Abstract

A U-Net model proposed by Ronneberger et al. which resembles an autoencoder but with convolutions instead of a fully connected layer was used to solve Carvana's image masking competition in Kaggle. The U-Net was trained only with data set provided and without any pretrained layers because it is a binary segmentation model.

Background

Carvana, a successful online used car startup, has seen opportunity to build long term trust with consumers and streamline the online buying process.

An interesting part of their innovation is a custom rotating photo studio that automatically captures and processes 16 standard images of each vehicle in their inventory. While Carvana takes high quality photos, bright reflections and cars with similar colors as the background cause automation errors, which requires a skilled photo editor to change[1].

Figure 1: Background removing process of Carvana

We are challenged to develop an algorithm that automatically removes the photo studio background. This will allow Carvana to superimpose cars on a variety of backgrounds. A dataset of photos which cover different vehicles with a wide variety of year, make, and model combinations.

Proposed Method

- **Proposed Model Architecture**

The U-Net model proposed by Ronneberger et al. was chosen.

U-Net model resembles an autoencoder but with convolutions instead of a fully connected layer. It improves upon the "fully convolutional" architecture primarily through expanding the capacity of the decoder module of the network[3].

There is an encoding part with the convolution of decreasing dimension and decoder part with increasing dimension as shown in figure 2.

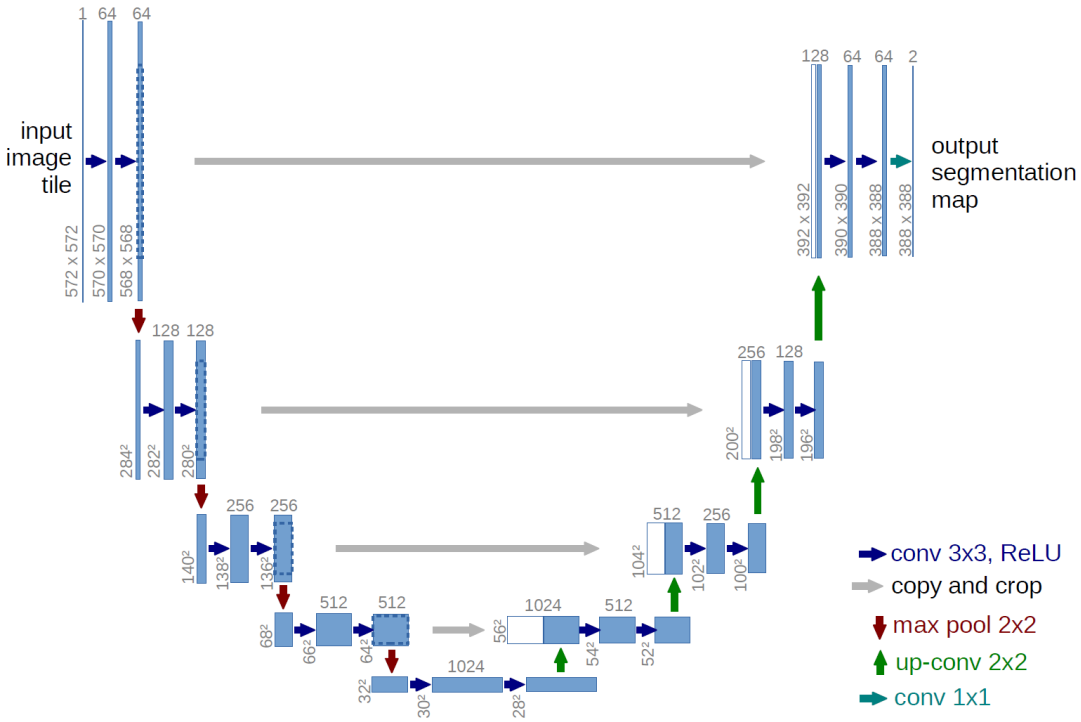

Figure 2: U-Net architecture proposed by Ronneberger et al.

- **Up-sampling Method**

Pooling layers down-sample the resolution by summarizing a local area with a single value whereas "unpooling" layers up-sample the resolution by distributing a single value into a higher resolution as shown in figure 3.

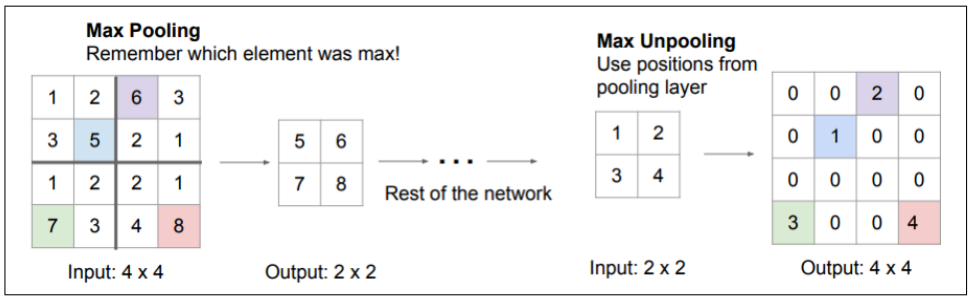

Figure 3: Max pooling and max un-pooling method

- **Performance matrix**

The performance matrix is a mean Dice coefficient or f1 score. The Dice coefficient can be used to compare the pixel-wise agreement between a predicted segmentation and its corresponding ground truth[2].

