LinkNetB7: LinkNet with Pretrained Encoder for Efficient Road Segmentation
CS-433 Machine Learning – Class Project 2

Aoyu Gong, Yuxuan Zong, Sijia Du
School of Computer and Communication Sciences, EPFL, Switzerland

***Abstract*—In this project, we focus on pixel-wise road segmentation from satellite images. To increase the accuracy of road segmentation, we design a deep neural network based on LinkNet architecture, named LinkNetB7. The network utilizes the pretrained EfficientNet-B7 as the encoder, and adopts the decoder similar to LinkNet. By using this encoder, the network takes advantage of the compound scaling method. Results with thorough comparisons are provided to validate our design. In the EPFL Road Segmentation Challenge, our network achieves an F1 score of 0.923, with a categorical accuracy of 0.958 on AIcrowd (submission-ID: 168621).**

I. Introduction

Road segmentation from satellite images has been becoming a research focus in recent years. It has been widely used in various fields, such as vehicle navigation, urban planning, and geographical information system [1]. Different methods have been proposed to tackle the problem of designing convolutional neural networks (CNNs) for efficient pixel-wise road segmentation. One of the most successful visual models is fully convolutional network (FCN) [2]. The main idea of this model is to use CNN as a powerful feature extractor by replacing fully connected layers with convolutional layers. FCN has been further improved, and more efficient models for pixel-wise road segmentation have been proposed, such as U-Net [3], LinkNet [4], and D-LinkNet [5]. Specifically, U-Net utilizes skip connections to concatenate layers in the encoder with layers in the decoder. In LinkNet, the decoder is shared with knowledge learnt by the encoder at all layers leading to an overall more efficient network.

In this project, we focus on LinkNet architecture and use the pre-trained EfficientNet-B7 [6] as the encoder to design a LinkNet-type network, named LinkNetB7. The remainder of this report is organized as follows. The dataset on AIcrowd is given in Section II. The network architecture is presented in Section III. The implementation details are specified in Section IV. Results with thorough comparisons are provided in Section V. Conclusions are given in Section VI.

II. Dataset

The training dataset for the road segmentation challenge contains 100 satellite images in RGB format. The resolution of every training image is 400×400 pixels. Moreover, each training image contains a corresponding mask in grayscale format, where white pixels stand for roads and black pixels

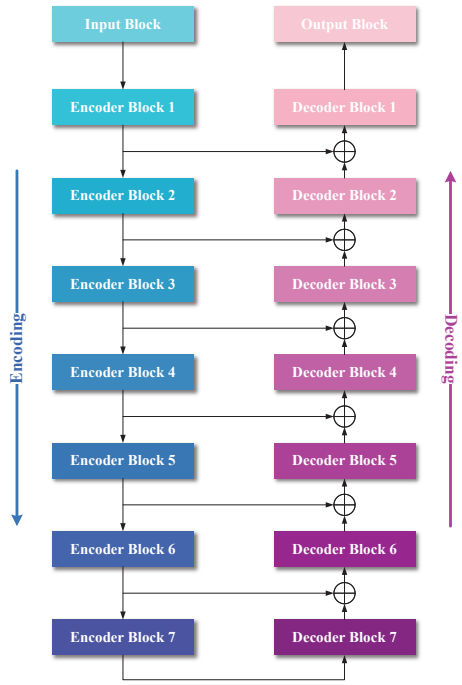


Figure 1: The architecture of proposed LinkNetB7. The left part is the encoder, while the right part is the decoder.

stand for background. It is worth mentioning that the pixels of these masks may not take values in $\{0, 255\}$. Therefore, a threshold for binarization is further adopted. To measure the performance, the testing dataset that contains 50 satellite images in RGB format is provided. The resolution of every testing image is 608×608 pixels. Both datasets are acquired from Google Maps. To create a submission, every predicted mask is divided into 1444 patches with the size of 16×16 pixels. The label of each patch is further assigned based on the average value of 256 pixels and a foreground threshold. The submissions should be submitted to AIcrowd [7].

