Removing Background with Semantic Segmentation Based on Ensemble Learning

Junhong Xu, Hanqing Guo, Aaron Kageza, Saeed AlQarni, Shaoen Wu
{jxu7, hguo, agkageza, saalqarni, swu}@bsu.edu

Abstract. This paper presents a deep learning approach to Kaggle Carvana Image Masking Competition, which aims at extracting the car objects in high quality images with the background removed. We formulate the background extraction problem as an image segmentation problem. In this challenge, we have evaluated different U-Net architectures. We have explored two different techniques in combining encoder downsampling features with decoder upsampling features. In addition, we have experimented replacing different pre-trained networks to accelerate the training process. Finally, we have trained different models at different image scales and predicted the final result with the ensemble method. Our final method has placed us at top 4% in the challenge and achieved a dice coefficient score of 0.99694.

Keywords: Background removal, deep learning, semantic segmentation

1 Introduction

Background subtraction is an important technique in surveillance, video analysis, and many other applications. This problem can be formed as a semantic segmentation problem where each pixel is classified into either object or background class. Figure 1 shows a picture of image segmentation in the car masking challenge.

Most approaches to image segmentation in the past are based on conditional random fields(CRF) that leverages the relationship between pixels [6]. Recent years, deep learning has dominated computer vision, speech recognition and many other areas. One popular network architecture is convolutional neural networks(CNNs) that can learn discriminate features from image data automatically [9]. Many network architectures have been proposed to improve the learning ability such as VGG [15], Inception network [17], and ResNet [4]. There are many existing work that address image segmentation using variants of these network architectures which have drastically improved segmentation results compared to traditional methods. This paper focuses on utilizing and modifying these existing network architectures to tackle Kaggle Carvana Image Masking Competition which requires to extract cars from background in high resolution images. This competition provides several challenges

MOBIMEDIA 2018, June 21-22, Qingdao, People's Republic of China
Copyright © 2018 ACM
DOI 10.4108/eai.21-6-2018.2276586

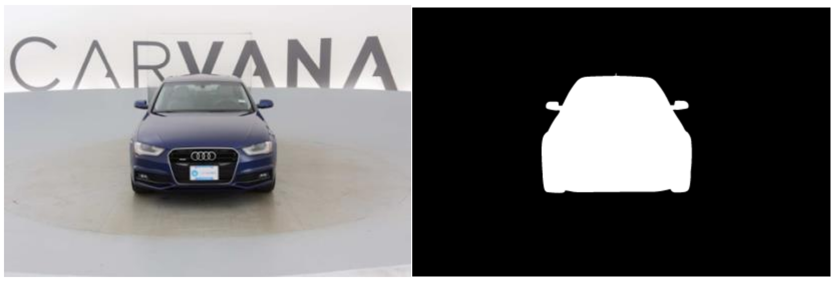


Fig. 1. Example of image segmentation in the car masking challenge.
The left image is the image to be segmented and the right one is the segmentation.

to the existing work: 1. although high resolution images provide finer details of objects, directly applying deep learning models on high resolution requires high computational cost which slows down training speed and 2. it requires the network to predict an accurate image mask which is challenging e.g. in Figure 1, differentiating the shadow from the chassis. To address these challenges, we propose four variants of U- Net [14] and RefineNet [10] with different pre-trained CNN models [4], [15] to speed up training as well as retain accurate predictions. In addition to binary classification loss, we also use focal loss [11] and soft dice loss to stabilize training process. To alleviate high computational cost and keep accurate predictions, the four networks are trained on different scales of images: 1280 × 1920, 1280 × 1536, 1280 × 1280, 1024 × 1024 respectively. The final predictions on the test set are produced by the ensemble of these models which placed us at rank 29 out of 735 participants.

In our experiments, we analyzed the performance of single network during training on training and validation sets. We also checked the performance of each model qualitatively by visualizing segmentation results.

In Section II, we briefly introduce deep learning and image segmentation. Section III gives detailed explanations of our network architectures, training process, and loss functions. Experiments and final results are given in Section IV. Finally, Section V summarizes the proposed approach.

2 Related Work

This section discusses popular CNN architectures and deep learning in semantic segmentation.

