EnDeepLabv3+: The 1st Place's Solution to Kaggle Bird Segmentation

Abstract - Deep learning models with the spatial pyramid pooling-based module and the encoder-decoder architecture are widely used in image segmentation. Recent work which has integrated the benefits of the above strategies is called DeepLabv3+. It enhances the performance by adding a simple yet effective shortcut connection in the decoder to refine the segmentation results. It also adds depth-wise separable convolution to both the Atrous Spatial Pyramid Pooling and the decoder modules, resulting in a more efficient and robust encoder-decoder network. We adopt this approach to do the bird segmentation task, and we use the standard measurements to evaluate the accuracy. In order to obtain a more comprehensive analysis, we conducted performance comparisons under different training model settings and various testing modifications. In the experiments, we considered one thousand images as training datasets and three hundred images as testing datasets. Using the IoU (intersection over union) as an evaluation metric, the numerical results would demonstrate the superiority of our EnDeepLabv3+ model (enhanced DeepLabv3+). The codes are available at `https://github.com/RYY0722/ee4211-seg-group10`.

1. Introduction

Image segmentation is an important component of many visual comprehension systems, which is a technique to segment pictures into several parts. It can be interpreted as a classification task for each pixel with semantic labels. Generally, the segmentation process would use a collection of object categories to label all image pixels at pixel level. In reality, segmentation is used in a wide variety of applications, involving medical image analysis, autonomous cars, and surveillance. In the machine learning field, numerous image segmentation algorithms have been developed and proposed, such as thresholding [1], region-growing [2], and K-mean clustering [3]. However, those algorithms have limitations and are case-dependent, achieving a relatively high accuracy in comparison to the most recent algorithms is difficult.

During the last decade, deep learning (DL) models have resulted in a new generation of image segmentation models with incredible performance breakthroughs. They reach the highest accuracy rates on many famous benchmarks. The development of DL models is leading to a paradigm change in the image segmentation field. DeepLabv1 [4] and DeepLabv2 [5] are two of the most widely used algorithms for picture segmentation. They have some outstanding characteristics. They use the dilated convolution to compensate for the network's declining resolution. It addresses the problems of max pooling and striding. More importantly, they employ Atruos Spatial Pyramid Pooling (ASPP), which probes an incoming convolutional feature layer with filters at different sampling rates, collecting objects and image context at many scales in order to robustly segment objects at multiple scales. Also, they enhance object boundary localization via merging the approaches of probabilistic graphical models and deep convolutional neural networks (CNNs).

More recently, an updated version of DeepLab model is introduced. DeepLabv3 [6] is a combination of cascaded and parallel modules for dilated convolutions. In the ASPP, the parallel convolution modules are grouped together, and then a batch normalization and a 1×1 convolution are added. Concatenating all the outputs and processing them with another 1×1 convolution produces the final results with logits for each pixel. In recent years, DeepLabv3+ [7] has been released. It employs an encoder-decoder architecture and atrous separable convolution, which is composed of depth-wise convolution (spatial convolution for each channel of the input) and pointwise convolution (1×1 convolution with the depth-wise convolution as input). DeepLabv3+ is encoded using the DeepLabv3 framework. The most relevant model is based on a modified Xception backbone with additional layers, dilated depth-wise separable convolutions rather than max pooling, and batch normalization. The top DeepLabv3+ pre-trained on the COCO and JFT datasets achieved an mIoU score of 89.0 percent in the 2012 PASCAL VOC challenge.

[1] 55701944, Year 4, CDE
[2] 56204372, Year 3, INFE
[3] 56046680, Year 4, CDE
[4] 55670096, Year 4, CDE

