REVISE U-NET FOR CAR IMAGE MASKING

A Thesis presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

YINI DUANMU

Dr. Advisor YI SHANG, Thesis Supervisor

DEC 2018

The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled:

REVISE U-NET FOR CAR IMAGE MASKING

presented by Yini Duanmu,

a candidate for the degree of Master of Science and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Yi Shang

Dr. Yunxin Zhao

Dr. Jianlin Cheng

ACKNOWLEDGMENTS

I would like to express my special appreciation and thanks to my advisor Professor Yi Shang, for not only providing me opportunity to embark on this project, but also his constant guidance and encouragement. I would also like to thank my committee members, Professor Jianlin Cheng and Professor Yunxin Zhao for their help in my studies and serving as my commitee members.

To my labmates and friends who shared their support either morally, financially and physically, which made my life in Mizzou more interesting and enjoyable, thank you.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

ABSTRACT

We proposed a practical deep fully convolutional architecture consisted of an encoder and a decoder for semantic pixel-wise segmentation. This architecture is designed based on the U-Net [1]. The architecture of the encoder network is the same as the 10 layers from VGG-16 [2] using pre-trained weights. The decoder up-samples and generates a pixel-wise classification for the input image. Our improvement was primarily motivated by the prior knowledge of our task. The images of our task all share the same high resolution and the same size, the objects are always at the center of the images.

Our architecture is motivated by the problems of segment large object in the high resolution image using state-of-the-art encoder-decoder network. These networks may generate segmentation predictions without very high accuracy boundaries or with holes inside of the car or missing part of the car due to the unclear boundaries in the input image. Our architecture is rectified these problems by: First, it use the pooling indices from the encoder to up-sample the feature map in the corresponding decoder, which gives the decoder a non-linearity up-sampling, and lead to higher accurate boundaries. Second, the coordinate maps are used to solve the coordinate transform problem. The correlation between the content and their coordinate can be learned. Third, the dilated convolution is used to enlarge the receptive field of the network. Features with more global information can be learned. Experimental results are shown to validate the advantages of the proposed methods compared with the original methods.

Chapter 1

Introduction

In the last few years, deep neural network become very popular in fields including computer vision, natural language processing, recommend system, audio recognition, speech recognition, etc. In particular, deep convolutional neural networks have seen huge success in many visual recognition tasks. In the field of computer vision, the typical use of deep convolutional neural networks is on the classification tasks, where we classified an image to a single class label. However in real life, a single class label to a image is not always enough. For example, the online stores often need product pictures without the background, which is a heavy task to remove the background from the millions of pictures. This task requires a semantic pixel-wise labeling, i.e., each pixel in the image should be classified.

Some of recent approaches used fully convolutional neural networks have tried to predict pixel-wise labeling. These architectures often contains with a encoder and a corresponding decoder. Although the results are pretty good, but still have some problem.

First, the encoders of these approaches often adopt the fully convolutional parts of architectures designed for category prediction tasks. The max pooling operation which often used in the encoder to subsample the feature maps leads to the lost

of spatial information. Our method, inspired by the SegNet [3], recode the pooling indices to keep these spatial information that reduce the pressure of the decoder and get more accurate boundary localization.

Second, the correlation between the content and their location is not taken into account in these approaches. Because the translation invariant of convolution gives the fully convolutional architecture the same property of translation invariant. This means the prior probabilities of the objects at different locations are the same. However, in our tasks, the objects are always at the center of the image, i.e., we need to give different priors to different parts of the image. We proposed a method to encode the coordinate maps into the feature map which allows the models to learn the correlations between the features and the location.

Finally, the receptive field of these architectures are too small to high resolution images. Due to the limitation of the number of parameters, these architectures cant afford too much layers of convolution, which gives these architectures the limitation of the reception field. We proposed to use dilated convolution instead of typical convolution to enlarge the reception field.

Using these methods we train our models and make comparison between them.

Chapter 2

Related work

In the past years before the development of the Deep Convolutional Neural Networks, the semantic segmentation systems often relied on hand-crafted features classifying pixels in the input images with some flat classifiers, such as Boosting [4], [5], Random Forest [6], [7], Support Vector Machine(SVM). There are approaches using paths as the input of these flat classifiers to make category prediction either for the center pixel of each patch [6], [7], [4], [5], or for all the pixels in each patch [8]. They will get a noisy prediction results of each pixels in the input image, then a pair-wise or higher order CRF are used to smooth the results and improve the accuracy. However the expression of these hand-crafted features are still limited.

Because the huge success of the application of the Deep Convolutional Neural Networks in the classification tasks, the richness of the features extracted from the deep convolutional neural networks are explored. These features are used to apply in structured prediction problems such as segmentation. Some of these approaches make a bottom-up segmentation which first segment the input image into regions, and then find all the regions from the same object, and then classify these proposals using the deep convolutional neural networks [9], [10]. These proposals varies a lot, there are bounding box proposals [11], masked regions [12] and superpixels [13]. The results

can get sharp boundaries due to the good bottom-up segmentation, but the results are still not good enough because errors like background wrongly segmented together with the object may occur. Besides, these segmentation parts are time consuming and tricky.

Multi-scale techeniques are also used, some of these approaches using multiple resolutions for a single image as the inputs of the deep convolutional neural networks. These results then combined to get the final prediction using a segmentation tree [14]. Others [15], [16] using features extract from different layers of a convolutional neural network for a single image to make the prediction for pixels. The common idea of these approaches is a higher accuracy and better boundaries can got because of richer features which combine the local information and global information together. However, because of the huge number of the parameters of these architectures, multi-stage training are need, which make the system complicate to design and a lot of hyper-parameters to adjust.

The latest approaches explored end-to-end training and directly provided the pixel-wise labelling. Architectures in works [17], [3], [1] are all consist of a encoder and a decoder. The encoders and decoders are all fully convolutional neural networks makes the whole architecture a fully convolutional neural networks. The encoder of [3] adopt the architecture of VGG16 [2] and use the weights pre-trained in the ImageNet and remove the fully connected layers. The decoder of [1] upsample the feature maps and combine them with the corresponding features maps from the encoders to make the localization. [3] use a non-linearity upsampling method, which record the location of the maximum value when making the max-pooling in encoder, and using these location to upsample the corresponding feature maps in the decoder. The fewer layers in the [1] may works in some simple tasks but often not enough for some more complicated cases. [3] focus on the improvement of the inference speed, but there are still something can be done to improve the performance. These approaches dont

take the size of receptive fields into consideration and most of these systems are spatial translation invariant, i.e., the correlation between the spatial localization and the objectness is ignored. To overcome these problems, a new architecture is designed.

Our work is based on the architectures of [3], [1], which are fully convolutional encoder-decoder networks. Our work adopted same encoder structure as [3], which is the convolutional part of VGG-16. The pretrained weights from ImageNet are also used. The pooling indices idea from [3] and the concatenate feature maps from encoder and corresponding decoder [1] are combined in our approach. These methods allow us to process a non-linearity upsampling which avoid the lost of spatial information and gain richer feature maps at the same time. The coordinate transform problem is solved by the CoordConv [18], and they use this idea to solve problems like classification, object detection or generative modeling. The CoordConv was used in our work to learn the correlation between the spatial and features. Inspired by[kaggle], dilated convolution (Atrous convolution) solving the limited receptive field is also used in our work.

Chapter 3

Proposed Methods

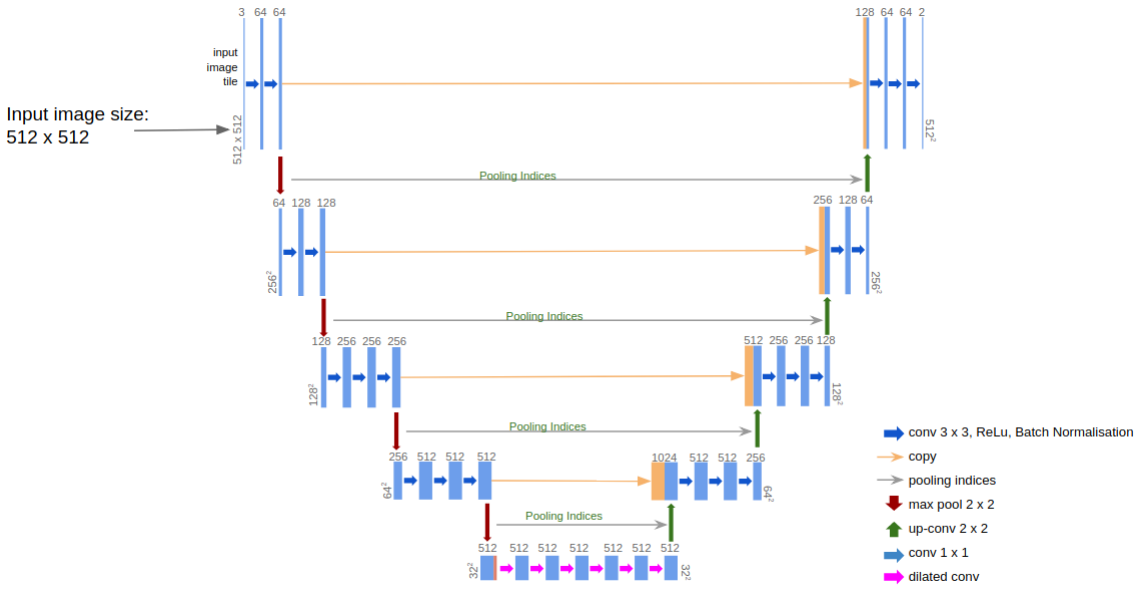

Figure 3.1: The proposed final architecture of this work.