AlexNet [8] was the pioneering deep learning architecture that won the ILSVRC-2012 image classification challenge. It consists of five convolutional layers with max-pooling layers and nonlinearities. Three fully-connected layers are applied to predict class predictions. VGG-16 [15] used a stack of convolutional layers and small receptive fields compared to AlexNet. This allows the model to have less parameters, but more layers with nonlinear transformations which makes the model learn more discriminative features. ResNet [4] builds upon residual blocks that allow training a deeper model with more than 100 layers. The residual block uses identity mapping to learn the residual between input and output. Our work utilizes pre-trained ResNet and VGG-16 on ImageNet dataset [3] in the encoder. The usage of pre-trained neural networks is called transfer learning and it is proven that transfer learning works better than random initialization [19].

Semantic segmentation has been extensively studied in the literature. It is a dense classification or per-pixel labeling problem where each pixel in the input image is assigned with a class label. Fully convolutional network (FCN) has been proposed to use convolutional layers to replace all fully-connected layers to produce a per-pixel prediction map instead of an array of class predictions [12]. It leverages the ability of CNNs as powerful models that are able to learn abstract features from input images. Because CNNs use downsampling method that reduces the spatial dimension, it loses important details such as boundaries that are crucial for semantic segmentation. This makes FCN not able to output accurate segmentation maps. To overcome this problem, encoder-decoder variants of FCN are proposed such as U-Net [14], SegNet [1], and RefineNet [10]. Instead of using downsampled feature maps directly for prediction, these models combine them with feature maps generated from decoder upsampling step. Another line of work uses dilated convolutions [20] and conditional random field(CRF) to sovle the problem of loss of crucial features. Dilated convolutions expand the convolution filters exponentially that make the use of upsampling features. DeepLab uses this operation and CRF as post-processing [2].

3 Methodology

3.1 Segmentation Models

This section shows how we rebuild U-Net encoder with two different pre-trained models in Section III-A1. In addition, we also explore two different techniques to combine encoder and decoder feature maps which is explained in details in Section III-A2. The overall architectures of our models are shown in Fig. 5. Since VGG-16 and ResNet-50 both have five blocks, Fig. 5 represents a unified version of our model.

1) Encoder: We use ResNet-50 and VGG-16 as encoders in our segmentation models as shown in Fig 5. We discard all fully-connected layers in the pre-trained models because they are used for image classification. Encoder networks are used for extracting features from an input image. These features are called feature maps. They are produced by sets of convolution, batch normalization [7], non-linear operations, and downsampling operations (spatial max pooling or strided convolution operation). These feature maps can be viewed as low-dimensional representations of the image that is separable by a linear classifier in image classification tasks. The decoder network upsamples the feature maps to predict the final segmentation result. Since these feature maps do not retain accurate boundary details because of downsampling operations, we use two different operations in decoder network to overcome this lossy representation [1].

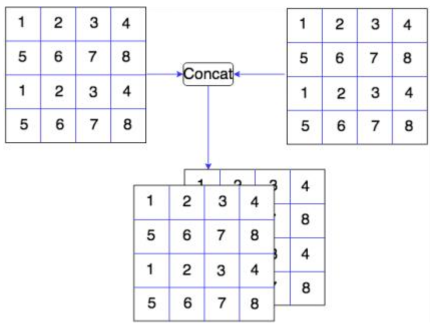


Fig. 2 Example of concat operation on two 1-channel feature maps resulting a two-channel feature map.

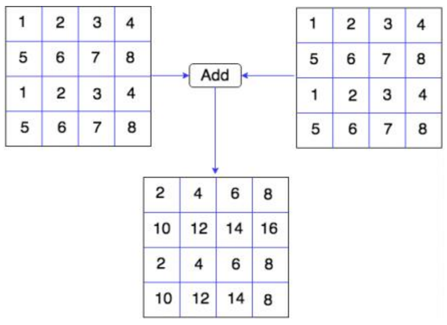


Fig. 3 Example of add operation on two 1-channel feature maps resulting a one-channel feature map which values are addition of the two input feature maps.

In all models, Convolutional Layer is a stack of convolution, batch normalization, and ReLU [13] operations. In ResNet-50, the first layer is a Convolutional Layer followed by max-pooling layer. After the first layer, there are four ResNet blocks with variant numbers of residual blocks. For a detailed architecture explanation of ResNet layers, please refer to [4]. Instead of using max pooling layers to reduce feature map spatial dimensions, ResNet uses strided convolution. This replacement shows the improved performance empirically in [16].

VGG-16 variant of the encoder consists of five blocks with 13 convolutional layers. Max pooling operation is applied between each block to reduce the spatial dimension of the feature maps. In both models, the feature map produced by the last encoder layer is fed into a middle Convolutional Layer with 1×1 convolution operation to increase the depth of the feature map but reserve the spatial dimension [18].