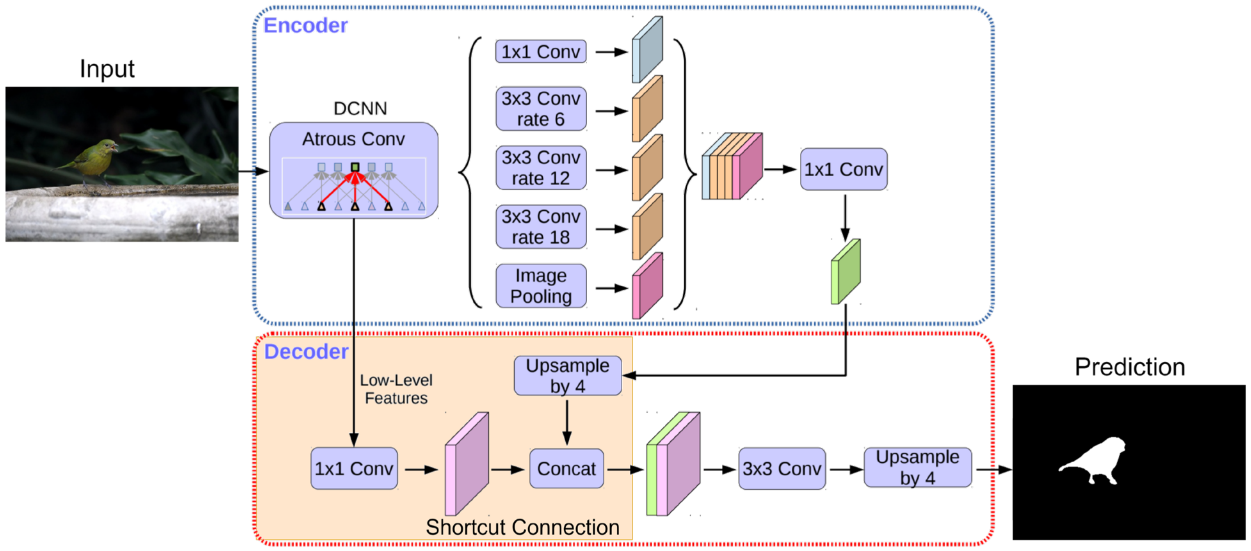

Figure 1. Overview of the EnDeepLabv3+. The orange box highlights the shorcut connection.

This project applies the DeepLabv3+ model for the bird segmentation task. We would optimize the model to obtain more accurate segmentation predictions. Experiments are conducted on the provided bird-only image datasets. The experimental results demonstrate that our proposed improvements are superior to the default model. The rest of this paper is organized as follows. The methodology is described in Section 2. In Section 3, the numerical results of our enhanced version DeepLabv3+ (EnDeepLabv3+) are reported. Ultimately, the conclusions are drawn in Section 4.

2. Methodology

Our method named EnDeepLabv3+ is an enhanced version of DeepLabv3+. Therefore, in this section, we would provide an overview of DeepLabv3+ first, and then discuss the improvements that we adjusted.

2.1. Overview of DeepLabv3+

The overall architecture of the DeepLabv3+ model is shown in Figure 1. It is obvious that DeepLabv3+ is composed mainly of an encoder and a decoder. Using the encoder part, we could extract the features from the incoming image. On the decoder part, we could decode the features to obtain the predicted result. This prediction refers to the categories for each pixel in the original image.

The main body of its encoder is composed of the deep convolutional neural network (DCNN) as the backbone and an atrous spatial pyramid pooling module (ASSP). The backbone is usually a pretrained classification network such as ResNet [8], VGG [9], and Xception [10]. And the ASSP consists of several atrous convolutional layers with different dilate rates to extract multiscale information. Atrous convolution is the key characteristic of the DeepLab models, which can provide the receptive field of different sizes without changing the size of the feature map. In this way, the models are able to capture multiscale information.

The concrete workflow in the model is illustrated as follows. Firstly, we would pass the image to DCNN for feature extraction. The backbone extracts features and then the extracted features are processed by a set of atrous convolution layers of different dialed rates. After that, these features are stacked and merged by a 1×1 convolutional layer. At that moment, the encoder part would gradually reduce the feature map, capture higher semantic information, and result in the shallow feature layer and the deep feature layer. On the decoder part, the first component in the decoder is a shortcut connection of the shallow and deep features, which will be discussed in detail in section 2.2. The decoder part would perform an upsampling process to the deep feature layer, then it would concatenate with the shallow feature layer. After merging these two convolutional layers, the features will be decoded by a 3×3 convolutional layer, and then it would be followed by an upsampling operation. Finally, the model adjusts the output prediction result to be the same size as the input image, and the predicted result can represent the categories of each pixel in the input image.