Our network is a fully convolutional neural network, consist of an encoder and a corresponding decoder. This architecture is illustrated in Figure 3.1, which is based on the U-Net model. Because the U-Net model is not good enough for our task, some modifications are used to improve the U-Net model. We used the first 10 layers of VGG-16 [2] which is designed for classification task as our encoder. To have a better

initialization, we adopted the pre-trained weights in the large dataset ImageNet to initialize our network. Each concolution in the network for both encoder and decoder is typically followed by a element-wise activate function ReLu (Rectified Linear Unit). Then the feature maps are batch normalized and down-sampled by a 2 by 2 max-pooling. There are 3 principal methods proposed to improve the base architecture to get this final architecture in Figure 3.1.

First, While doing the max-pooling, we record the location of the maximum value (pooling indices) where the value is taken to down-sample the feature maps. These pooling indices are then used to up-sampling feature maps for the decoder. These up-sampled feature maps concatenated with the feature maps from the corresponding encoder served as the input for each decoder. Second, Coordinate maps are encoded into the middle of the architecture to solve the coordinate transform problem. Third, the dilated 3 by 3 convolution are used instead of typical 3 by 3 convolution between the encoder and the decoder to enlarge the receptive field of the network. Finally, a pixel-wise convolution layer served as a classifier is used to generate the pixel-wise segmentation. The result of this architecture shows a big improvement.

3.1 Encoder Architecture

Our architecture is inspired by the U-Net [1] and SegNet [3], we analyzed their encoders. The encoder of U-Net consist of 8 convolutional layers, each followed by a ReLu while the encoder of SegNet consist of 13 convolutional layers, each followed by a ReLu and Batch Normalization. When the feature maps are convolved in U-Net with no padding, which means after each convolution, the size of the feature maps changed. While SegNet use padding in convolution and the size of the feature maps will always remain the same. Because the U-Net is deal with the biomedical images, these kind of images have no direction, that is the images are always make sense then

they are rotated or flipped randomly. Since the size of the input image and the output segmentation result are different, the authors padded the input image by mirroring to get the full size of result. However, this mirroring padding doesn't make sense for our task, the car image in our dataset can be flipped bottom-up. Of course, we can use zero-padding instead, but the performance is bad, we will discuss this problem later.

The U-Net is used for biomedical tasks, in which the structure of the object(different kind of cells) in biomedical image are relatively simple unlike general object. So less parameters are needed in this kind of tasks, and the pre-trained weights is useless, since these weights are often trained in ImageNet, the general object dataset. A deeper architecture is needed for our task. The architecture of SegNet is a perfect candidate for our task. The first 10 convolution layers from SegNet are used in our work, and the weights are initialized by the ImageNet.

3.2 Up-sampling Method

The up-sampling methods in the U-Net and SegNet are different. U-Net use transposed convolution to up-sample the feature map in each blocks of the decoder. Transposed convolution often used to up-sample features, is an opposite operation to convolution. Figure 3.2 provides an example for transposed convolution. The first grid is called the input feature map, the middle grid is the kernel and the last one is the output feature map. As the example shows, a 2×2 kernel applied to a 3×3 input using 2×2 strides. Each cell in the input feature map multiplied by the kernel generates the cells with the same color as the input cell in the output feature map.

The spatial information lost due to the max-pooling will never recover using this up-sampling method. However, inspired by the SegNet, we solved this problem by recoding the pooling indices during the max-pool operation in the encoder. Then a

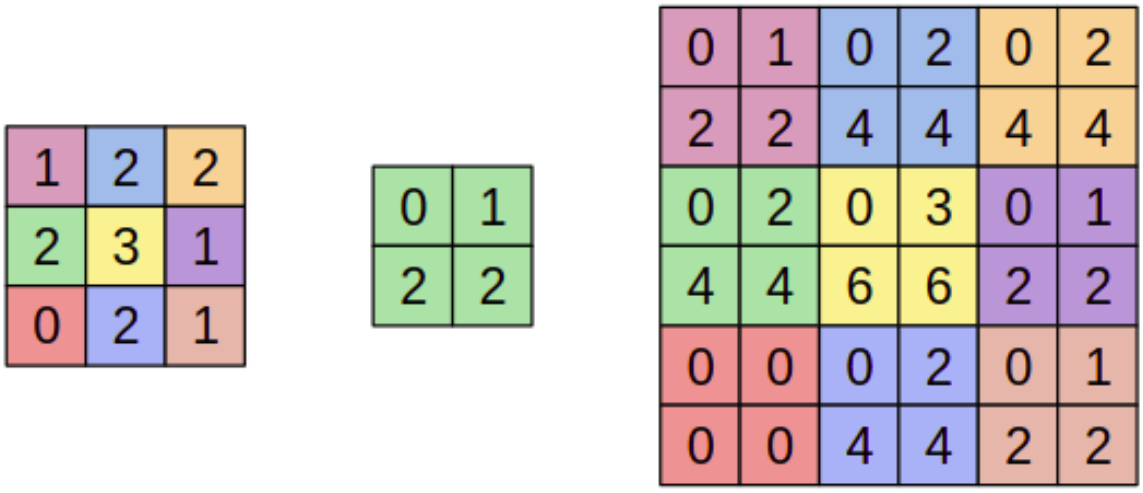

Figure 3.2: Up-sampling using transposed convolution with $kernelsize = 2 \times 2$, $strides = 2 \times 2$. The first element 1 in the left side figure is up-sampled to the up-most and left-most2×2 grid in the right-side figure using the kernel in the middle of the figure.

non-linearity up-sampling is applied using these pooling indices. Figure 3.3 shows an example. The pooling indices are recorded when max-pooling are occured at the red dashed circle. As the figure shows, 2 x 2 max-pooling will lead to 4 possible maximum locations, thus a 2 bit number can be used to save each index. At the corresponding up-sampling at the green dashed circle, the recorded pooling indices are used to map the low resolution feature map into a high resolution feature map. The left-bottom part shows the max-pooling operation and the right-bottom part shows the up-sampling operation. As the result goes, we will get a sparse large feature map.

Compared to the transposed convolution, the one using the pooling indices is definitely avoid of the lost of spatial information. Because the features after the max-pooling are made up from the features at the maximum locations(pooling indices) before the max-pooling. When at the corresponding up-sampling, it is make sense to recover the features to where it comes from(using pooling indices). This accurate up-sampling method will lead to a more accurate localization of the segmentation.

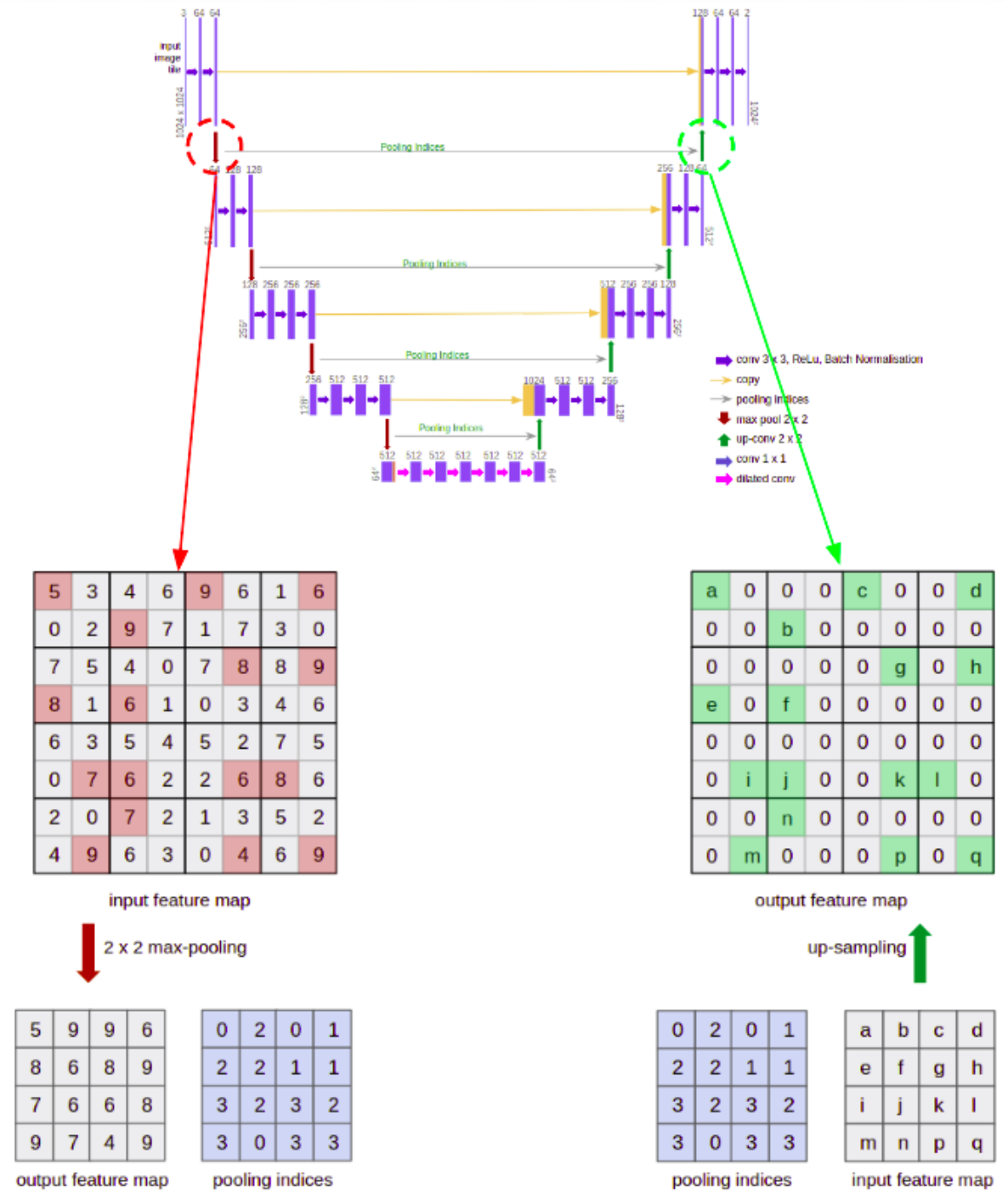

Figure 3.3: This figure shows an example of down-sampling and corresponding up-sampling. The figures in the bottom half are the details of the model in the upper half of the picture. The left side part of the figure is an example of the down-sampling which is max-pooling operation indicated by the dash red circular. The right side part of the figure is the corresponding up-sampling of the indicated dash green circular using the pooling indices saved at the left figures.