The formula is given by:

$$\frac{2\sum_{pixels} y_{true} y_{pred}}{\sum_{pixels} y_{true}^2 + \sum_{pixels} y_{pred}^2}$$

where y_{pred} is the predicted set of pixels and y_{true} is the ground truth.

- **Loss Function**

The loss function is a soft Dice loss which can be minimized because we directly use the predicted probabilities instead of thresholding and converting them into a binary mask[2].

The formula is given by:

$$1 - \frac{2\sum_{pixels} y_{true} y_{pred}}{\sum_{pixels} y_{true}^2 + \sum_{pixels} y_{pred}^2}$$

With respect to the neural network output, the numerator is concerned with the common activations between our prediction and target mask, whereas the denominator is concerned with the quantity of activations in each mask separately.

Experiments

1. **Data Preprocessing**
   a. **Resizing**
      All the image was resized to 1024*2014.
   b. **Augmentation**
      It was used to increase training data size and it has regularization effect.
   c. **Normalization**
      The image pixel was divided by 255 for normalization. This is because https://stackoverflow.com/questions/20486700/why-we-always-divide-rgb-values-by-255
2. **Model architecture**
   a. **Model summary**

Table 1 shows the details of each layer.

Table 1: Model summary

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 1024, 1024, 3)	0	
conv2d (Conv2D)	(None, 1024, 1024, 16)	448	input_1[0][0]
conv2d_1 (Conv2D)	(None, 1024, 1024, 16)	2320	conv2d[0][0]
max_pooling2d (MaxPoolind2D)	(None, 512, 512, 16)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 512, 512, 32)	4640	max_pooling2d[0][0]
conv2d_3 (Conv2D)	(None, 512, 512, 32)	9248	conv2d_2[0][0]
dropout (Dropout)	(None, 512, 512, 32)	0	conv2d_3[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 256, 256, 32)	0	dropout[0][0]

conv2d_4 (Conv2D)	(None, 256, 256, 64)	18496	max_pooling2d_1[0][0]
conv2d_5 (Conv2D)	(None, 256, 256, 64)	36928	conv2d_4[0][0]
dropout_1 (Dropout)	(None, 256, 256, 64)	0	conv2d_5[0][0]
up_sampling2d (UpSampling2D)	(None, 512, 512, 64)	0	dropout_1[0][0]
conv2d_6 (Conv2D)	(None, 512, 512, 32)	8224	up_sampling2d[0][0]
concatenate (Concatenate)	(None, 512, 512, 64)	0	conv2d_3[0][0] conv2d_6[0][0]
conv2d_7 (Conv2D)	(None, 512, 512, 32)	18464	concatenate[0][0]
conv2d_8 (Conv2D)	(None, 512, 512, 32)	9248	conv2d_7[0][0]
up_sampling2d_1 (UpSampling2D)	(None, 1024, 1024, 32)	0	conv2d_8[0][0]
conv2d_9 (Conv2D)	(None, 1024, 1024, 16)	2064	up_sampling2d_1[0][0]
concatenate_1 (Concatenate)	(None, 1024, 1024, 32)	0	conv2d_1[0][0] conv2d_9[0][0]
conv2d_10 (Conv2D)	(None, 1024, 1024, 16)	4624	concatenate_1[0][0]
conv2d_11 (Conv2D)	(None, 1024, 1024, 16)	2320	conv2d_10[0][0]
conv2d_12 (Conv2D)	(None, 1024, 1024, 2)	290	conv2d_11[0][0]
conv2d_13 (Conv2D)	(None, 1024, 1024, 1)	3	conv2d_12[0][0]

b. Optimizer

Adam with learning rate decay

c. Loss

Sum of binary cross entropy loss and soft dice loss.

3. Results

The model showed an unpromising segmentation result as shown in figure 4.

Figure 4: Segmentation result

Conclusion

The U-Net model showed an unpromising segmentation result. Further debug process is needed to find the root cause. One of the possible reason might be incorrect data preprocessing.

References

1. Carvana Image Masking Challenge. Retrieved from https://www.kaggle.com/c/carvana-image-masking-challenge
2. Jordan, J. (2018, May 21). An overview of semantic image segmentation. Retrieved from https://www.jeremyjordan.me/semantic-segmentation/
3. Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241). Springer, Cham.

Appendix

- **Version control**

Library	Version
Python	3.6.6
TensorFlow	1.10
Skimage	0.15
Numpy	1.15.1

- **Github**
  https://github.com/ChuaHanChong/kaggle_carvana_image_masking_challenge