III. Network Architecture

The architecture of proposed LinkNetB7 is presented in Figure 1. The left part of this network is the encoder while the right part is the decoder. This network utilizes the pre-trained EfficientNet-B7 [6] as the encoder. As presented in Figure 1, the encoder starts with an input block. This block

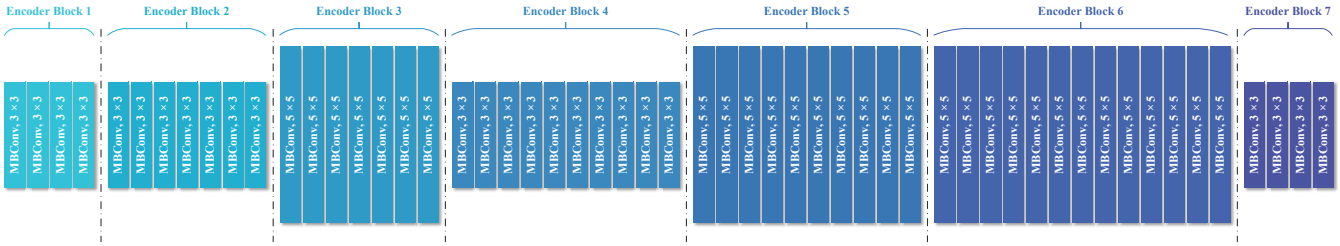


Figure 2: The architecture of encoder blocks. The basic building block of each encoder block is MBConv (mobile inverted bottleneck convolution) block. The height of each MBConv block is proportional to its kernel size.

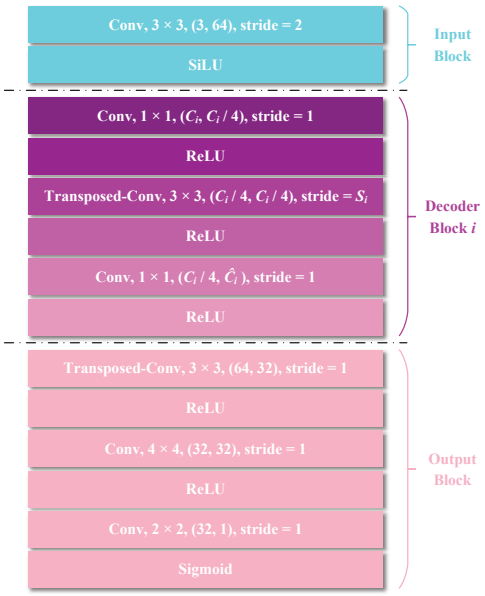


Figure 3: The architecture of the input block, decoder block $i \in \{1, 2, \ldots, 7\}$, and output block, where $S_i = 2$ for each $i \in \{1, 2, 3, 4, 6\}$ and $S_i = 1$ for each $i \in \{5, 7\}$.

applies a convolution over an input image with a kernel of size 3×3 and a stride of 2, and SiLU activation function [8], [9]. The architecture of encoder blocks is presented in Figure 2. The basic building block of every encoder block is MBConv (mobile inverted bottleneck convolution) block [10], with squeeze-and-excitation optimization. Similarly, the architecture of decoder blocks is shown in Figure 3, where C_i and $\hat{C}_i$ denote the number of input and output channels of decoder block i respectively for each $i \in \{1, 2, \ldots, 7\}$. The transposed convolution is leveraged in the decoder [11]. Specifically, every `Conv`, $K \times K$, $(C_{\text{in}}, C_{\text{out}})$, stride $= S$ or `Transposed-Conv`, $K \times K$, $(C_{\text{in}}, C_{\text{out}})$, stride $= S$ operation has three parameters, where $K \times K$ denotes the size of the kernel, $(C_{\text{in}}, C_{\text{out}})$ denotes the number of input and output channels, and S is the stride of the convolution. For the input block and decoder blocks, batch normalization is performed after each convolutional layer. In this network, we feed the output of each encoder block as the input of its corresponding decoder block by skip connections. The last block of this network is an output block. Its architecture is also presented in Figure 3. Table I illustrates the input and output feature maps of each block in this network.

Table I: The input and output feature maps.