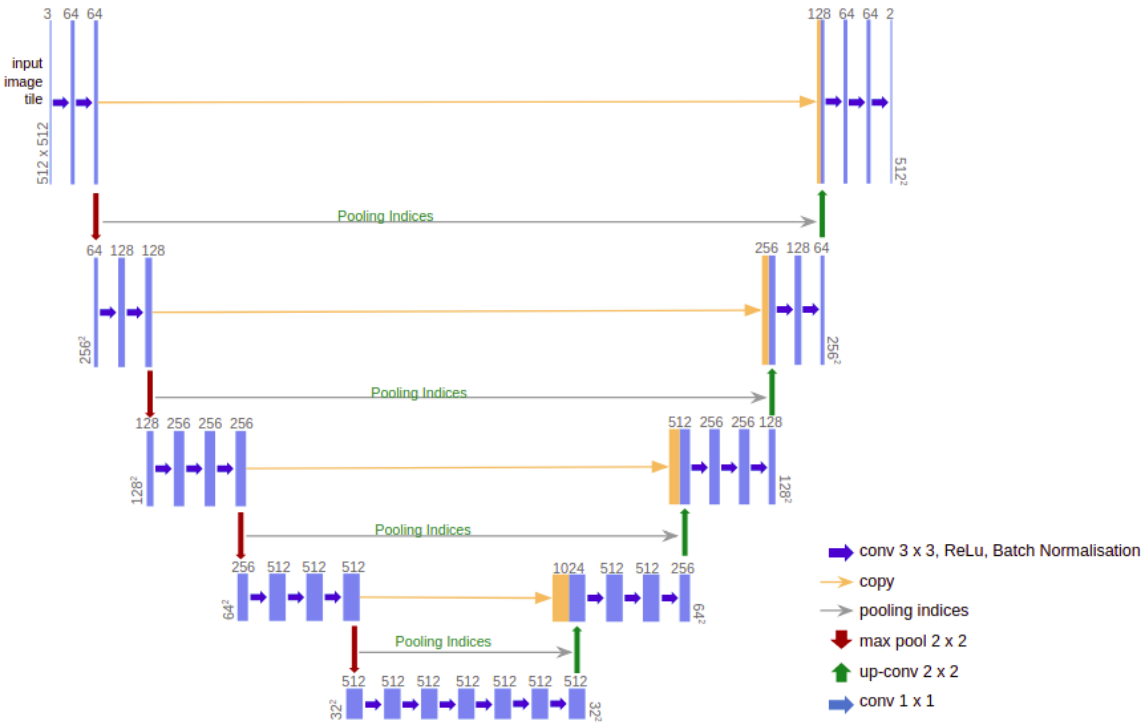

Figure 3.4: Architecture-1: using max-pooling indices to up-sample feature maps instead of transposed convolution.

As a result, this up-sampling method is adopted to improve the U-Net, and this new architecture is shown in Figure 3.4. This architecture is an encoder-decoder fully convolutional neural network. Typically, each convolution layers are followed by a ReLu and Batch Normalization. There are totally 4 max-pooling and corresponding up-sampling using pooling indices. The decoder has a symmetrical shape to the encoder, which is also consist of 10 convolution layers. For each block in the decoder, the feature maps first up-sampled from the output feature map of the previous block and then concatenate it with the feature map from the corresponding encoder. The pooling indices recorded before are used to produce a non-linearity up-sampling. Then in the next several convolution operation, we can half the output channels to let the final output channels of this block match the number of channels of the corresponding pooling indices. For example, the output feature map of the first block has 64 channels. Which means, the followed max-pooling operation which down-sampled

the feature map from 512 to 256 compute in these 64 channels while generating a 64 channel pooling indices. So where the up-sampling in the corresponding decoder, the feature map with the size of 256 need to be up-sampled to the size of 512, these feature map need to have same number of the channels as the pooling indices. The up-sampled feature maps from the previous decoder are concatenate with the feature maps which are the output of the corresponding encoder. These feature maps are served as the input of the decoder. Comparisons are made to show the improvement of this new architecture 1 to the U-Net and the SegNet. We will discuss it later.

3.3 Coordinate Maps

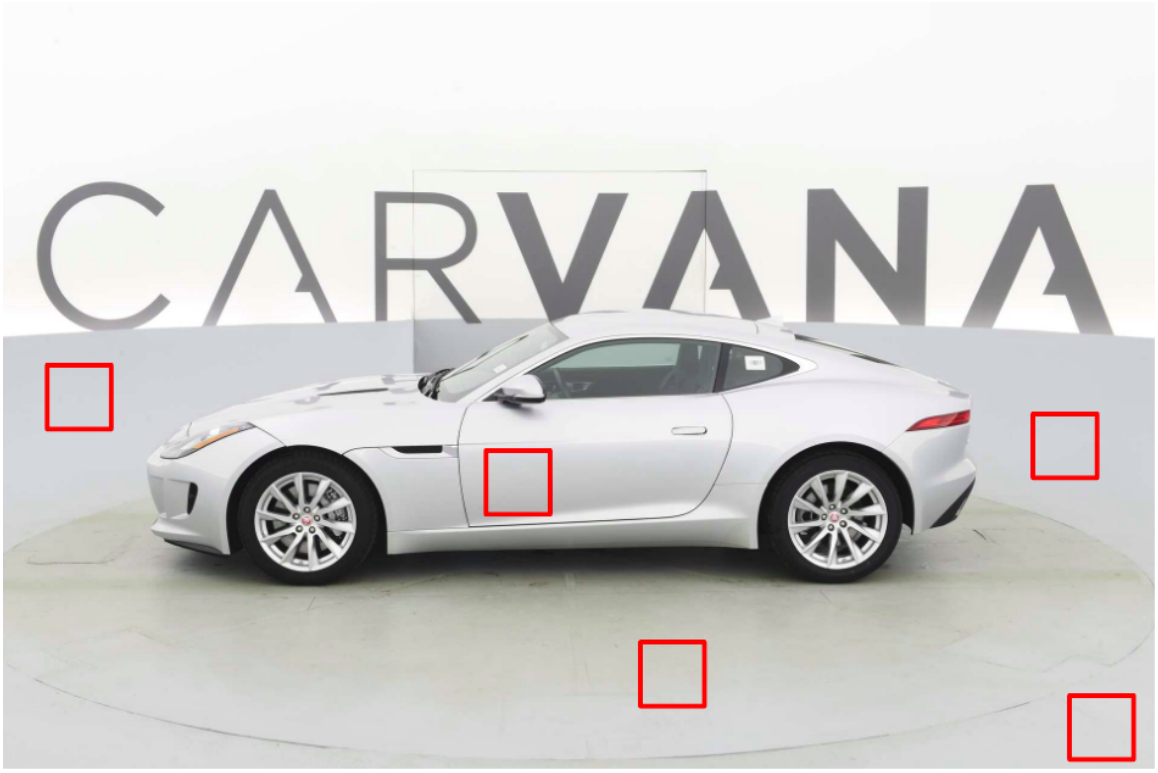

Figure 3.5: An example image shows that the content which can be from the car or the background in the red rectangles are almost the same.

[18] pointed out the straightforward stacks of convolutional layers are not quite the

right model to transform spatial representations between a Cartesian representation to a pixel-based representation. The authors solve this coordinate transform problem by adding extra, hard-coded input channels that consist of coordinates of the input image named CoordConv. Since for our dataset, the images in the dataset share the same size and the objects are always at the center of the image and share the same size. There are some correlations between the object and its location. For example, as the Figure 3.5 shows, the content in the red squares are almost the same. The square appeared at the center of the image is more likely from the body of the car, the squares appeared at the boundary of the image seems as the part of the background. Besides, certain part of the car will only appears at some certain locations. So, there is correlation between the content of image and their coordinates. The correlation can be learned by concatenate coordinate maps into feature maps. For the normal convolution with square kernel size k and with c input channels and c output channels will contain $cc'k^2$ weights. After adding coordinate maps with d channels, the number of input channels will be $c+d$, which lead to a increment of the parameters with $dc'k^2$.