2.2. Shortcut Connection

One important feature of DeepLabv3 is the shortcut connection, which is highlighted in orange in Figure 1. Shortcut connection is added to preserve low level features in the decoding process. The rationale behind is that segmentation is highly related to segmentation task is highly correlated with shallow features [11,12,13]. Suppose the shallow features from the backbone is denoted as F_s, and the output feature by the encoder is denoted as F_e. The concrete process is as follows:

1. F_s is processed by a 1×1 convolutional layer, which gives F_s'.
2. $F_e \uparrow$ is produced by upsample F_e by a factor of 4.
3. F_s is concatenated with $F_e \uparrow$, where the output is denoted as F_d.

After that, F_d is further processed by the decoder to produce the final prediction. Since the F_s is the output of Conv2 layer in ResNet [8], the channel of it is 256. For another thing, the output feature of the decoder is also 256. Therefore, in order to prevent the high-level features obtained by the encoder from being weakened, 1x1 convolution is used to reduce the dimension of the low-level features (Step 1). Thus, for this 1×1 convolutional layer, the input channel number is fixed at 256 while the output channel number, which is also called shortcut dimension, can be manually selected.

Intuitively, for large shortcut dimension, the information from the shallow features is better preserved, while for small ones, the computational cost is lower. Since improving the model performance is the main objective of this project, we increase the shortcut dimension despite the resultant higher computational cost. Detailed selection process and the corresponding performance will be reported at section 4.4.

2.3. Input Ensemble

We also improve the model performance via input ensemble. Such technique is a kind of post training optimization, which does not change the model architecture but smooth the predictions by providing input image at different viewpoints and scales. Concretely, input ensemble refers to processing the input into multiple variants and making the predictions by combining the results of these image. The goal of using this technique is to provide different versions of the input image to the model, which consists of different viewpoints and scale of the input. In this way, some falsely predicted pixels can be corrected by other ensembled inputs.

2.3.1 Flipping

The first method to ensemble the input is flipping, which can be illustrated with Figure 2. To be specific, the input image I is flipped horizontally, which gives I'. Secondly, the model will make predictions on both images, which are denoted as $\mathcal{G}(I)$ and $\mathcal{G}(I')$, respectively. The final output is the point-wise average of these two segmentation maps.

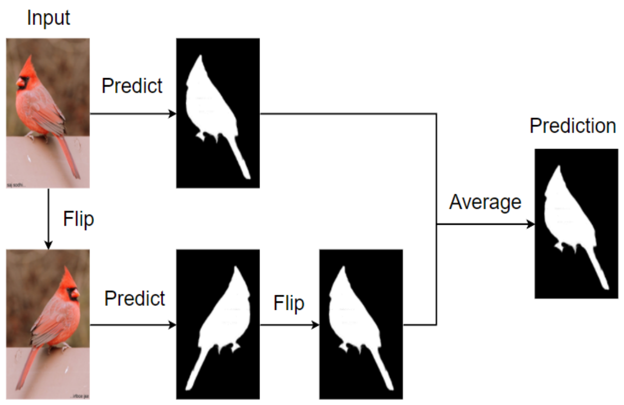

Figure 2. Illustration of flipping.

2.3.2 Multiscale Image Pyramids

We also add the multiscale image pyramids of the input image to the ensemble. Image pyramids are a collection of images of different resolution by up-sample or down-sample the original input image. This technique is particularly useful since it introduces scale invariance by providing a range of possible resolution of the original image to the network. As a result, this input ensemble becomes able to detect birds of various sizes.

The whole process can be illustrated with Figure 3. Firstly, the original image is resized to multiple scales, where the model will make predictions for each entry. Then, the prediction maps are resized to the original scale using bicubic-interpolation, and the average of them gives the final output.

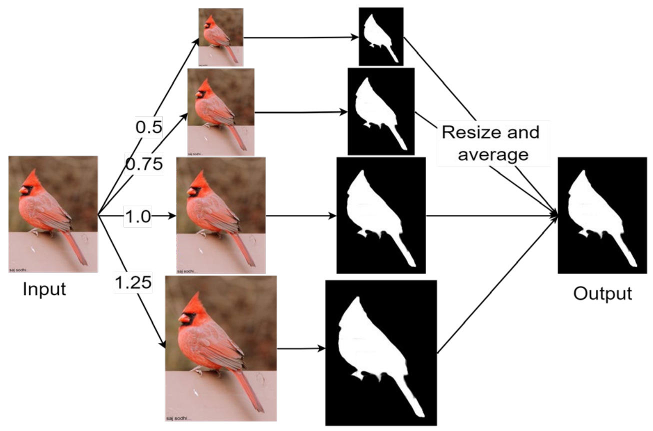

Figure 3. Illustration of multi-scale ensemble.