Blocks	Input Resolution	Output Resolution	#Input Channels	#Output Channels
Input	384×384	192×192	3	64
Encoder 1	192×192	192×192	64	32
Encoder 2	192×192	96×96	32	48
Encoder 3	96×96	48×48	48	80
Encoder 4	48×48	24×24	80	160
Encoder 5	24×24	24×24	160	224
Encoder 6	24×24	12×12	224	384
Encoder 7	12×12	12×12	384	640
Decoder 7	12×12	12×12	640	384
Decoder 6	12×12	24×24	384	224
Decoder 5	24×24	24×24	224	160
Decoder 4	24×24	48×48	160	80
Decoder 3	48×48	96×96	80	48
Decoder 2	96×96	192×192	48	32
Decoder 1	192×192	384×384	32	64
Output	384×384	384×384	64	1

Similar to LinkNet and D-LinkNet, the proposed network makes use of the output of each encoder block, and feeds it into the corresponding decoder block. By doing this, we aim at recovering spatial information lost in the encoding which can be used by the decoder. Moreover, since the decoder is shared with the knowledge learned by the encoder, it uses fewer parameters. Compared with LinkNet and D-LinkNet, the proposed network uses the pre-trained EfficientNet-B7 as the encoder instead of using the pre-trained ResNet [12]. By doing this, we aim at taking advantage of the compound scaling method that uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients [6].

IV. Implementation

In this section, we introduce the training data processing, loss function, testing data processing.

A. Training Data Processing

To adapt to the architecture of the proposed network, we first resize training images and their corresponding masks, denoted by training pairs. Let $H \times W$ denote the resolution of training pairs. Next, let $H' \times W'$ denote the resolution of resized training pairs, satisfying $H' = 32q$ and $W' = 32k$ where $q, k \in \mathbb{Z}^+$. Here, we set H' and W' to 384. For each resized training pair, we implement the following steps.

1) **Rotation:** We rotate each resized training pair by $\theta \in \{0°, 45°, 90°, 135°\}$ (counter-clockwise).
2) **Flipping:** For each rotated training pair, we keep the original pair, and further implement the horizontal flip and vertical flip.
3) **Shifting:** We shift each flipped training pair randomly. Let S_0 and S_1 represent the shift along two axes. Both random variables satisfy $S_0, S_1 \sim U(-16, 16)$.

In all three steps, the pixels outside the boundaries of the processed training pairs are filled by `reflect`. Given that 100 training pairs are used for training, we will have 1200 processed training pairs after the training data processing.

B. Loss Function

We define the loss function starting from the binary cross entropy (BCE) loss:

$$L_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(p_i) + (1 - y_i)\log(1 - p_i),$$

where $y_i \in \{0 \text{ (background)}, 1 \text{ (road)}\}$ specifies the label of pixel i, p_i is the predicted probability of pixel i being roads, and N is the number of pixels. Inspired by the Focal Loss [13], we further modify the BCE loss by adding two tunable parameters $\alpha \in [0, 1]$ and $\gamma \in \mathbb{Z}^+$ as following:

$$L'_{\text{BCE}} = \alpha\big(1 - \exp(-L_{\text{BCE}})\big)^{\gamma} L_{\text{BCE}}.$$

Besides the modified BCE loss, we also use the intersection over union (IoU) as the evaluation metric. This metric can be seen as a similarity measure among a finite number of sets [14]. For two sets A and B, it can be defined by

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

Based on this expression, the IoU loss can be defined by

$$L_{\text{IoU}} = 1 - \sum_{i=1}^{N} \frac{y_i p_i}{y_i + p_i - y_i p_i}.$$

Combining the modified BCE loss and the IoU loss, the BCE-IoU loss used in our problem can be computed by

$$L_{\text{BCE-IoU}} = \beta L'_{\text{BCE}} + (1 - \beta) L_{\text{IoU}},$$

where $\beta \in [0, 1]$ is a tunable parameter.

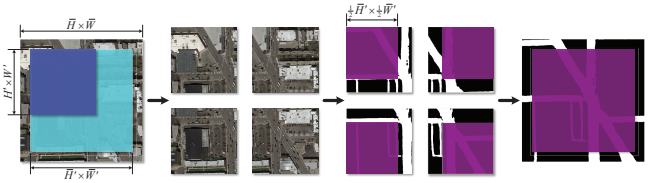


Figure 4: The procedure for the testing data processing.