The coordinate maps are basically composed of coordinates of the input image. As the Figure 3.6 shows, according to the input image(left image) size, x and y coordinate maps can be created. Each value in the x coordinate map shows the x coordinate of corresponding pixel of input image, and the same as y coordinate map. For example, the coordinate of the red point in the Figure 3.6 is (615, 450), then the value of the corresponding pixel in the x coordinate map is 615 and the corresponding pixel in the y coordinate map is 450. After the visualization of these coordinate maps, we will see a image with gradually changing color. All the images with the same size will create exactly same coordinate maps. However, the images will be cropped before being passed into the model. The green square in Figure 3.6 shows the crop of the image. The input image will be randomly cropped to size 512 x 512, then the coordinate maps will be cropped at the same location. Which means the coordinate maps of the

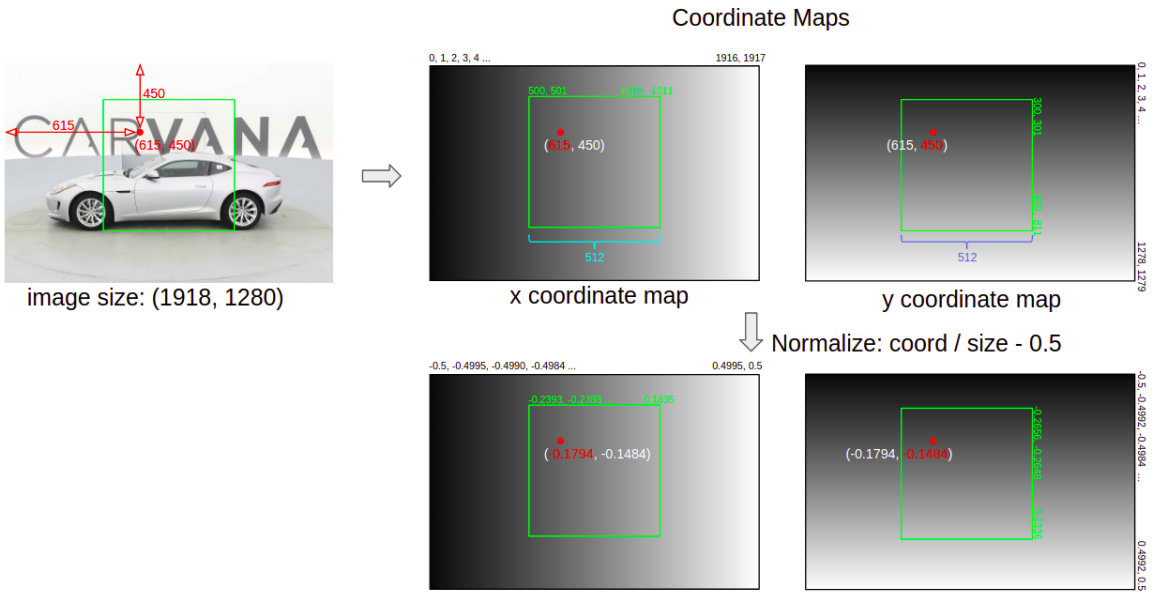

Figure 3.6: An example of how to generate the coordinate maps for the 512×512 subimage indicated by the green rectangle in the original RGB image. The red point in each image means the same point the position (615, 450). The first black white image is the x coordinate map whose value is the x coordinate of each pixel in the original image and the second one is the y coordinate map whose value is the y coordinate of each pixel in the original image. The location and the size of the green rectangles are all the same, means the random cropped image in the original RGB image and the corresponding crop in the coordinate maps. The images below show the coordinate map after the normalization.

input images are different because of the random crop. The normalization should be implement to the coordinate maps. The x/y coordinate map will be divided by the image width/height and then subtracted by 0.5, ranging the coordinate map between -0.5 to 0.5. Finally these coordinate maps will be concatenate to the feature maps to let the model learn the correlation between the content and its coordinate.

There are multiple ways to concatenate these coordinate maps, we can insert them into each one of these feature maps. Two typical locations are tried in our work, at the beginning and at the middle of the architecture. Figure 3.7 shows the architecture for concatenating the coordinate maps at the beginning. The cropped input image is concatenate with the x coordinate map and the y coordinate map making it a totally 5 channel image before pass it through the network. The remaining network is stay

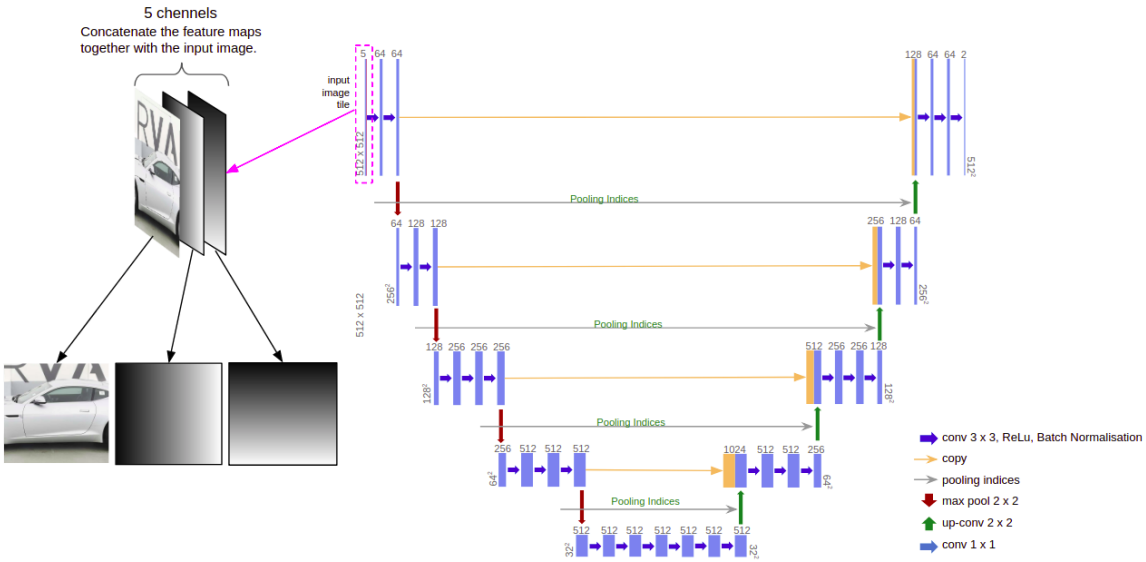

Figure 3.7: Architecture-2: The coordinate maps are concatenate to the input RGB image makes the input image 5-channels.

the same. The number of channels after this 3×3 convolution is 64, so we can get the increased amount of parameters is $1152(3 \times 3 \times 2 \times 64)$, which is a small increment.

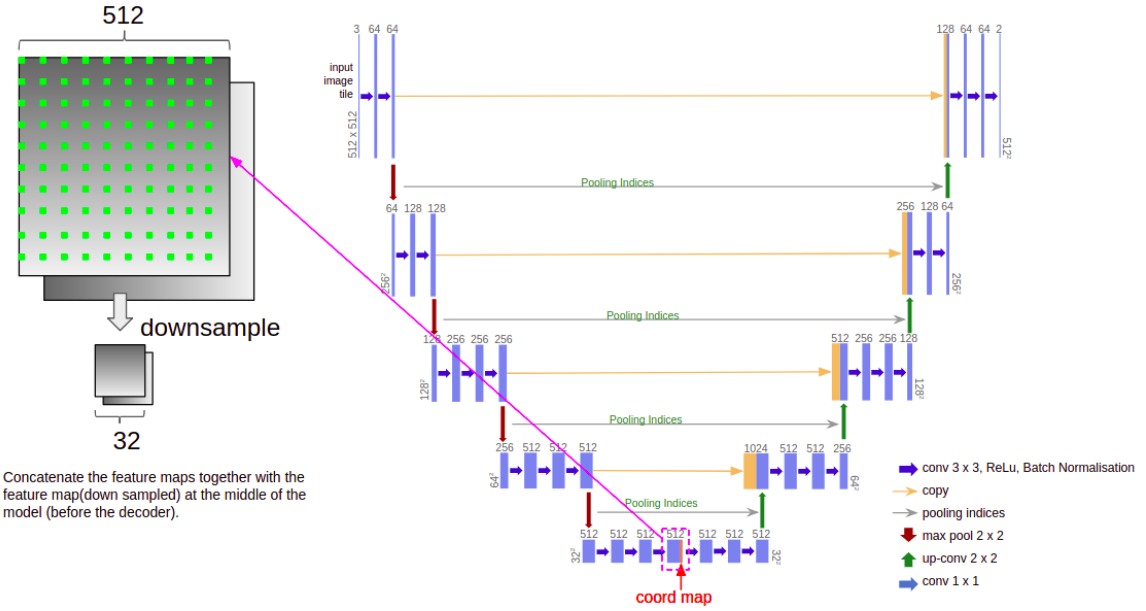

Figure 3.8: Architecture-3: Encode the coordinate maps into the feature maps at the middle of the model. The left part image shows an example of down-sampling the coordinate maps, makes the coordinate maps become size of 32×32, which is the same as the size of the feature maps in the middle of the model. Take a pixel for every 16 pixels to down-sample the 512×512 coordinate map into 32×32.

Figure 3.8 shows the architecture for concatenating the coordinate maps at the middle. As we see the architecture as the combination of an encoder and a decoder, the coordinate maps are concatenated to the feature maps which can be seen as the output of the encoder and the input of the decoder. The size of the feature maps at the center of the network is 32 x 32(the size of the input image is 512×512), because they are down-sampled by the 4 max-pooling. The size of coordinate maps used to concatenate with these feature maps is also need to be 32×32. Each feature vector in feature maps at the middle of the network can be seen as extracted from the around area in the corresponding location in the input image. So the 512×512 to 32×32 down-sampled coordinate maps can be obtained by taking one pixel for each 16 pixels. Then the coordinate maps are concatenated with the feature maps in the middle of the network. We can also calculate the increment of the parameters, which is $9216(3 \times 3 \times 2 \times 512)$, since the number of channels after this 3×3 convolution is 512. Experiments are made to see the performance of both of the networks, and an improvement can be seen using coordinate maps.

3.4 Dilated convolution

[14], [15], [16] demonstrated the importance of global information used the features. Since the large object in the image and the limited number of convolutional layers and max-pooling layers, the receptive field is not large enough. The evidence is the existence of holes inside of the car body or the missing part of the car, see Figure 3.9. Because the area around these parts are correctly predicted, these mistakes are mainly due to the limited receptive field. The architecture can be improved by enlarge its receptive field to encode more richer features with more global information.

