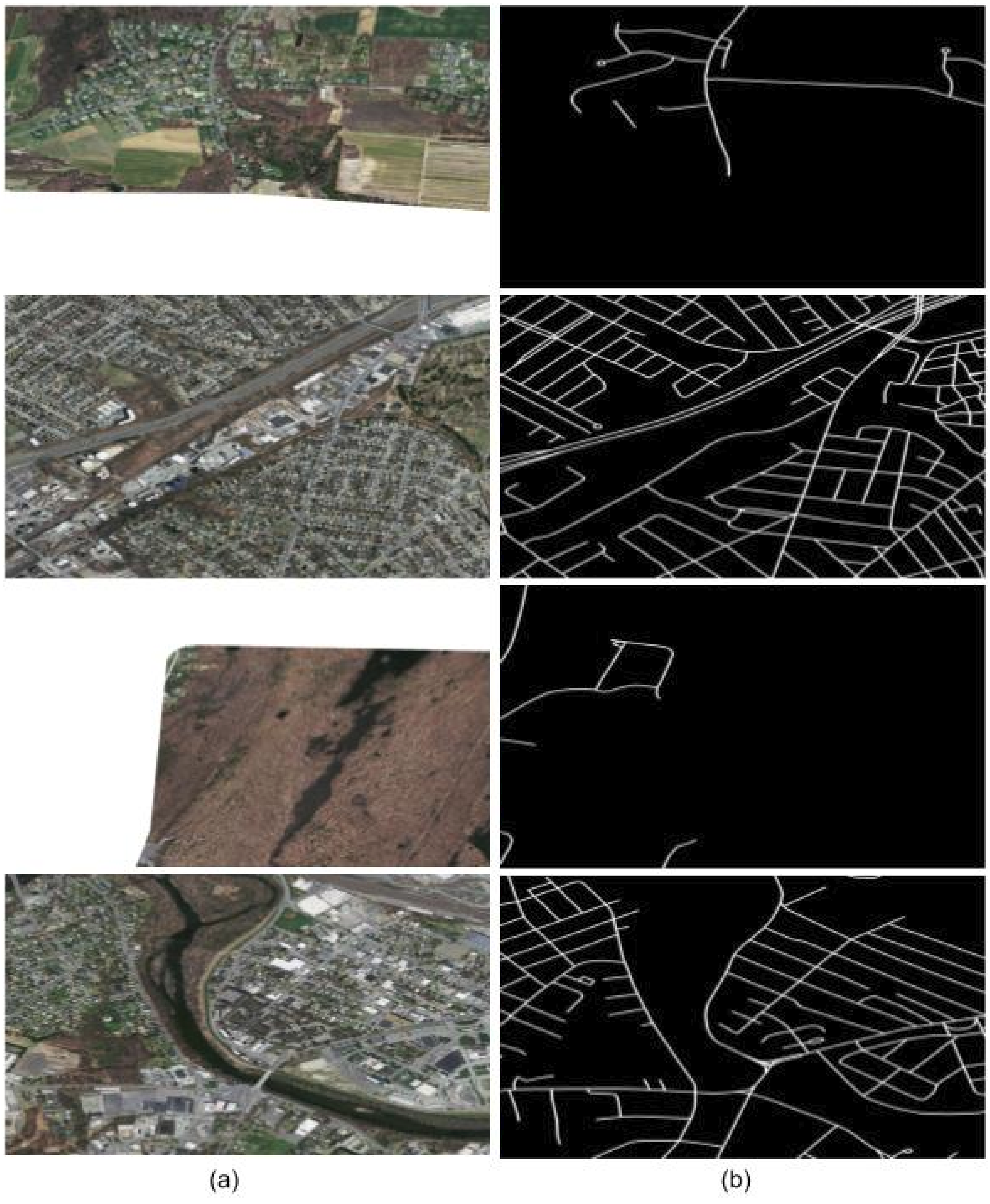

Figure 5.1: Samples from training data of Massachusetts road dataset. (a) Input image; (b) Groundtruth.

and can be used as loss function for updating network parameters in each iteration of backpropagation algorithm.

Nonetheless, there are other loss functions that can be used for dense prediction task. One of such loss functions is mean squared error (MSE) that is commonly used in low-level vision tasks such as super resolution, image denoising, etc. We believed that using dice loss would result in a precise model since it values the intersection of both ground truth and prediction yet using MSE would result in a model with higher recall. As compared to mean squared error, dice loss gives less weight to outliers meaning it values strong local connectivity of pixels. In the end we tested our model with a weighted combination of both MSE and dice loss (refer to Equation 4.2). The final model that is trained with a weighted combination of both loss functions have a better performance than model's that were trained with only individual losses.

We trained the same network with batch size 8 of crop size 256x256 images and initial learning rate 0.001. The training time kept same for all models with different loss functions. We initialized the encoder part of all models from a ResNet-50 model that was pretrained on ImageNet dataset for image classification. For combination of Dice and MSE loss we multiplied Dice loss with $\alpha = 10$ to be in the same order of magnitude with MSE loss. The result of all models are in table 5.1. We noticed that training with BCE converge to all zero output. One possible explanation for this behavior could be the class imbalance and the presence of log function in computing binary cross entropy. As we observed, training with Dice loss would result to a more precise model but with lower recall. However, training with MSE would result with a model with higher recall and lower precision.