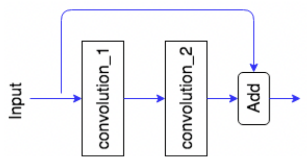


Fig. 4: A residual convolutional unit.

2) Decoder: Decoder is used for upsampling feature maps produced from the middle layer and output the final segmentation result. Instead of using single convolutional layers in the decoder, we use residual convolutional unit (RCU) as shown in Fig.4. It has been shown that adding an identity mapping allows training deeper networks and overcome vanishing gradient problem [5]. After each residual block, the channel size of each feature map is reduced by a factor of 2, but the spatial dimension is increased by the same factor due to upsampling. The last layer in the decoder is a convolutional layer with 1×1 convolution operation followed by a sigmoid operation. It produces a single channel image. Each pixel in the image represents the probability of it belongs to a car. The final segmentation map is produced by thresholding the probability at each pixel. Because each encoder downsampling step has one corresponding decoder upsampling step, we preserve the dimensionality at the final dense segmentation result. In order to reduce the effect of lossy feature representation caused by downsampling, we combine encoder and decoder features. In this paper, we explore two different strategies of combing these features: adding or concatenating. Fig. 3 and 2 illustrate these two strategies. Concatenation operation concatenates two feature maps along the depth dimension resulting in a new feature map that has the number of channels equal to the addition of the number of channels of the two inputs. Add operation adds the values of each feature map element-wise. In Section IV, we analyze the different performance of these two operations.

3.2 Multi-Loss Training

Instead of using a single binary classification loss to train our models, we combine three losses together to stabilize and accelerate training process.

1) Binary cross entropy loss: since segmentation is a dense classification problem, we use the standard binary cross entropy loss as the primary loss function. It is written as

$$\mathrm{BCE} = (-\mathrm{y}\log(\mathrm{p}) + (1-\mathrm{y})\log(1-\mathrm{p})), \qquad (1)$$

where y is the ground truth label of the pixel and p is the output of our model.

2) Focal loss: focal loss is a variant of cross entropy loss that puts more weight on hard and misclassified examples [11]. In binary classification case, it is defined as:

$$\mathrm{FL} = -(\mathrm{y}\log(\mathrm{p})(1-\mathrm{p})\gamma + (1-\mathrm{y})\log(1-\mathrm{p})\mathrm{p}\gamma), \qquad (2)$$

where γ is the focusing parameter that controls the weight assigned to different examples. It penalizes more on the examples that gives high probability on the misclassified examples. This gives the ability of our model to refine the edge prediction since edge pixels are harder to segment.

3) Soft dice loss: Since the competition ranking is based on dice coefficient, it is reasonable to use dice coefficient to train our models. Dice coefficient is given by:

$$\mathrm{Dice} = 2 \times \frac{|\mathrm{X} \cap \mathrm{Y}|}{|\mathrm{X}| + |\mathrm{Y}|}, \qquad (3)$$

where X is the prediction and Y is the ground truth label of each pixel. We modify the dice coefficient so that it can be used by gradient descent for training. The modified version is called soft dice loss and is given by:

$$\mathrm{SoftDice} = \frac{2 \times \sum_{h=1}^{H} \sum_{w=1}^{W} P_{hw} Y_{hw}}{\sum_{h=1}^{H} \sum_{w=1}^{W} P_{hw} + \sum_{h=1}^{H} \sum_{w=1}^{W} Y_{hw}}, \qquad (4)$$

where P is the prediction map of our model. Each pixel value in the prediction map is the probability of it is a car. The final loss is given by:

$$\mathrm{Loss} = \mathrm{BCE} + \mathrm{FL} + \mathrm{SoftDice} + \gamma \left|\left|\mathrm{W}\right|\right|^{2}, \qquad (5)$$

where W is the weight parameters and γ controls the regularization strength.

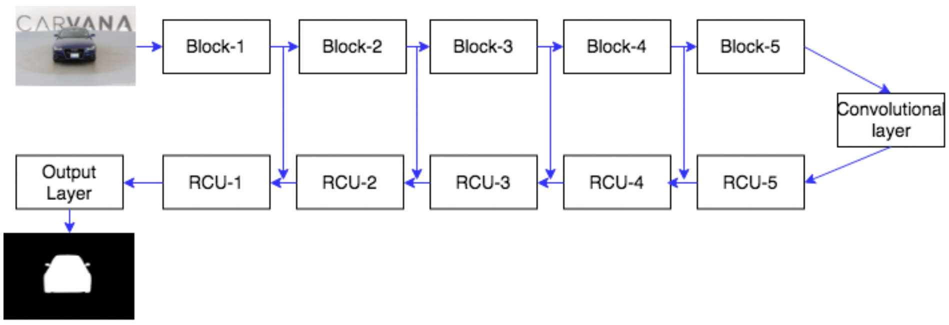


Fig. 5: The overall model architecture.