As [18] pointed out that dilated convolution(atrous convolution) can be used to enlarge the field of view of filters without increasing the number of parameters or

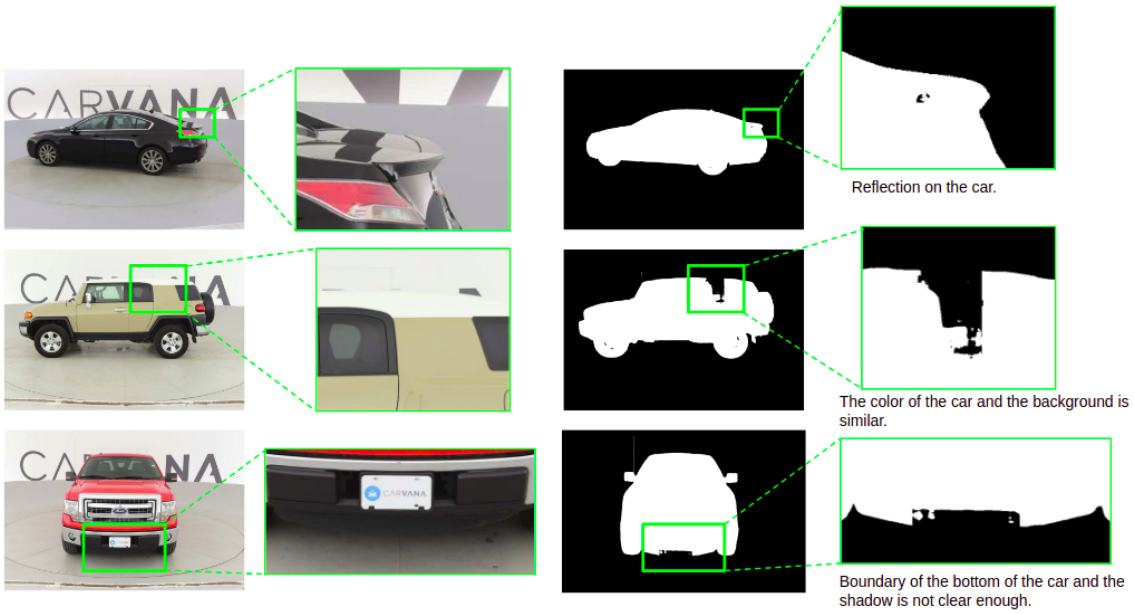

Figure 3.9: Some typical prediction error in the images for the limited receptive filed.

0	1	2
2	2	0
0	1	2

Figure 3.10: The kernel used to illustrated the process of the normal convolution and the dilated convolution.

the number of computation. Figure 3.11 shows an example of normal convolution. Figure 3.12 shows an example of dilated convolution. Both of the two examples use the kernel shows in Figure 3.10. At each location, an element-wise product is computed between the kernel and the corresponding shaded area and the results are summed up to get the value of the output in this location. Because the dilation which controls the spacing between the kernel points, a output feature map with the same size as the input can be obtained by setting the padding size as the same as the dilation. For example, when dilation = 2(the spacing between the kernel points is 1), then $padding = 2$. We can find that the normal convolution and the dilated

convolution with $dilation = 1$ are equivalent.

In our work, the normal convolution in the middle of the architecture are replaced with the dilated convolution as Figure 3.1. Different dilation sizes are set in these convolution layers. Because there are 6 convolution layers, the $dilation = 2^{i-1}$ where i is the index of the convolution are set. The dilation for the final convolution and the size of the feature maps are both 32, which means the final feature maps have the full receptive field. When these feature maps are concatenate with the feature maps from the encoder, a richer features can be created with more global information and local information. Comparison also made to see the influence of these dilated convolutions.

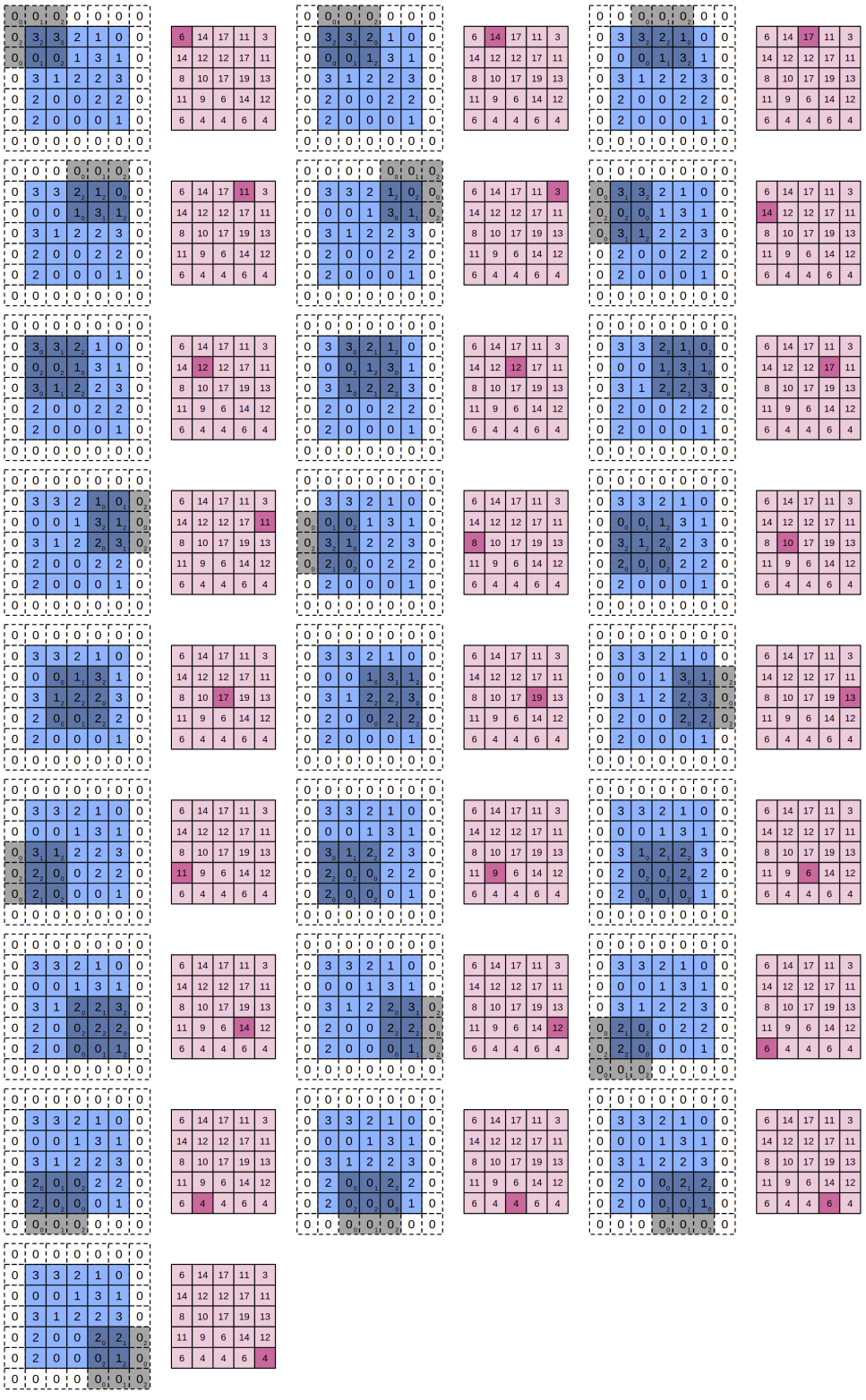

Figure 3.11: An example of normal convolution with $kernel\ size = 3 \times 3, padding = 1, stride = 1 \times 1$

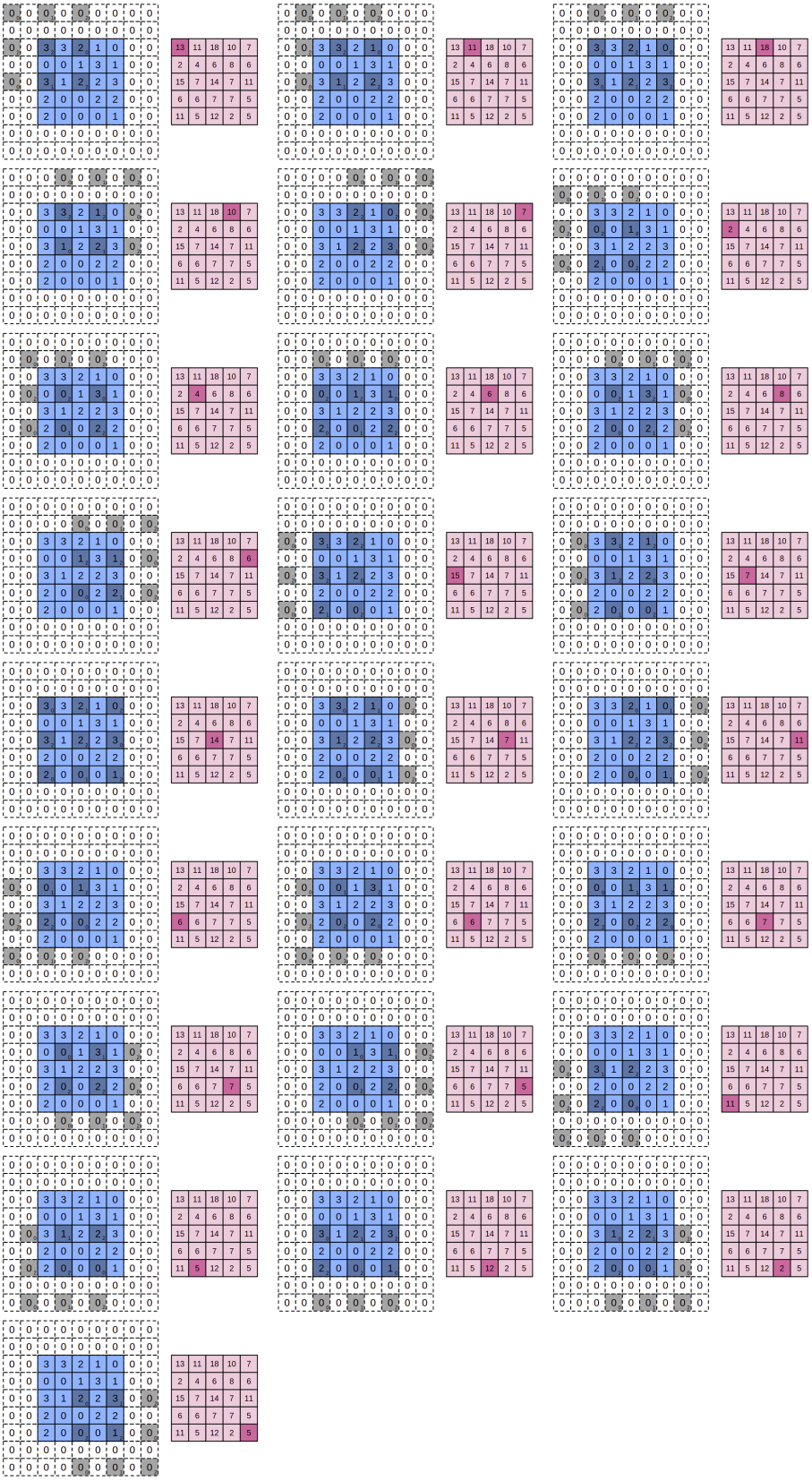

Figure 3.12: An example of dilated convolution with $kernel\,size = 3 \times 3, dilation = 2 \times 2, padding = 1, stride = 1 \times 1$