3. Experiments and Results

3.1. Datasets and Performance Measurements

In the experiments, a total of one thousand images for training and three hundred images serve as the held-out test set, where the ground truth labels are not available.

In the evaluation part, we majorly focus on the Jaccard Similarity Coefficient, which is well known as the IoU. Also, other commonly used measurements would be considered as our reference metrics or supported measurements, including Accuracy, Precision, Hausdorff Distance, and Dice Coefficient. The IoU is the area of intersection between the predicted segmentation and the ground truth, it is defined as

$$IoU(A,\ B)\ = \frac{\left|\left|A\cap B\right|\right|}{\left|\left|A\cup B\right|\right|}$$

where A and B are two segmentation masks for a given class, $\left|\left|A\cap B\right|\right|$ and $\left|\left|A\cup B\right|\right|$ are the cardinal sets (for images, the area in pixels). The IoU indices are bounded between 0 (where there is no overlap) and 1 (where there is perfectly overlap). In terms of the confusion matrix, the IoU index can also be expressed in terms of true positives (TP), false positives (FP), and false negatives (FN) as

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

In this project, we have tried different methods to increase the IoU as TP increases. The input images are augmented with the following four strategies:

1. Flipping with a probability of 0.5.
2. Rotation with a random angle $\in [-10°, 10°]$
3. Rescaling with a factor $\in \left[\frac{8}{9},\frac{9}{8}\right]$
4. Random color distortion in HSV space with a value $\in [-10,10]$.

The inputs of the model at training stage are obtained are randomly cropped patches of size 384×384 and resized to 256×256.

3.2. Backbone selection

We tried out different backbones for the segmentation model. Figure 4 shows the performance of different backbones when training the model for 200 epochs. The result showed that Resnet 152 is the best among these three backbones with score reaching 0.79712. Therefore, we use Resnet 152 as the backbone in the final model implementation.

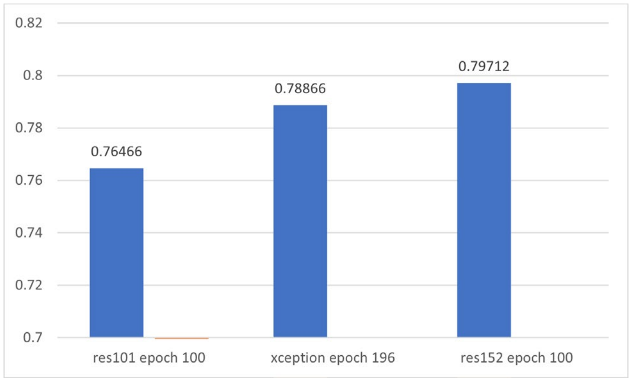

Figure 4. Performance of DeepLabv3+ with different backbones when trained with 200 epochs.

3.3. Shortcut dimension selection

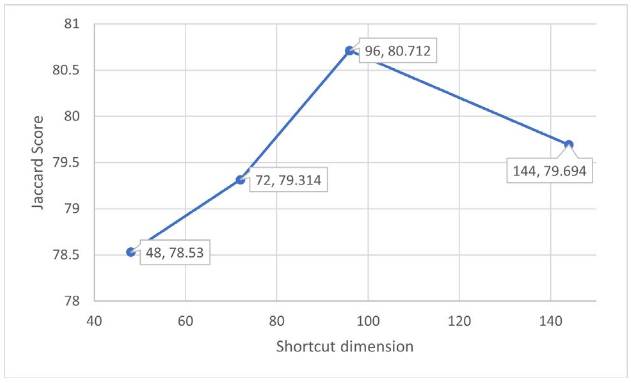

Figure 5. Performance change of DeepLabv3+ with shortcut dimension, where ResNet 152 is the backbone.

We also explore the model performance change when the shortcut dimension changes in order to find out the optimal settings. As shown in Figure 5, we found that the performance is improved when the shortcut dimension is increased at the beginning. However, after reaching the peak value at shortcut dimension of 96, the performance starts to drop. Therefore, the shortcut dimension is set to 96 in this project.