Table II: Varying α for $L_{\text{BCE-IoU}}$ ($\gamma = 3$ and $\beta = 0.7$)

α	γ	β	F1 Score	Accuracy
0.05	3	0.7	0.919	0.956
0.15	3	0.7	0.921	0.957
0.25	**3**	**0.7**	**0.922**	**0.958**
0.35	3	0.7	0.918	0.955
0.45	3	0.7	0.916	0.954

C. Testing Data Processing

Considering the fact that the resolution of testing images and training images are different, we implement the following testing data processing. First, each testing image is also resized. Let $\bar{H} \times \bar{W}$ and $\bar{H}' \times \bar{W}'$ denote the resolution of testing images and resized ones. Here, we set

$$\bar{H}' = \lfloor \frac{\bar{H} H'}{H} \rfloor, \ \bar{W}' = \lfloor \frac{\bar{W} W'}{W} \rfloor.$$

Each resized testing image is divided into four patches with the size of $H' \times W'$, one at each corner. For each patch, its mask is predicted correspondingly. Next, the four masks are merged into one mask with the size of $\bar{H}' \times \bar{W}'$. This mask is further restored to the original resolution, and is used to create a submission file. The procedure for the testing data processing is presented in Figure 4.

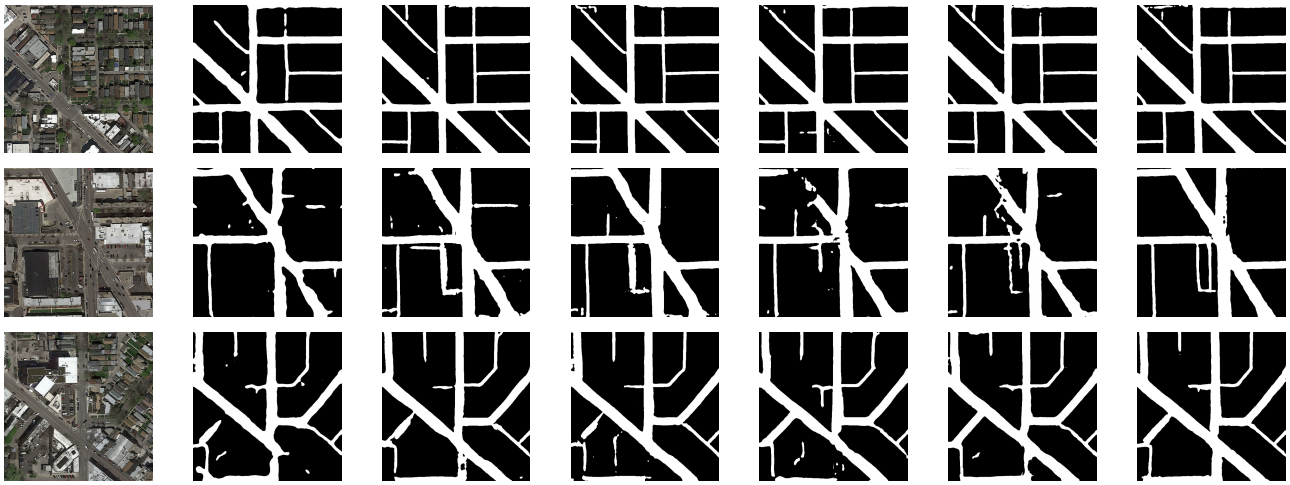
V. Results

This section includes two subsections. The first subsection compares the performance of our network under different parameter settings. The second subsection further compares the performance of our network with other networks using the parameter setting determined in the first subsection. The threshold for binarization is set to 0.3, and the foreground threshold is set to 0.25, as introduced in Section II. We train our network using Adam algorithms, and set the batchsize to 8. Moreover, we set the initial learning rate to 10^{-3}, and decay the learning rate by 0.5 every 10 epochs.

In the following subsections, each result is obtained using NVIDIA Tesla P100, of which the memory is 25.46GB, on Google Colab Pro.

A. Comparisons under Different Parameter Settings

We select three tunable parameters for the BCE-IOU loss by testing the results of road segmentation on AIcrowd, and use the F1-score and Accuracy as measures. Table II shows the results as a function of α for $\gamma = 3$ and $\beta = 0.7$. We



(a) Input Image (b) U-Net (c) LinkNet34 (d) D-LinkNet34 (e) D-LinkNet50 (f) D-LinkNet101 (g) LinkNetB7

Figure 5: Segmentation results of different models.