Chapter 4

Experiment

4.1 Training

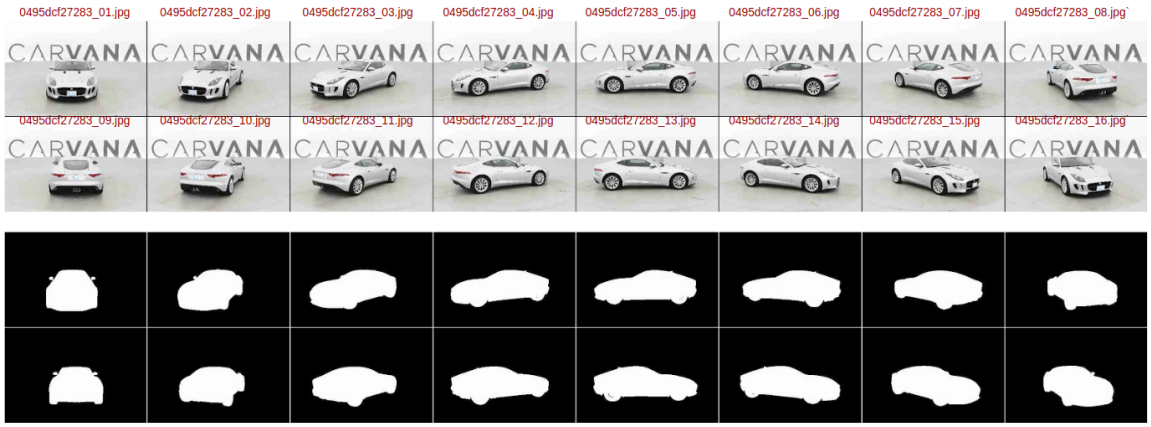

Figure 4.1: Example images of a car in the dataset with 16 different angles. Each image is named as id_angle.png which is shown as the red text above each image.

We use Carvana Image Masking dataset[kaggle dataset] to benchmark the performance of our architecture. This dataset is large, containing 5088 training and 100064 testing RGB images. These images all share the same resolution of 1912 x 1280. The task of the challenge is to automatically segment the car in the test images. Some images in the test set with car images are ignored in scoring to avoid of hand label-

Figure 4.2: Example images of different cars with the same angle in the dataset.

ing. So the challenge is to segment 2 classes: car and the background. These images are taken in a custom rotating photo studio which can automatically captures and processes 16 standard images with 16 standard view angles of each vehicles in the inventory, like Figure 4.1 shows, the above RGB images are the input images and the black white images below are result images. The dataset of photos covers different kind of vehicles with a wide variety of color, make, year, model combinations, some examples show in Figure 4.2. The position and view angle of the camera are fixed, so that the cars are always at the center of the images. The distance between the camera and the cars are same, which means the scales of the cars are similar.

All the weights in the convolutional layers are initialized using the method called kaiming uniform, which is described in [19]. Because the first 10 layers of our architecture are the same as the first 10 layers of VGG-16. For the weights in the first 10 layers, we initialize the weights using VGG-16 which is pre-trained in the large dataset ImageNet.To train all the variant of architectures, we use Adaptive Moment Estimation (Adam) which computes adaptive learning rates for each parameter. We set the initial learning rate of 0.001 and halving the learning rate for each 40 epochs and set the momentum of 0.9 using framework Pytorch. We splitted 1017 images from the original 5088 images in the training set to make a validation set, then we have a new training set with 4071 images and validation set with 1017 images which is the ratio of 4 to 1. We train 250 epochs for all the networks, with the batch size of 4 images. The images are cropped to 512 x 512 and shuffled before they are divided into mini-batches ensuring that each image appear only once in each epoch. We select

the model which performs best in the validation set.

The final layer of all the architectures in our work is a convolutional layer mapping a multi-channels feature map into 2 channels feature map with channel 0 corresponding to the background and channel 1 corresponding to the car. To compute the loss, we use a pixel-wise soft-max function combined with a cross entropy function. The pixel-wise soft-max function calculating the probability of each pixel belongs to the background or the car defined as

$$p_{car}(x) = \frac{e^{a_1}(x)}{e^{a_0}(x) + e^{a_1}(x)} \tag{4.1}$$

$$p_{bg}(x) = \frac{e^{a_0}(x)}{e^{a_0}(x) + e^{a_1}(x)} \tag{4.2}$$

where $a_0(x)$ denotes the activation in feature channel 0 at the pixel position $x \in \Omega$ with $\Omega \subset Z^2$ and $a_1(x)$ is the activation in feature channel 1 at the same pixel position. pcar and pbg is the probability of the pixel at position x belongs to the car and the background, and $p_{car} = 1 - p_{bg}$. The cross entropy penalizes at each position the deviation of pcar or pbg from 1 when the pixel is from the car or the background using

$$E = -\frac{1}{N}\sum_{x\in\Omega} ylog(p_{car}(x)) + (1-y)log(p_{bg}(x)) \tag{4.3}$$

where y is the true label of each pixel. When $y = 0$ at pixel position x, we calculate $log(p_{car}(x))$, and when $y = 1$ at pixel position x, we calculate $log(p_{bg}(x))$ instead. Then we compute the average of them to get the final loss for this prediction. Then we compute and back-propagate the average loss for all the predictions in the mini-batch to optimize our weights.

4.2 Analysis

The evaluation metric used in the challenge of this dataset is the mean Dice coefficient. It can be used to compare the pixel-wise agreement between a predicted segmentation and its corresponding ground truth. The mean Dice coefficient is defined as

$$DSC = \frac{2 \times |X \cap Y|}{|X| + |Y|} \tag{4.4}$$

where X is the predicted set of pixels and Y is the ground truth. The Dice coefficient is defined to be 1 with both X and Y are empty. Because of the specialize of this problem, there are only 2 classes: the car and the background and it is relatively easy to tell the difference between the car and the background raising a problem that it is critical to get the precise boundaries of the car. Because of the fact that most of the pixels can be easily predicted, the mean Dice coefficients of different models all looked very high such as 99.56%, 99.60%, the comparison between the performances of the models will rely on the values after the hundredths place. To intuitively know the difference, we estimate the car in the image approximately occupied 600,000 pixels, then the difference of 0.01% is approximately 60 pixels. So the score with 0.01% higher means there are approximately 60 more pixels predicted correct. What's more, because there are two main difficulties of the segmentation in this dataset. First, the boundaries of the white or gray cars are not clear enough, because these color are too similar as those from the background. The white cars whose color is similar as the background is much harder to find the boundaries than the cars with other colors. Second, the car in different types are imbalanced. Most of the cars in the dataset are mid-size cars, and there are also a few of pickups and some other types of cars. So the model is more good at to find the boundaries of the mid-size cars. The error of 60 pixels/car will not evenly distributed in the images, instead, most of the images are looks quite good, but there are some unacceptable error appear in a small part

of the images. Thus, difference of the value after the hundredths place of the score computed by the mean Dice coefficient is also very important.

We trained and test all of the architectures discussed before. Since the ground truth of the test set is not released, we submitted our result, and the competition holder(Kaggle) would calculate the score which is the mean Dice coefficient for us. For each architecture, we select the model which has the highest mean Dice coefficient in the validation set to make the prediction on the test set. All the models are trained using a NVIDIA 1080Ti GPU with cuDNN v7 acceleration.

In Table 4.1 we report the numerical results of the standard U-Net, SegNet and our architecture 1. We also show the size of the trainable parameters and the time cost for both training and testing. From the result, we find that the score improved 0.1328% and 0.084% compared to the standard U-Net and SegNet, which means our modification of the up-sampling method indeed helps the model to generate more accurate segmentation without too expensive additional time consumption. To compare with the U-Net, we find that the up-sampling using the pooling indices from the Architecture-1 performs better than the transposed convolution from the U-Net. To compare with the SegNet, we find that the concatenating of the feature maps from the encoder helps the feature maps gain more detailed information thus lead to a better performance.

Model Name	DSC-Test set	#Parameters	Training Time	Testing Time
U-Net	99.5604%	31,043,586	137.6 ms/image	34.421 ms/image
SegNet	99.6092%	29,444,162	165.13 ms/image	30.50 ms/image
Architecture-1	**99.6932%**	32,799,554	171.75 ms/image	42.793 ms/image

Table 4.1: Result comparison between the standard U-Net and SegNet and the Architecture-1. This table shows the improvement of our Architecture-1.