	MSE loss	Dice loss	BCE loss	Dice+MSE loss
relaxed precision	0.85	0.89	X	0.88
relaxed recall	0.86	0.76	X	0.83
mean IOU	40.12	41.24	X	42.64

Table 5.1: The effect of loss function choice on the performance of model

5.2.2 Deep supervision

Deep supervision [48] is shown to stabilize the training process of deep networks and also helps to get a better performance. Training deeper network (e.g. more than 20 layers) for dense prediction can be cumbersome task. In order to smooth this process, [48] proposed adding auxiliary loss computation branches to intermediate layers of the networks. Also PSPNet [5] trained their model with deep supervision loss as well. Here we add this auxiliary loss to the last layer of encoder. In order to compute the auxiliary loss, we downsampled the groundtruth label map to the size of encoder output. The network incorporated with deep supervision produces better output. The quantitative results of model with and without deep supervision is in table 5.2. Also to have a visual comparison, we show some of output of the model with and without deep supervision in 5.3.

5.2.3 Batch size

The effect of batch size on the performance of semantic segmentation model is mentioned in the chapter 3. Furthermore, it was already emphasized in [36][6] paper. We were curious whether the number of batches are important to the final performance of the model in aerial image segmentation as well. So we

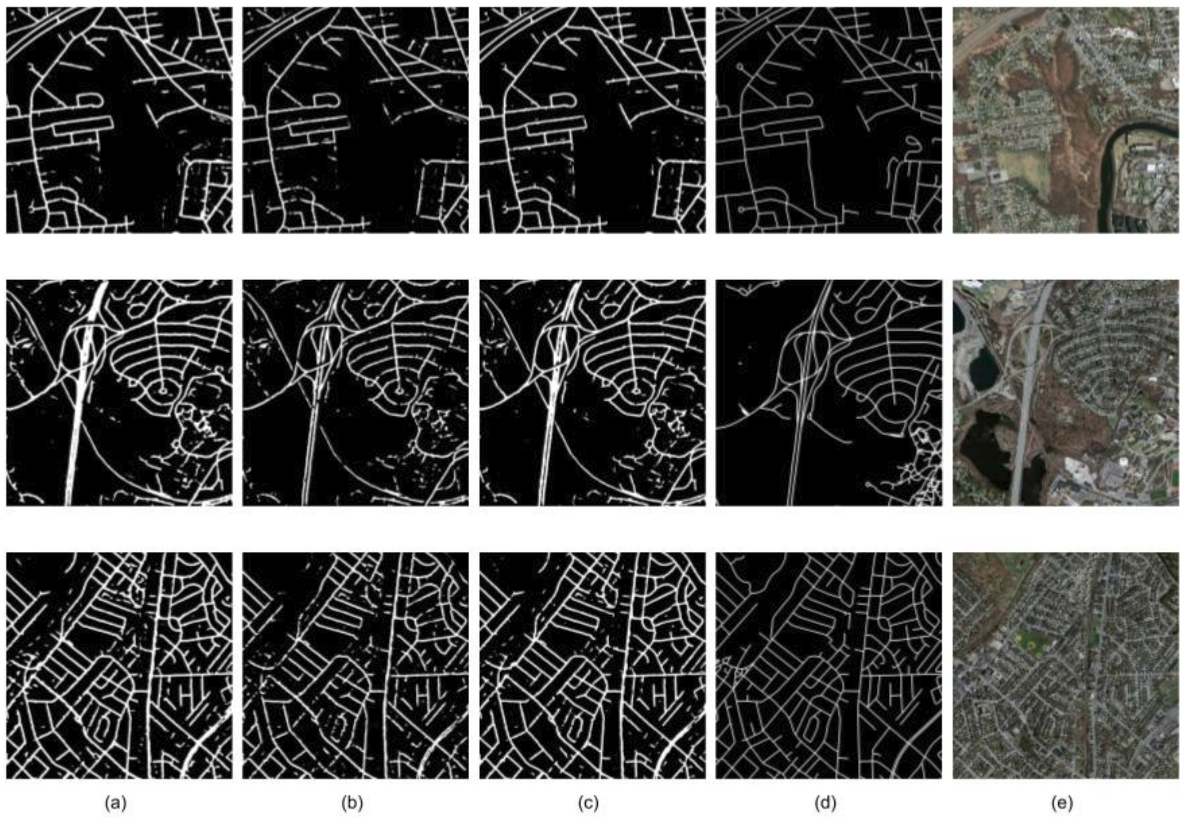

Figure 5.2: Example results on the validation set of Massachusetts road dataset. (a) Using MSE loss; (b) Using dice loss; (c) Using weighted MSE and dice loss; (d) Ground truth; (e) Input image.

	with deep supervision	without deep supervision
relaxed precision	0.85	0.82
relaxed recall	0.86	0.85
mean IOU	40.12	37.67

Table 5.2: The effect of deep supervision (e.g. auxiliary loss) on the performance of model

experimented with ResNet-50 as encoder with batch size 1, 2, 4 and 8 with cropsize 256×256 without using deep supervision. Trained models performance can be seen in table 5.3. We observed that when a model is trained with 1 batch it can have a better performance compare to the model that was trained with 8 batches of images. This may be due to the fact that batch normalization has different behavior during training and inference. Furthermore it was shown that model's final performance is highly sensitive to the computed batch normalization running statistics (mean and variance). Unfortunately we couldn't train a model with large batch size (bigger than 16) because of GPU memory limit. We also would like to mention other normalization techniques that are less sensitive to the batch size like weight normalization [49].