4 Experiements

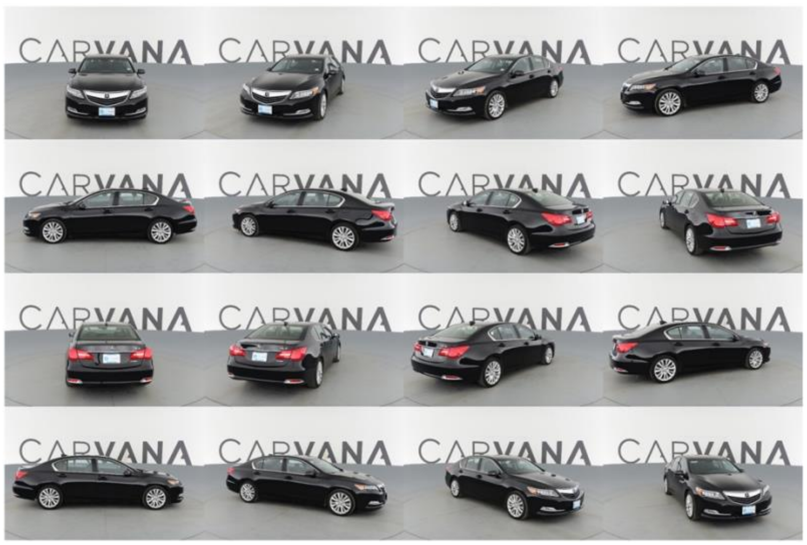
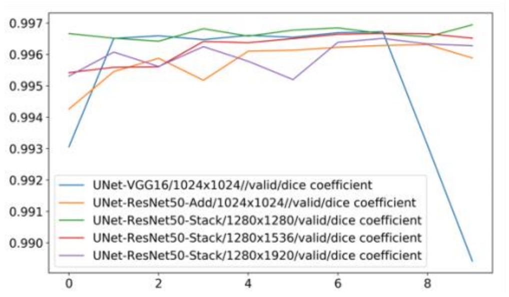
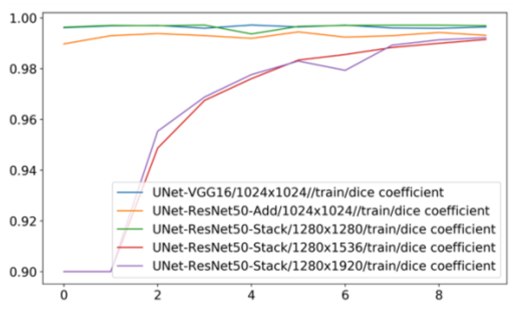


Fig. 6 Images of one car are captured from different viewpoints.

The training dataset consists of 5088 car images. Each car has 16 images corresponding to 16 different captured angles as shown in Fig. 6. Each image has resolution of 1280×1918. Because all the car images are captured at the same angles, training with these images will make the models overfit to the dataset quickly. However, this is not a concern in this competition because the testing dataset also consists of the images captured using the same method but with different cars. We want our models to overfit on the position not colors or shapes of the cars, so we only use two types of augmentation methods: cropping and color adjusting. Cropping makes smaller cars look bigger and color adjusting makes cars to have different colors.



(a) Dice coefficient for validation dataset.



(b) Dice coefficient for training dataset.

Fig. 7 Dice coefficient of five models.

We randomly split the dataset into 4788 images used for training and 300 images used for validation. Instead of training all models on 1280×1918 resolution, we train different models on different resolutions as shown in Table I. We trained five models by using different resolution and different encoder networks. We only use batch size of 8 for training the model with input resolution larger than 1280 × 1280 because of memory constraint. We trained all models with learning rate 0.01 in 10 epochs. Regularization strength γ is set to 0.0001 to all models. The dice coefficient is shown in Fig. 7.