In Table 4.2 we also compare the performance of the Architecture-2, which concatenate the coordinate maps at the beginning of the model and Architecture-3, which concatenate the coordinate maps at the middle of the model. The experiment shows,

the architecture-3 performs best. Although the scores are seem very close, there are still some improvement in a very small part of images. When the features are difficult to be classified, the coordinate maps can help to make the prediction according to the spatial location of the features. As the examples show in Figure 4.3, the images at the middle are generated by Architecture-1, and the images at the right side are generated by Architecture-3. We can see in the images generated by Architecture-1, there are some holes in the middle of the car or some error at the top of the images, however, these mistakes are corrected by the mask generated by Architecture-3. Because areas which are located at the middle of the images are very likely belong to the car, and the areas at the top of the images are more likely to be from the background. The coordinate maps encoded the coordinates into each of the features help the models perform better in these problems.

Model Name	DSC-Test set	#Parameters	Training Time	Testing Time
Architecture-1	99.6932%	32,799,554	171.75 ms/image	42.793 ms/image
Architecture-2	99.6870%	32,800,706	172.59 ms/image	42.901 ms/image
Architecture-3	**99.6937%**	32,808,770	170.50 ms/image	42.839 ms/image

Table 4.2: Result comparison between the Architecture-1, Architecture-2 and Architecture-3. This table shows the improvement of encoding the coordinate maps into the middle of the model.

Table 4.3 shows the comparison between the Architecture-3 and the Final Architecture. Even though the results above is very encouraging, there are still exist some other problems shown in Figure 3.9. These mistakes are often caused by the unclear boundaries in the original image. Because we can't see the boundaries clearly, when we manually do the segmentation, we will guess them by the shape of the car or the experiences learned from other cars. After we enlarge the models' receptive field, the models will learn to draw the boundaries like human beings and the mistakes are fixed show in Figure 4.4. The Final Architecture which using dilated convolution perform the best without any additional parameters.

We make the summation for the above analysis and comparison, and shows that:

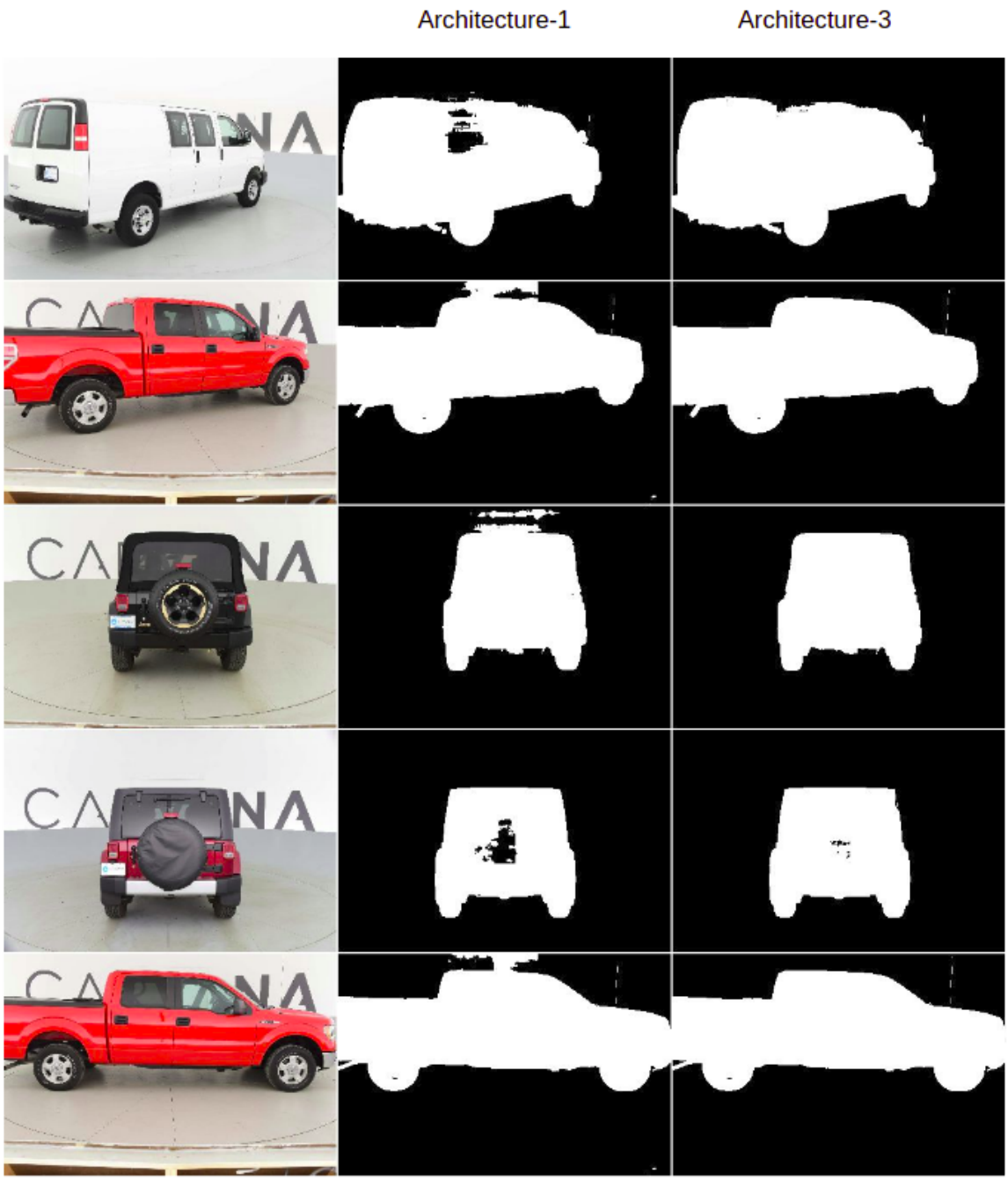

Figure 4.3: The result comparison images of how the coordinate maps help to rectify the errors. The middle predictions are generate by Architecture-1, which don't encode the coordinate maps. The right predictions are generated by Architecture-3, which encode the coordinate maps into the middle of the model.

1. The combination of the concatenating of feature map from the encoder and up-sampling using pooling-indices helps the decoder generate features with more

Model Name	DSC-Test set	#Parameters	Training Time	Testing Time
Architecture-3	99.6937%	32,808,770	170.50 ms/image	42.839 ms/image
Final Architecture	**99.7023%**	32,808,770	180.46 ms/image	46.609 ms/image

Table 4.3: Result comparison between the Architecture-3 and our Final Architecture. This table shows that our Final Architecture is the best model.

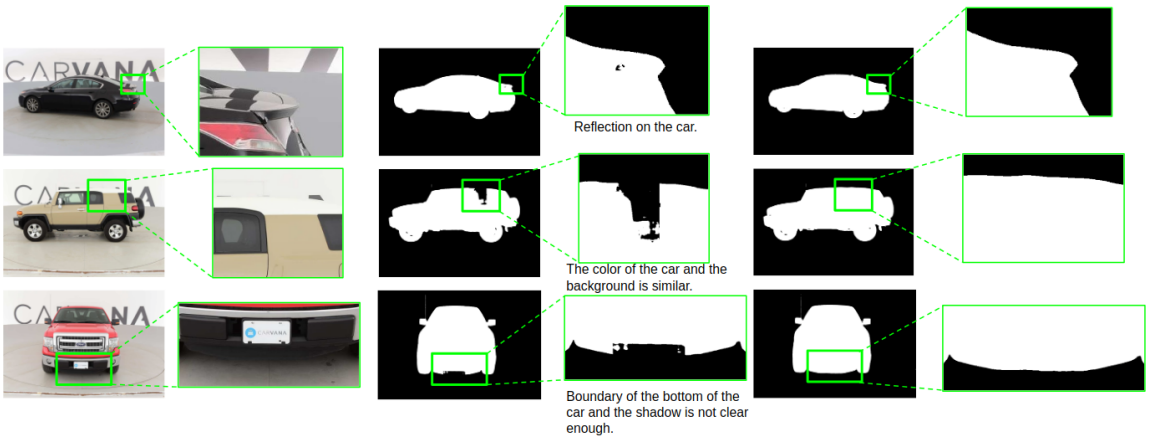

Figure 4.4: Some example images show that the mistakes of the Architecture-3 are rectified by our Final Architecture. These images shows the importance of the enlarge the receptive field.

accurate spatial information.

2. The coordinate maps which are encoded into the features help break the translation invariant of the convolution. Thus the same pattern appears in different places will be classified to different categories.

3. The dilated convolution helps the model to enlarge the receptive field and learn more contextual information. As a result. the dilated convolution can help to generate the boundaries which are not even clear in the original images using these contextual information.

Chapter 5

Discussion and Future Work

5.1 Discussion

As it is shown above, we trained both the U-Net and the SegNet with the scores of 99.5604% and 99.6092%. It seems strange the SegNet performs better than the U-Net, because even though the reuse of the pooling indices in the up-sampling of the SegNet can prevent the loss of the spatial information, the U-Net who concatenate the feature maps from the encoder can keep even more detailed information than SegNet which containing the loss spatial information due to the max-pooling. Since the feature maps which are concatenated to the decoder in the U-Net, are from the activations which are just before the max-pooling layers, means that these features contain all the information including the max pooling indices. The U-Net should be better than the SegNet. The problem of the bad performance of the U-Net is due to the Batch Normalization and the convolution used in the model without any padding.

The authors of U-Net use convolutions without any padding, so the input size and output size of the feature maps of each convolution are different. Since the kernel size used in these convolutions are 3×3, if the size of the input feature map is $C \times W \times H$,

then the size of the output feature map will be $C \times (W-2) \times (H-2)$. That is, the model only predict the pixels, whose full context is available in the input image. Figure 5.1 shows an example image of the input size indicated by the green rectangle and the output size indicated by the rectangle. The task of [U-Net] is biomedical segmentation, the images in the dataset have no direction. So the authors make mirror padding when predicted the images around the borders. But mirror padding is not make sense in our task, the car is never upside down. So zero padding is used instead, like the right part of Figure 5.1.