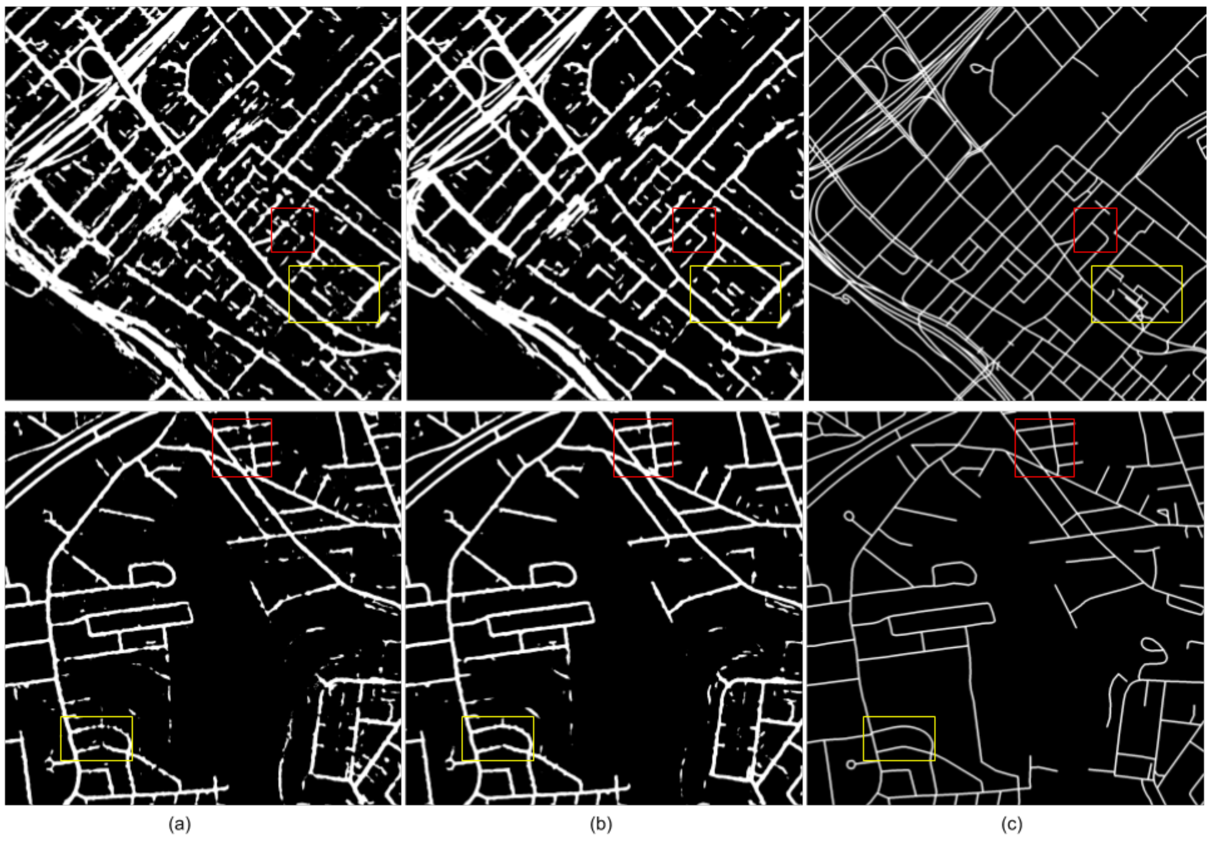

Figure 5.3: Example results on the validation set of Massachusetts road dataset. (a) Without deep supervision; (b) With deep supervision; (c) Ground truth.

	Batch size 1	Batch size 2	Batch size 4	Batch size 8
relaxed precision	0.91	0.84	0.83	0.82
relaxed recall	0.73	0.86	0.85	0.85
mean IOU	38.87	39.56	38.89	37.67

Table 5.3: The effect of batch size on the performance of model

5.2.4 Decoder feature map upsampling

In this section we will discuss the role of decoder in road extraction task. Unlike RefineNet[4] and U-Net[2], PSPNet[5] did not rely on a multi-layer decoder to upsample and transform the encoded feature map to the final prediction output. Furthermore, the first version of DeepLab-v3 [6] also did not employ multi-layer decoder. However, in their next work DeepLab-v3+ [50], they used a simple decoder mainly based on concatenating early layers with higher layers and upsampling them with bilinear interpolation of DeepLab decoder. In this section, we wanted to know the difference (in terms of performance) between a model with and without a decoder module. We would like to mention that there are many ways to devise a novel decoder. However, it wasn't our focus. Our main focus in this part is whether a simple decoder can improve the performance or not. For this matter, we devised our decoder based on U-NET (one of the early simple ways to upsample feature maps in deep convolutional neural networks).