Table 1. Five configurations of our models

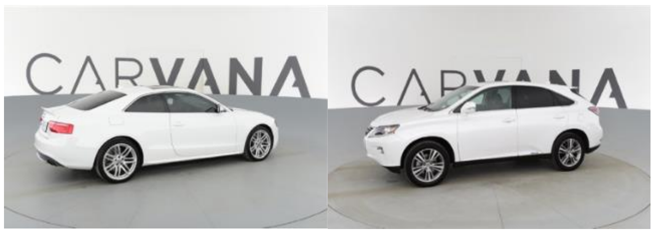
Encoder	Combining method	Training resolution	Batch size
VGG-16	Concat	1024 x 1024	16
ResNet-50	Concat	1280 x 1280	16
ResNet-50	Concat	1280 x 1536	8
ResNet-50	Concat	1280 x 1920	8
ResNet-50	Add	1024 x 1024	16

Based on Equation 3, dice coefficient represents how similar the models' segmentation results with the ground truth results, thus the higher the coefficient, the better the model predictions are. Although all the models perform similarly during the training process, validation results vary a lot. Encoder with VGG-16 overfits the training data after the seventh epoch. We hypothesize this is because VGG-16 has more parameters than ResNet50 and trained with small number of data makes the network overfitting. ResNet-50 with concat operation trained on resolution of 1280 × 1280 gives the best result over all the models. This is because 1280 × 1280 preserves a finer detail than 1024 × 1024 resolution and it is trained with batch size of 16 which gives a better estimate of running variance and running mean than higher resolution models trained with batch size of 8.

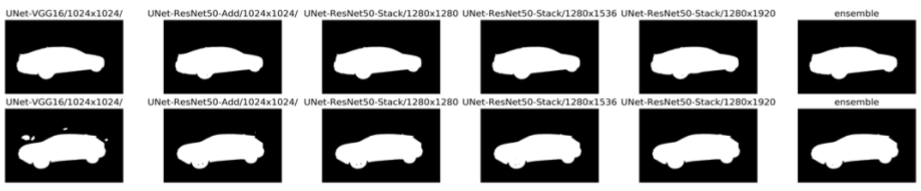
In addition to analyzing single models, we also use majority vote to ensemble the segmentation predictions of each model. The qualitative results are shown in Fig 8. Compared to VGG-16 encoder, ResNet-50 encoder is able to predict more accurate boundaries and details such as antenna. In addition to these findings, we can also see that add operation has the similar results compared to concat operation though it requires less parameters in each decoder upsampling step. The final ensemble model gives smoother boundaries.

5 Conclusion

In this work, we form the background subtraction problem as a semantic segmentation problem. We found that ResNet-50 has better performance than VGG-16 in image segmentation problem. In addition, add operation achieves comparable result with concat operation but with fewer computation. We use majority voting method to ensemble 5 single models which places our team in the top 4% (29 out of 735) of the Kaggle private leaderboard.



(a) Input images.


UNet-VGG16/1024x1024/
UNet-ResNet50-Add/1024x1024/
UNet-ResNet50-Stack/1280x1280
UNet-ResNet50-Stack/1280x1536
UNet-ResNet50-Stack/1280x1920
ensemble
UNet-VGG16/1024x1024/
UNet-ResNet50-Add/1024x1024/
UNet-ResNet50-Stack/1280x1280
UNet-ResNet50-Stack/1280x1536
UNet-ResNet50-Stack/1280x1920
ensemble

(b) Output segmentation maps of five models and majority voting ensemble.

Fig. 8 The results of five single models and their ensemble.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep con- volutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence, 39(12):2481– 2495, 2017.

[2] 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. arXiv preprint arXiv:1606.00915, 2016.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 248–255. IEEE, 2009.

[4] 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, pages 770–778, 2016.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In European Conference on Computer Vision, pages 630–645. Springer, 2016.

[6] X.He,R.S.Zemel,andM.A ́.Carreira-Perpin ̃a ́n.Multiscaleconditional random fields for image labeling. In Computer vision and pattern recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE computer society conference on, volume 2, pages II–II. IEEE, 2004.

[7] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning, pages 448–456, 2015.

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

[9] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. nature, 521(7553):436, 2015.

[10] 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.

[11] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dolla ́r. Focal loss for dense object detection. arXiv preprint arXiv:1708.02002, 2017.

[12] 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, pages 3431–3440, 2015.

[13] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltz- mann machines. In Proceedings of the 27th international conference on machine learning (ICML-10), pages 807–814, 2010.

[14] 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, pages 234–241. Springer, 2015.

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

[16] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806, 2014.

[17] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In AAAI, volume 4, page 12, 2017.

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015.
[19] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In Advances in neural information processing systems, pages 3320–3328, 2014.
[20] S. Zhou, J.-N. Wu, Y. Wu, and X. Zhou. Exploiting local structures with the kronecker layer in convolutional networks. arXiv preprint arXiv:1512.09194, 2015.