3.4. Epoch Selection

During testing, we trained the model for 280 epochs and saved for every 20 epochs. Figure 6 shows how the model performance change for different number of epochs. We found that continuing to train the optimal model on validation set can still improve the model performance. However, the model tends to overfit after epoch 250. Therefore, we train the model for 250 epochs in the final implementation.

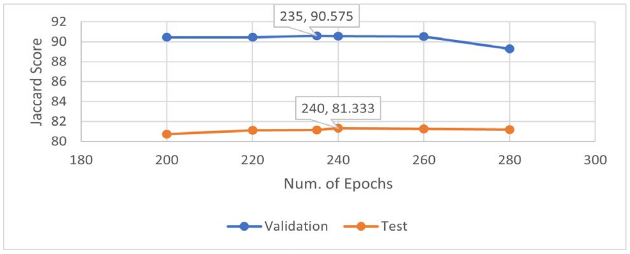

Figure 6. Model performance change on the validation and test set as the number of epochs increases.

3.5. Ablation study of Input Ensemble

Table 1 shows the model performance with or without flipping at inference time respectively. From the above, we can see that such flipping technique is pretty powerful when working with different backbones. For both backbones, the score has improved by around 1.5 % in performance after adding the flipping ensemble, which demonstrates the great effectiveness of this trick.

Table 1. Performance of DeepLabv3+ of different backbones when testing with /without testing.

Backbone	without flipping	with flipping
Res101	76.466	**78.166**
Res152	81.333	**83.037**

We also tested out different scale options of the image pyramid in the input ensemble. To find out the most suitable scales for this task, we tried different cases including the single scale [1.0], and different scaling within [0.5] and [3.0]. As shown in Figure 8, it can be observed that case 4 scaling from [0.5] to [2.5] achieves the best performance which achieved the best score around 0.85166; thus, the scaling in case 4 is used in this project. The details of different settings are illustrated as follows.

Case 1: [1.0]
Case 2: [0.5, 0.75, … , 1.75]
Case 3: [0.5, 0.75, … , 2.0]
Case 4: [0.5, 0.75, … , 2.5]
Case 5: [0.5, 0.75, … , 2.75]
Case 6: [0.5, 0.75, … , 3.0]

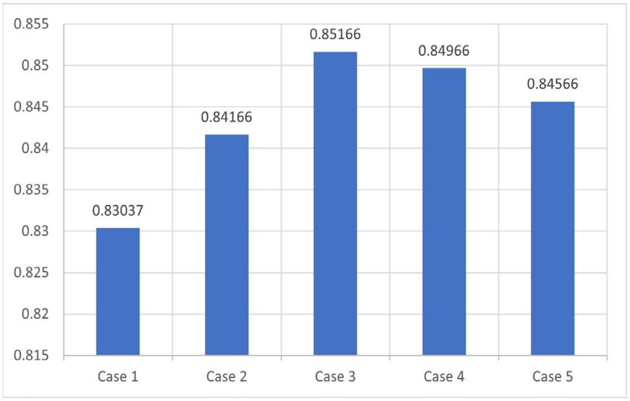

Figure 7. Performance change of DeepLabv3+ with backbone of Res 152 and trained for 200 epochs when using different scales of image pyramid.