In the controlled study of our decoder module, we further investigated the effect of feature map upsampling choices. We fixed the encoder part (ResNet-50) and the skip-connection links from the mid layers of the encoder. All the training parameters were kept same except the upsampling choices.

In our encoder-decoder model we have five upsampling layers since there are five down-sampling layers in our deep encoder. We have four skip-connection links in the form of concatenating the feature maps from the encoder module with the upsampled feature maps of our decoder (refer to Figure 5.4).

Here we studied the effect of using transposed convolution or a bilinear up-convolution as our upsampling layer (blue blocks in Figure 5.4). In an experiment we used five bilinear interpolation for our upsampling layers. Here we should note that unlike transposed convolution, bilinear interpolation does not have any learnable parameters that can be optimized by the network. In another experiment, we used 5 transposed convolution with kernel size 4x4 and stride 2 as an upsampling layer. To make a fair comparison, we made an experiment and utilized an additional convolution layer after each bilinear interpolation layer to have a model with the exact same number of parameters as the model with transposed convolution. We call this layer bilinear up-convolution. The performance of each model is in Table 5.4. We observed that using transposed convolution instead of a simple bilinear interpolation would increase the performance about 1.65%. This is somewhat expected since transposed convolution would benefit from additional tunable parameters for upsampling the feature maps. We also observed that a model with bilinear interpolation followed by a convolution layer that has equal parameters with the model that utilizes transposed convolution has 1.3% better performance in terms of mean intersection over union. The better performance would be the result of alleviating the checkerboard artifact that is produced as a side-effect of using transposed convolution.

One way to look at transposed convolution is a convolution that is applied on the input feature maps that are interleaved with zero between each element (Figure 5.5). This type of convolution would result in some area of input feature map to have uneven overlap. The uneven overlap on a 2D image would result in artifacts that look like checkerboard with some areas have varying magnitude than the others. The checkerboard artifact can affect the final performance of model for road extraction task (Figure 5.6).

	Bilinear upsampling	Transposed convolution	Bilinear up-convolution
relaxed precision	0.73	0.74	0.77
relaxed recall	0.89	0.87	0.89
mean IOU	32.65	34.30	35.60

Table 5.4: Decoder upsampling choice effect on the final performance of the road extraction model

5.2.5 Output stride

Using dilated convolution enable us to have control over the ratio of encoder output to it’s input which is known as output stride. The other benefit of dilated convolution is we can use the weight that are trained

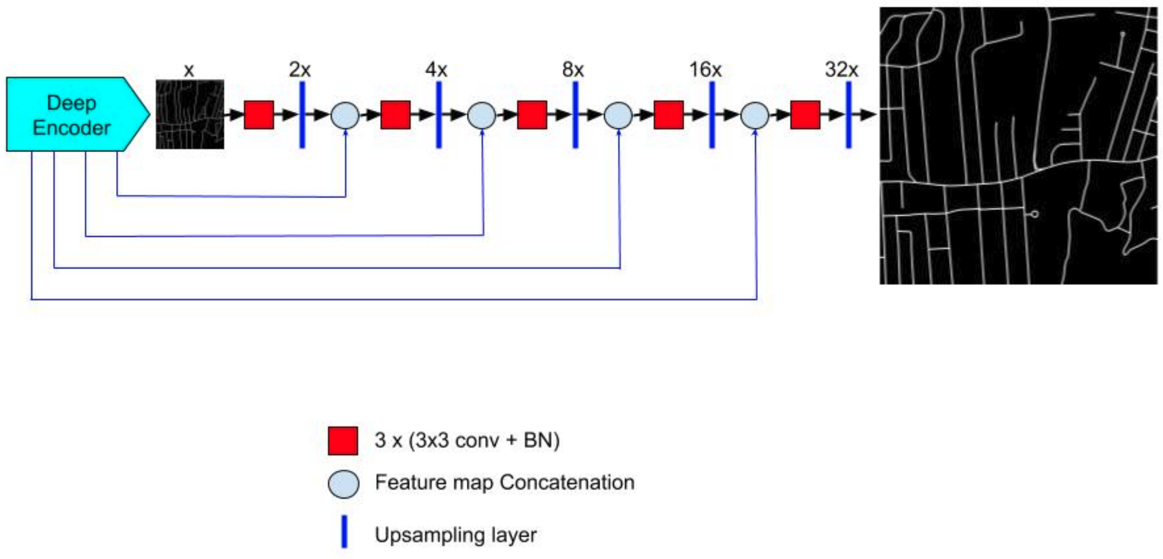

Figure 5.4: The schematic of our decoder module.