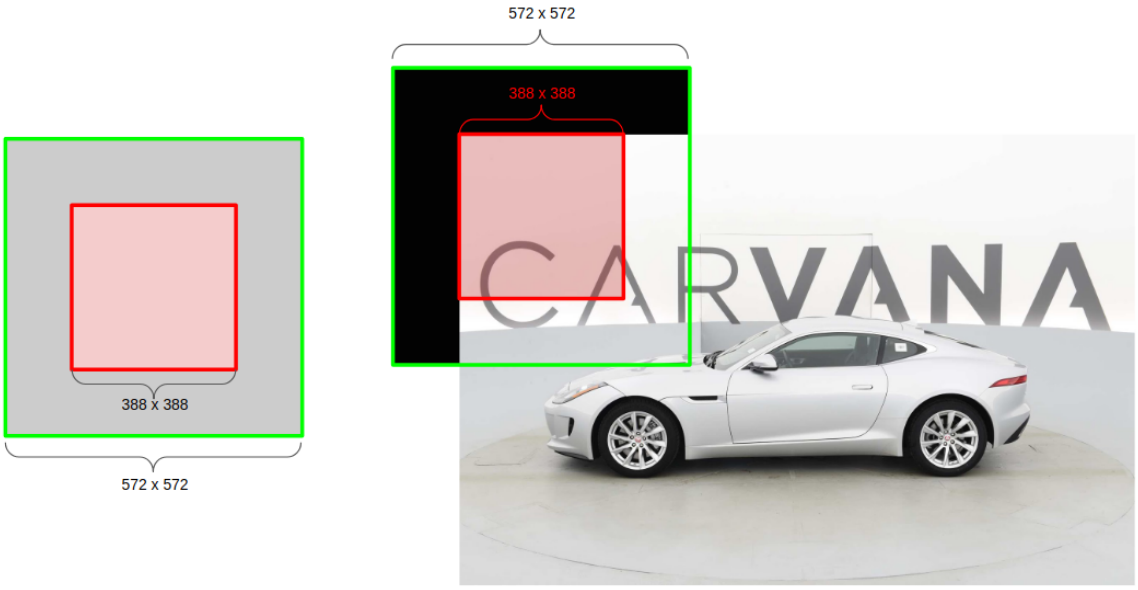

Figure 5.1: Examples to show the different sizes of the input and output images of the U-Net. If we want to predict the image in the red rectangle, the larger image shows as the green rectangle should be input into the model. The right part image shows if we want to predict the red image which located at the boarder of the original image, zero padding which is shown as the black area is needed to enlarge the input image to the desire size.

During training, the Batch Normalization use the mean and variance calculated from the mini-batch to normalize the activations. While testing, the Batch Normalization use the mean and variance calculated by the moving averages when the model training which can be seen as the estimation of the mean and average over the whole dataset. Due to the limited memories of GPU, the batch size we use during training

is 4, which is very small. Because of the zero padding we discussed above, the mean and variance will be very different according to the location to cropped images. So when inference, bad performance will occur because of using the global mean and variance which is very different from the mean and variance of the mini-batch. Figure 5.2 shows some example, there are some strange error indicated in red rectangles. Even we use the mini-batch mean and variance instead of global ones, there are still error such as Figure 5.3. The reason is also because the means and variances used in training and test differ greatly. One of a good way to solve this problem is using big batch size while training, however the it will cost too much memories which will beyond the capacity of the GPU. Another way to solve this problem is to use padded convolutions instead of unpadded convolutions, therefore the mean and variance will be steady. But even we change to the padded convolutions, the U-Net which doesn't use pre-trained weights also performs worse than the SegNet. So in our work, the structure of the encoder which is identical to the VGG-16, can use the pre-trained weights to improve the performance.

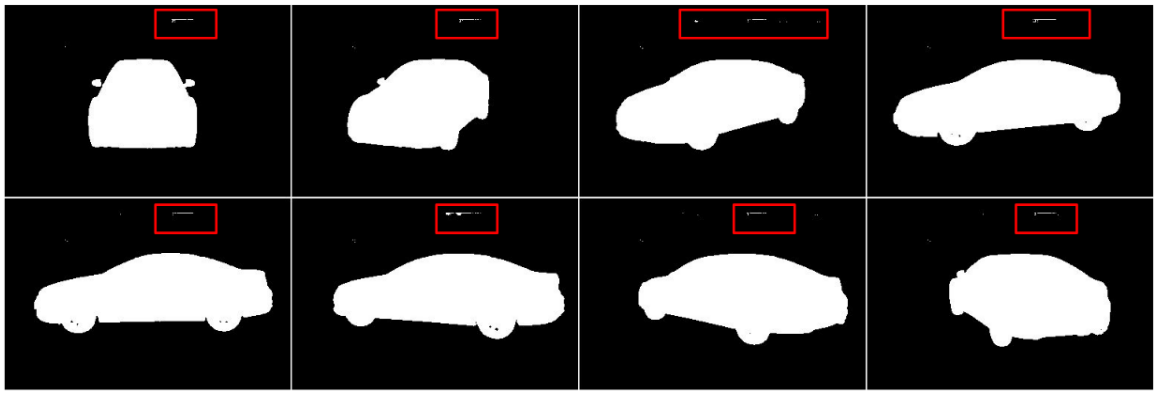

Figure 5.2: Example images of the mistakes of the U-Net prediction. Mistakes in red rectangles are shown when we use the mean and variance calculated by the moving average in the Batch Normalization when inference.

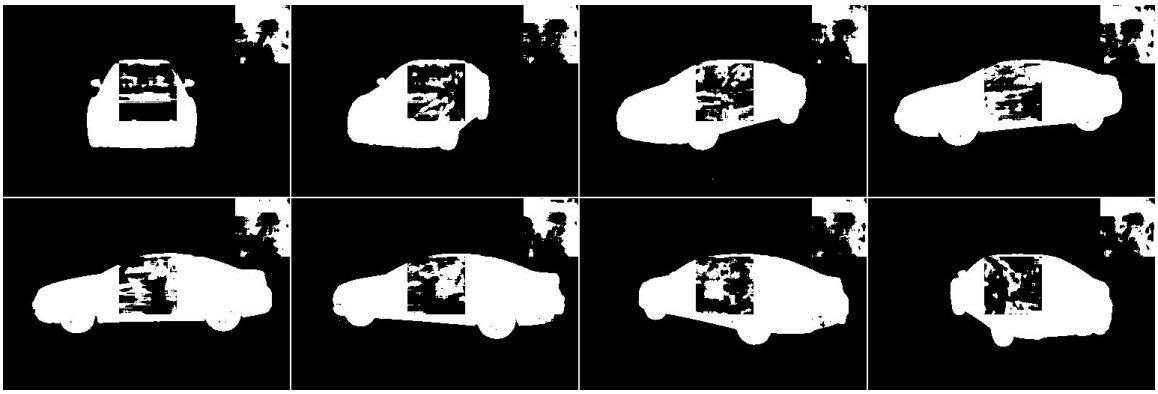

Figure 5.3: Example images of the mistakes of the U-Net prediction. Mistakes of strange blocks are shown when we use the mean and variance calculated by the mini-batch in the Batch Normalization when inference.

5.2 Future Work

In the future we would like to make use of Generative Adversarial Networks (GANs) to improve the performance. Currently we have two ideas:

1. For the future, we would like to try to use our proposed model as a generator, and add another CNN-based classifier as a discriminator to form adversarial networks [20].

2. Train GANs independently with the proposed model. Then the Discriminator may be used to detect the possible misclassified area. And then those possible misclassified areas could be rectified by some post-processing methods or more complicated models (such as CNN with larger receptive fields or ensemble the results from multi-scale image pyramid).

Chapter 6

Summary

We proposed a end-to-end learning deep fully convolutional neural network architecture for semantic segmentation. The motivation behind this architecture is the need to segment large objects in the high resolution images. We analyses our architecture with other benchmarks and some variants to demonstrate the effectiveness of three mainly modification: 1. Concatenating the feature map from the encoder to the decoder while using the max-pooling indices to up-sampling the feature map from the previous block in the decoder. 2. Encoding the coordinate maps into the feature map. 3. Using the dilated convolution instead of traditional convolution. Our work performs competitively and achieves higher scores. We are sure that our proposed architecture can be applied to many semantic segmentation tasks.

Bibliography

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[3] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[4] Paul Sturgess, Karteek Alahari, Lubor Ladicky, and Philip HS Torr. Combining appearance and structure from motion features for road scene understanding. In *BMVC-British Machine Vision Conference*. BMVA, 2009.

[5] L'ubor Ladickỳ, Paul Sturgess, Karteek Alahari, Chris Russell, and Philip HS Torr. What, where and how many? combining object detectors and crfs. In *European conference on computer vision*, pages 424–437. Springer, 2010.

[6] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *Computer vision and pattern recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[7] Gabriel J Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *European conference on computer vision*, pages 44–57. Springer, 2008.

[8] Peter Kontschieder, Samuel Rota Bulo, Horst Bischof, and Marcello Pelillo. Structured class-labels in random forests for semantic image labelling. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2190–2197. IEEE, 2011.

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[10] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, pages 297–312. Springer, 2014.

[11] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 328–335, 2014.

[12] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[13] Mohammadreza Mostajabi, Payman Yadollahpour, and Gregory Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3376–3385, 2015.

[14] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2013.

[15] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 447–456, 2015.

[16] Wei Liu, Andrew Rabinovich, and Alexander C Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015.

[17] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[18] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. *arXiv preprint arXiv:1807.03247*, 2018.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.