Table III: Varying γ for $L_{\text{BCE-IoU}}$ ($\alpha = 0.25$ and $\beta = 0.7$)

α	γ	β	F1 Score	Accuracy
0.25	1	0.7	0.920	0.956
0.25	2	0.7	0.917	0.955
0.25	**3**	**0.7**	**0.922**	**0.958**
0.25	4	0.7	0.921	0.957
0.25	5	0.7	0.920	0.957

Table IV: Varying β for $L_{\text{BCE-IoU}}$ ($\alpha = 0.25$ and $\gamma = 3$)

α	γ	β	F1 Score	Accuracy
0.25	3	0.4	0.920	0.954
0.25	3	0.5	0.919	0.956
0.25	**3**	**0.6**	**0.923**	**0.958**
0.25	3	0.7	0.922	0.958
0.25	3	0.8	0.915	0.954

note that the performance slightly decreases as α becomes larger. Similarly, Table III shows the results as a function of γ for $\alpha = 0.25$ and $\beta = 0.7$. Table IV shows the results as a function of β for $\alpha = 0.25$ and $\gamma = 3$. Among these parameter settings, $\alpha = 0.25$, $\gamma = 3$ and $\beta = 0.6$ works best, and this setting will be used in the next subsection.

B. Comparisons with Other Networks

Table V compares the performance of different models by testing the results of road segmentation on AIcrowd, and use the F1-score and Accuracy as measures. To be specific, the standard U-Net [3], LinkNet [4] and D-LinkNet [5] are used for comparisons. We observe that proposed LinkNetB7 slightly outperforms other models: 3.01% improvement over U-Net, 1.32% improvement over LinkNet34, and 0.98% improvement over D-LinkNet34 when comparing the F1 score.

Table V: Segmentation results of different models.

Models	Backbone	F1 Score	Accuracy
U-Net	N/A	0.896	0.943
LinkNet34	ResNet34	0.911	0.951
D-LinkNet34	ResNet34	0.914	0.953
D-LinkNet50	ResNet50	0.907	0.949
D-LinkNet101	ResNet101	0.908	0.950
LinkNetB7	**EfficientNet-B7**	**0.923**	**0.958**

The results indicate that the design of proposed LinkNetB7 is reasonable as the network adopts an architecture similar to LinkNet and D-LinkNet, and further takes advantage of the compound scaling method. Although our network performs best, it is worth mentioning that it takes much longer time to converge than LinkNet34 does since more parameters are used. Figure 5 presents predicted masks of three randomly selected testing images generated by different models. It is shown that our network has two main advantages: edges of predicted masks are more accurate, and the roads obscured by cars or trees can be segmented more accurately.

VI. Conclusion

In this project, we propose a deep neural network based on LinkNet architecture, named LinkNetB7, for pixel-wise road segmentation from satellite images. A novel feature of this work is to utilize the pretrained EfficientNet-B7 as the encoder. Proposed LinkNetB7 is able to achieve an F1 score of 0.923, with a categorical accuracy of 0.958 on AIcrowd. Moreover, our network outperforms other LinkNet-type networks on the given testing dataset. This result may be further improved by implementing more complex data augmentation such as color jittering, or test time augmentation (TTA), and using more efficient pre-trained networks as the encoder.

ACKNOWLEDGEMENTS

The authors thank the TML and MLO groups for their careful guidance.

REFERENCES

[1] X. Yang, X. Li, Y. Ye, R. Y. Lau, X. Zhang, and X. Huang, "Road detection and centerline extraction via deep recurrent convolutional neural network U-Net," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 9, pp. 7209–7220, 2019.

[2] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 3431–3440.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput. Assist. Interv. (MICCAI)*, 2015, pp. 234–241.

[4] A. Chaurasia and E. Culurciello, "Linknet: Exploiting encoder representations for efficient semantic segmentation," in *Proc. IEEE Vis. Commun. Image Process. (VCIP)*, 2017, pp. 1–4.

[5] L. Zhou, C. Zhang, and M. Wu, "D-linknet: Linknet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) Workshops*, 2018, pp. 182–186.

[6] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 6105–6114.

[7] "EPFL Road Segmentation: Challenges." [Online]. Available: https://www.aicrowd.com/challenges/epfl-ml-road-segmentation

[8] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Netw.*, vol. 107, pp. 3–11, 2018.

[9] P. Ramachandran, B. Zoph, and Q. V. Le, "Swish: a self-gated activation function," *arXiv preprint arXiv:1710.05941*, vol. 7, 2017.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 4510–4520.

[11] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2010, pp. 2528–2535.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778.

[13] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 2980–2988.

[14] A. Buslaev, S. Seferbekov, V. Iglovikov, and A. Shvets, "Fully convolutional network for automatic road extraction from satellite imagery," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) Workshops*, 2018, pp. 207–210.