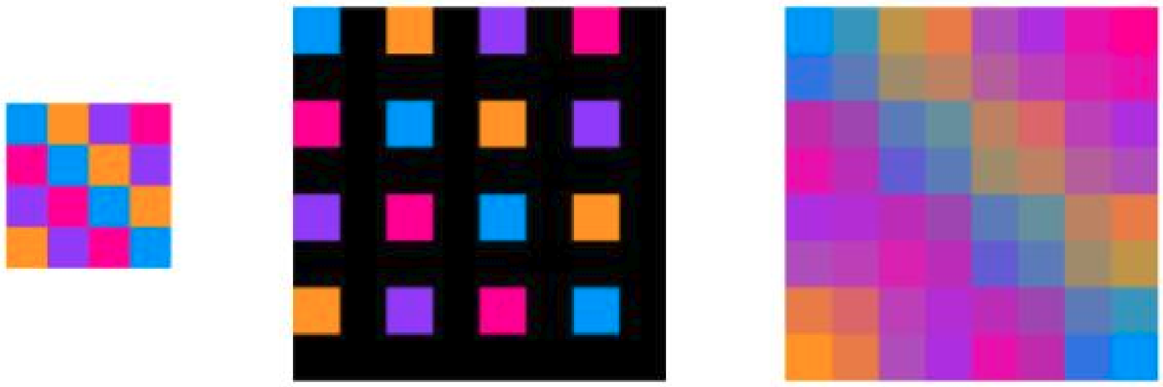

Figure 5.5: The difference of transposed convolution and bilinear interpolation. Left: Input feature map. Middle: Input to transposed convolution. Right: Bilinear upsampled input feature map.

with vanilla convolution. This made dilated convolution an important tool for dense prediction tasks. Specially, when for some of these tasks like aerial image segmentation a large (relative to ImageNet) dataset is not available and therefore, a pretraining from an image classification model would heavily ease the training process.

In this section we investigate the role of output stride in our model's performance. We experimented with various output stride of ResNet-50 model (e.g. 32, 16, 8, 4). Note that using output stride 32 means we are using dilated convolution with rate 1 which acts like a vanilla convolution. As a result of changing the output stride, we were bound to make some minor modifications to the decoder module as well. These modifications include manipulating the bilinear upsampling ratio after the first convolution layer in decoder. The quantitative results are in table 5.5 and to figure 5.7 depict some of the results.

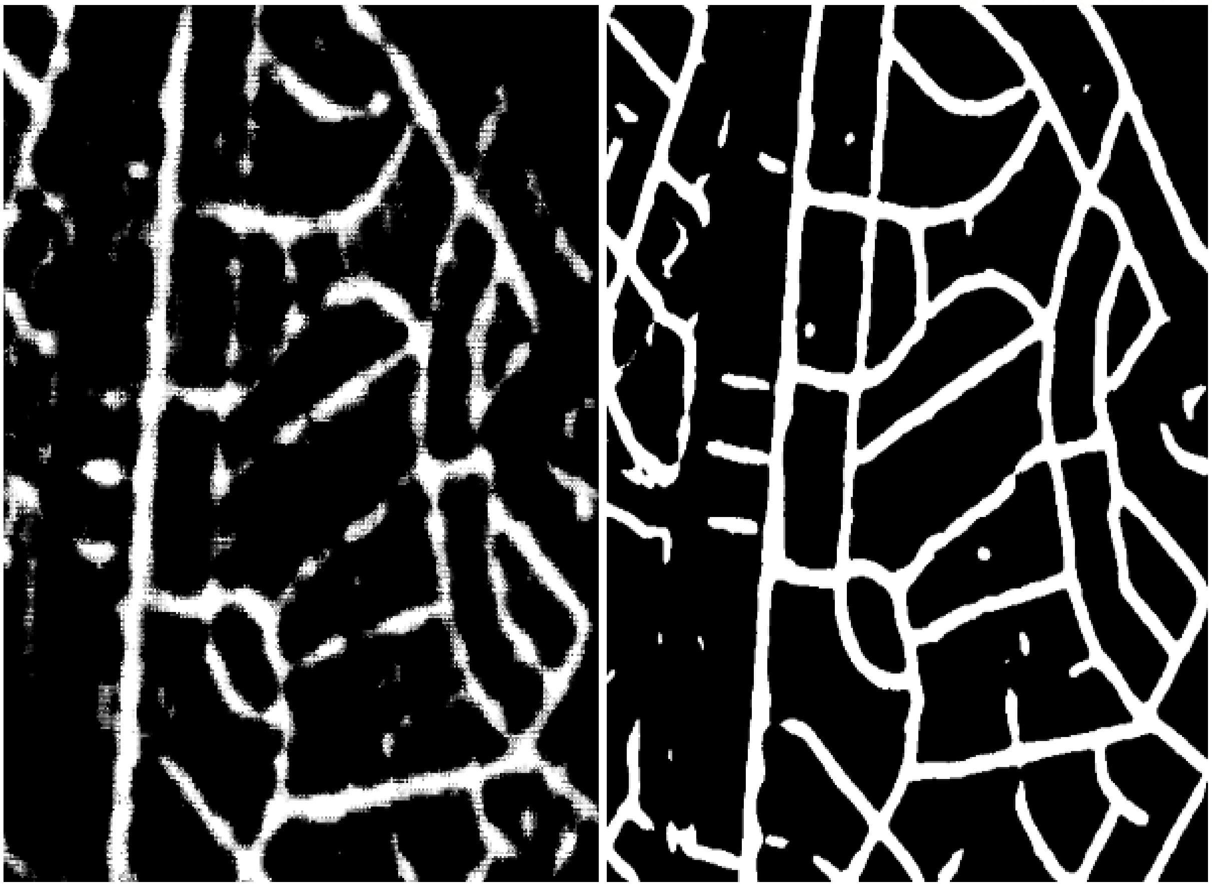

Figure 5.6: Checkerboard artifacts in the prediction of a decoder with transposed convolution. Left: The result of a model with a decoder that has transposed convolution as an upsampling layer, Right: The result of a model with Bilinear up-convolution (Bilinear interpolation with a convolution