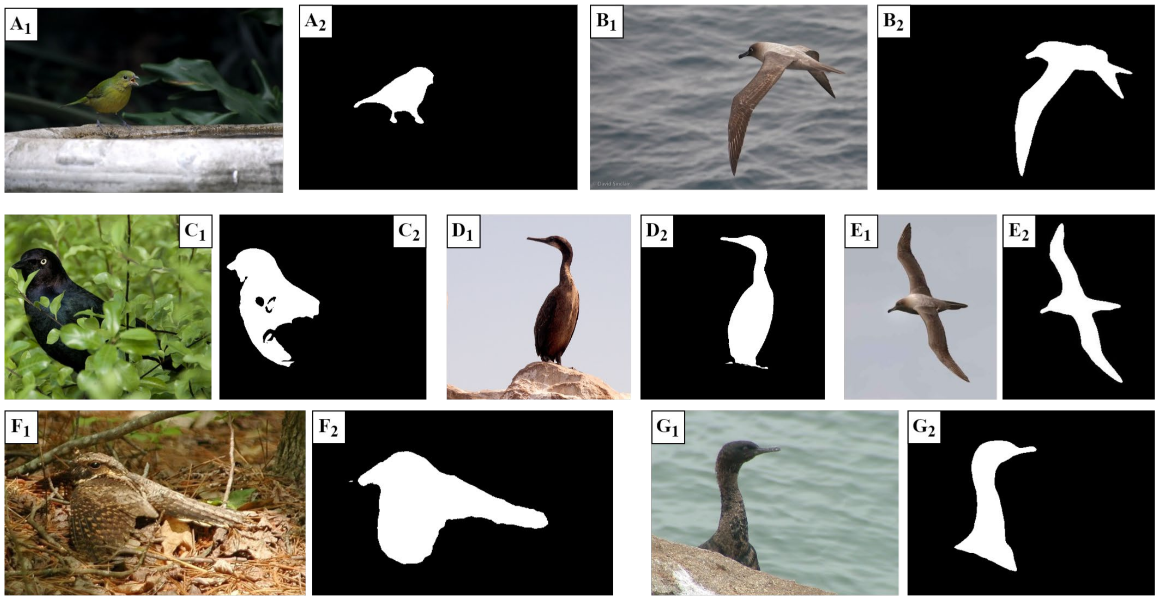

Figure 8. Visual results. The colorful images are the inputs, and the binary images are the segmentation results.

3.6. Overall Performance

By integrating the improvements, our EnDeepLabv3 model finally achieved an IoU of 0.852 on the held-out test set, ranking *first* on the leaderboard.

Some representative samples are shown in Figure 8. It can be observed that our model can handle images of various characteristics and has produced pretty accurate results as compared to human observation. The model can still segment the bird correctly no matter it takes up a large or small portion of the whole picture. Moreover, despite that the shape of the flying birds (B, D) are quite different from the standing ones (A, D), the model can handle both situations smoothly. Notably, even the bird is occluded by a large portion (C, G) or it share very similar color with the background (F), the model can still predict quite accurate results.

3.7. Implementation Details

All the experiments are conducted on a workstation with Windows 10 64-bit OS and a Nvidia GTX 1080Ti GPU. All codes are written in Python 3.9.12 and pytorch, with the following dependencie: numpy, opencv, pandas, pillow, pythorch, scipy, scikit-image, torchvision with GPU acceleration enabled. An SGD optimizer with momentum of 0.9 is used to train the model. The initial learning rate is set to 0.001 and a exponential decay scheduler with . The batch size is set to 8.

4. Conclusion

In conclusion, our contribution for improving the DeepLabv3 on the bird segmentation task can be divided into two parts, one is at training stage and the other is at inference time. During training, we add the data augmentation, change the backbone, carefully select the shortcut dimension and training epochs. During inference, we ensemble the input images by applying flipping and multi-scale analysis, which are shown to be effective to boost the model performance. By these improvements, we improve the model score from around 0.76399 to 0.85166, ranking first on the leaderboard. In the future development, we would focus on finer segmentation predictions by introducing boundary-aware loss functions and developing more dedicated metrics.

References

[1] N. Otsu, “A threshold selection method from gray-level histograms,” *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[2] R. Nock and F. Nielsen, “Statistical region merging,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 26, no. 11, pp. 1452–1458, 2004.

[3] N. Dhanachandra, K. Manglem, and Y. J. Chanu, “Image segmentation using k-means clustering algorithm and subtractive clustering algorithm,” *Procedia Computer Science*, vol. 54, pp. 764–771, 2015.

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Semantic image segmentation with deep convolutional nets and fully connected crfs,” *arXiv preprint arXiv:1412.7062*, 2014.

[5] 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,” IEEE transactions on pattern analysis and machine intelligence, vol. 40, no. 4, pp. 834–848, 2017.

[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] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in Proceedings of the European conference on computer vision (ECCV), 2018, pp. 801–818.

[8] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[9] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[10] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258).

[11] Bilinski, P., & Prisacariu, V. (2018). Dense decoder shortcut connections for single-pass semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 6596-6605).

[12] Geirhos, R., Jacobsen, J. H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., & Wichmann, F. A. (2020). Shortcut learning in deep neural networks. Nature Machine Intelligence, 2(11), 665-673.

[13] Liu, T., Chen, M., Zhou, M., Du, S. S., Zhou, E., & Zhao, T. (2019). Towards understanding the importance of shortcut connections in residual networks. Advances in neural information processing systems, 32.