	Output stride 4	Output stride 8	Output stride 16	Output stride 32
relaxed precision	0.85	0.82	0.81	0.79
relaxed recall	0.86	0.84	0.82	0.82
mean IOU	40.12	36.20	34.72	34.43

Table 5.5: The effect of output stride on the performance of model

5.2.6 Test time augmentation

In this section, we are testing the performance of our simple yet effective test time augmentation. As a common practice for many of the state-of-the-art models like [5][6], which use left right flipping during test time for increasing the performance we make use of D_4 dihedral group 4 for test time augmentation [51]. Precisely, we rotate each image with three 90 degree rotations(refer to figure for 5.8). We do the inverse rotation on the output of the model and average all to make the final prediction. Here we report the result of model with and without test time augmentation. Refer to table 5.6.

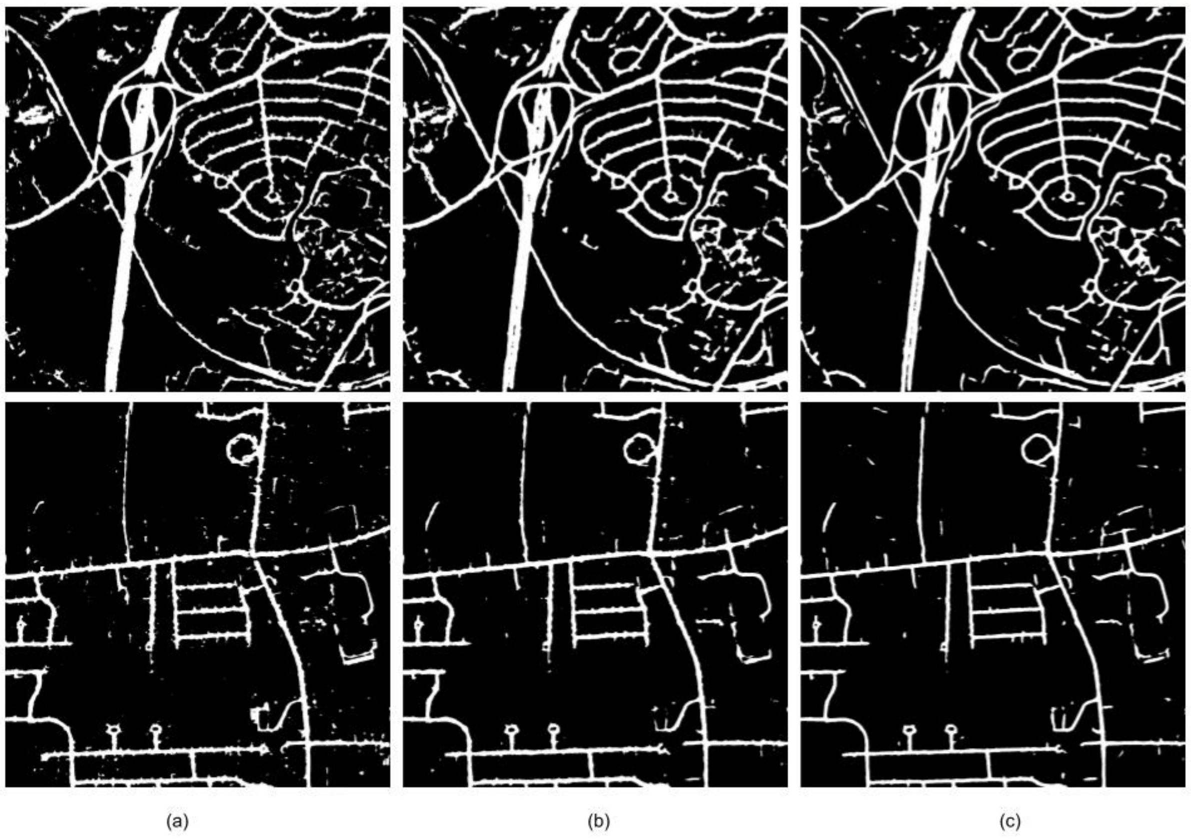

Figure 5.7: Example results on the validation set of Massachusetts road dataset. (a) Output stride 32; (b) Output stride 8; (c) Output stride 4.

	with test time augmentation	without test time augmentation
relaxed precision	0.88	0.86
relaxed recall	0.83	0.84
mean IOU	42.64	41.35

Table 5.6: The effect of test time augmentation on the performance of model

5.2.7 Comparison with other models

The common metric that is used for reporting each model's prediction accuracy for road extraction task is called break-even point [7][37]. Break-even point is the point that precision and recall have the closest value to each other. A higher break-even point means a better performance in both precision and recall. Table 5.7 is for comparison of our proposed model in terms of break-even point for road extraction task on Massachusetts roads test dataset against other deep learning based models.

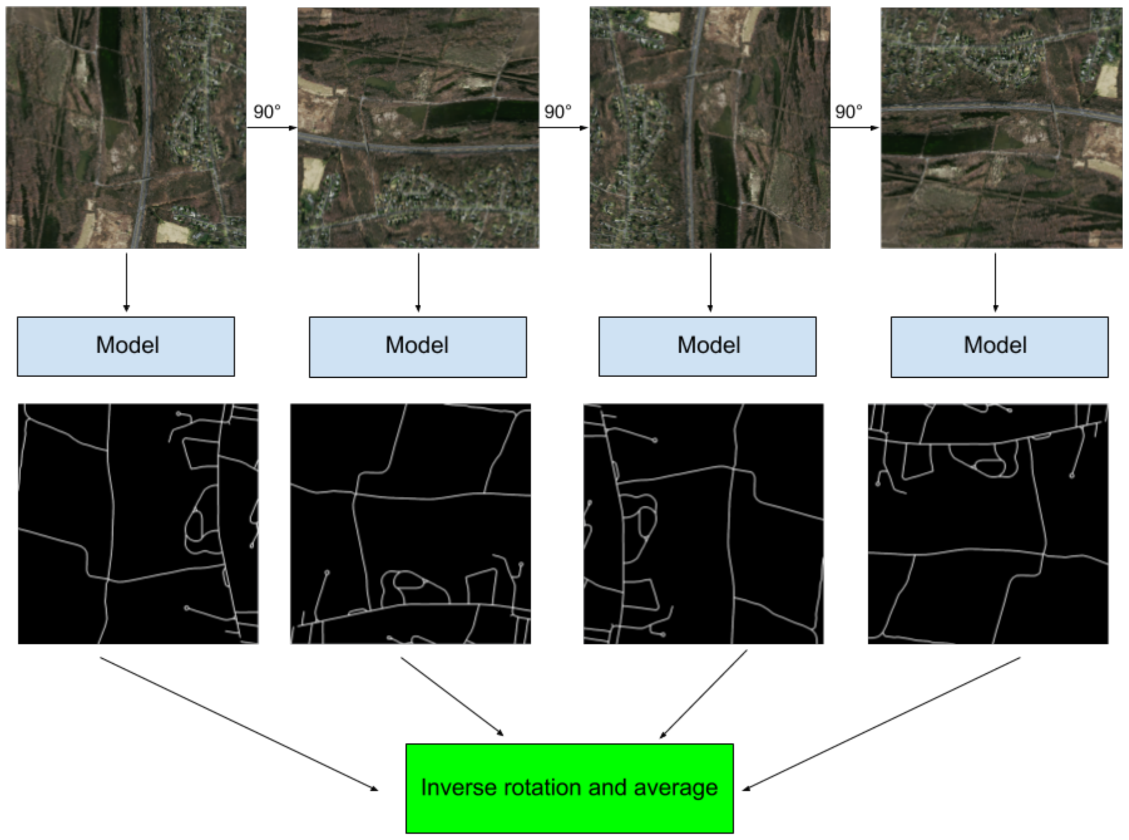

Figure 5.8: Test time augmentation for final prediction.

Model	Break-even point
Mnih-CNN [7]	0.8873
Mnih-CNN+CRF	0.8904
Saito-CNN [37]	0.9047
Ours (DFRCN)	**0.9132**

Table 5.7: Comparison of the road extraction models in terms of break-even point.

Chapter VI

Conclusion

Semantic image segmentation is a sub-field of computer vision that aims to partition each image into predefined semantically meaningful parts. In recent years, approaches to solve this problem were mainly based on deep learning algorithm. To be precise, they were all a variant of Fully Convolutional Network (FCN). In this thesis, we tried to implement one of the famous recently proposed Pyramid Scene Parsing Network (PSPNet) since there were no available source code to replicate the training process. Along the implementation, we found some important details regarding the performance such as training batch normalization layer parameters.

Furthermore, we devised our own Deep Fully Residual Convolutional Neural Network (DFRCN) for semantic image segmentation. In our experiments, we tested our model on road extraction task which is challenging problem in aerial image segmentation. We made a through analysis on the components of our model and the contribution of each component to the final performance. In the end, we would like to mention that our model can be used for any semantic segmentation task with minor modification because we devised it in such a way that can used as an off-the-shelf framework.

Acknowledgements

I would like to express my gratitude for:

- My parents, Nasrin and Kazem, for the support throughout my life;
- My supervisor, Jaesik Choi, for the valuable advice and support towards my graduate studies;
- My siblings; Tata, Hosein and Ehsan, for believing in me and the choices I made;
- My lab-mates from Statistical Artificial Intelligence Lab, for the helpful comments and discussions;
- All the people I have met in UNIST and all around the country during my wonderful stay in Korea.

References

[1] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[3] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1520–1528.

[4] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2881–2890.

[6] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[7] V. Mnih, "Machine learning for aerial image labeling," Ph.D. dissertation, University of Toronto, 2013.

[8] K. Kuan, M. Ravaut, G. Manek, H. Chen, J. Lin, B. Nazir, C. Chen, T. C. Howe, Z. Zeng, and V. Chandrasekhar, "Deep learning for lung cancer detection: Tackling the kaggle data science bowl 2017 challenge," *arXiv preprint arXiv:1705.09435*, 2017.

[9] M. Rajab, M. Woolfson, and S. Morgan, "Application of region-based segmentation and neural network edge detection to skin lesions," *Computerized Medical Imaging and Graphics*, vol. 28, no. 1-2, pp. 61–68, 2004.

[10] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, "Network dissection: Quantifying interpretability of deep visual representations," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 3319–3327.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[12] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[13] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[14] D. P. Casasent, J. S. Smokelin, and A. Ye, “Wavelet and gabor transforms for detection,” *Optical Engineering*, vol. 31, no. 9, pp. 1893–1899, 1992.

[15] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[16] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Region-based convolutional networks for accurate object detection and segmentation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 142–158, 2016.

[18] K. E. Van de Sande, J. R. Uijlings, T. Gevers, and A. W. Smeulders, “Segmentation as selective search for object recognition,” in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1879–1886.

[19] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[20] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, “Deconvolutional networks,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2528–2535.

[21] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.

[22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, “Imagenet large scale visual recognition challenge. arxiv preprint arxiv: 14090575,” 2014.

[23] H. Caesar, J. Uijlings, and V. Ferrari, “Coco-stuff: Thing and stuff classes in context,” *arXiv preprint arXiv:1612.03716*, 2016.

[24] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, “Real-time human pose recognition in parts from single depth images,” in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. Ieee, 2011, pp. 1297–1304.

[25] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, “Deep neural networks segment neuronal membranes in electron microscopy images,” in *Advances in neural information processing systems*, 2012, pp. 2843–2851.

[26] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs (2016)," *arXiv preprint arXiv:1606.00915*, 2016.

[27] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *european conference on computer vision*. Springer, 2014, pp. 346–361.

[29] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[30] J. Fu, J. Liu, Y. Wang, and H. Lu, "Stacked deconvolutional network for semantic segmentation," *arXiv preprint arXiv:1708.04943*, 2017.

[31] P. Luo, G. Wang, L. Lin, and X. Wang, "Deep dual learning for semantic image segmentation," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, 2017, pp. 2737–2745. [Online]. Available: https://doi.org/10.1109/ICCV.2017.296

[32] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[33] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," *arXiv preprint arXiv:1404.5997*, 2014.

[34] Z. Wu, C. Shen, and A. v. d. Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *arXiv preprint arXiv:1611.10080*, 2016.

[35] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[36] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun, "Megdet: A large mini-batch object detector," *arXiv preprint arXiv:1711.07240*, 2017.

[37] S. Saito, T. Yamashita, and Y. Aoki, "Multiple object extraction from aerial imagery with convolutional neural networks," *Electronic Imaging*, vol. 2016, no. 10, pp. 1–9, 2016.

[38] Z. Zhang, Q. Liu, and Y. Wang, "Road extraction by deep residual u-net," *IEEE Geoscience and Remote Sensing Letters*, 2018.

[39] B. Liu, H. Wu, Y. Wang, and W. Liu, "Main road extraction from zy-3 grayscale imagery based on directional mathematical morphology and vgi prior knowledge in urban areas," *PloS one*, vol. 10, no. 9, p. e0138071, 2015.

[40] G. Cheng, Y. Wang, S. Xu, H. Wang, S. Xiang, and C. Pan, "Automatic road detection and centerline extraction via cascaded end-to-end convolutional neural network," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 6, pp. 3322–3337, 2017.

[41] G. Cheng, F. Zhu, S. Xiang, and C. Pan, "Road centerline extraction via semisupervised segmentation and multidirection nonmaximum suppression," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 4, pp. 545–549, 2016.

[42] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, vol. 1, no. 10, p. e3, 2016.

[43] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE, 2016, pp. 565–571.

[44] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal, "The importance of skip connections in biomedical image segmentation," in *Deep Learning and Data Labeling for Medical Applications*. Springer, 2016, pp. 179–187.

[45] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[46] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, 2013, pp. 1139–1147.

[47] V. Mnih and G. E. Hinton, "Learning to detect roads in high-resolution aerial images," in *European Conference on Computer Vision*. Springer, 2010, pp. 210–223.

[48] L. Wang, C.-Y. Lee, Z. Tu, and S. Lazebnik, "Training deeper convolutional networks with deep supervision," *arXiv preprint arXiv:1505.02496*, 2015.

[49] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 901–909. [Online]. Available: http://papers.nips.cc/paper/6114-weight-normalization-a-simple-reparameterization-to-accelerate-training-of-deep-neural-networks.pdf

[50] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *CoRR*, vol. abs/1802.02611, 2018. [Online]. Available: http://arxiv.org/abs/1802.02611

[51] V. Iglovikov and A. Shvets, "Ternausnet: U-net with VGG11 encoder pre-trained on imagenet for image segmentation," *CoRR*, vol. abs/1801.05746, 2018. [Online]. Available: http://arxiv.org/abs/1801.05746