MultiResUNet

MultiResUNet : Rethinking the U-Net Architecture for Multimodal Biomedical Image Segmentation

A

OWOP season1 02

조유민

Paper

Ibtehaz N, Rahman MS (2020) MultiResuNet: Rethinking the U-Net architecture for multimodal biomedical image segmentation. Neural Netw 121:74–87

MultiResUNet : Rethinking the U-Net Architecture for Multimodal Biomedical Image Segmentation

Nabil Ibtehaz[1] and M. Sohel Rahman[1,*]

[1]Department of CSE, BUET,
ECE Building, West Palasi, Dhaka-1205, Bangladesh
[*]Corresponding author

1017052037@grad.cse.buet.ac.bd , msrahman@cse.buet.ac.bd

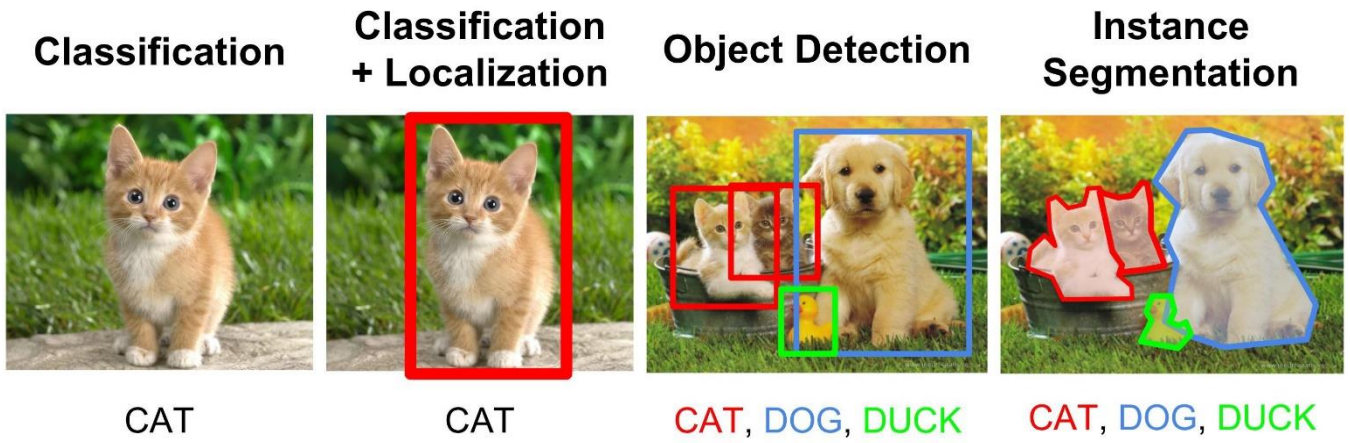
February 12, 2019

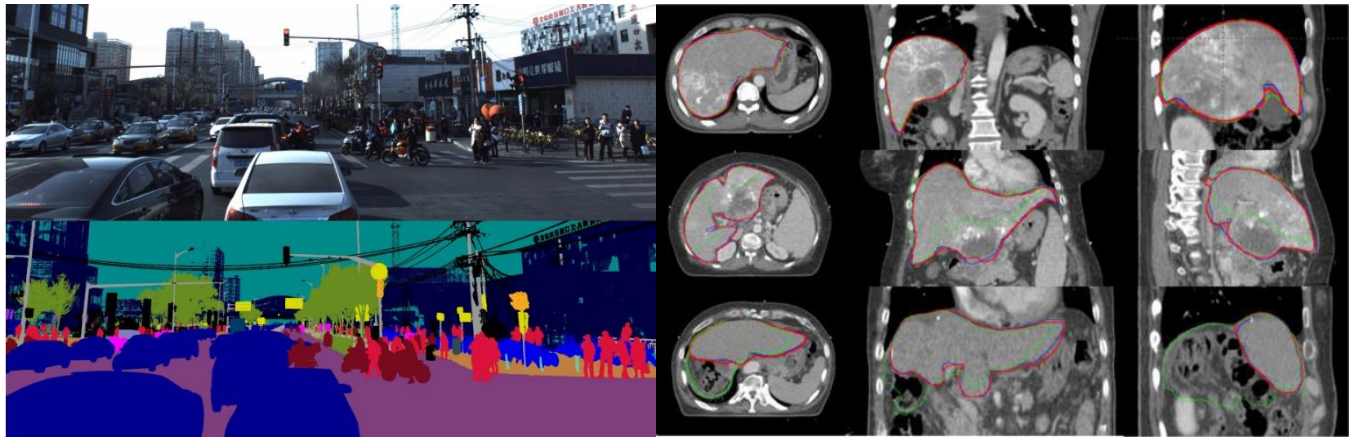
Abstract

In recent years Deep Learning has brought about a breakthrough in Medical Image Segmentation. U-Net is the most prominent deep network in this regard, which has been the most popular architecture in the medical imaging community. Despite outstanding overall performance in segmenting multimodal medical images, from extensive experimentations on challenging datasets, we found out that the classical U-Net architecture seems to be lacking in certain aspects. Therefore, we propose some modifications to improve upon the already state-of-the-art U-Net model. Hence, following the modifications we develop a novel architecture MultiResUNet as the potential successor to the successful U-Net architecture. We have compared our proposed architecture MultiResUNet with the classical U-Net on a vast repertoire of multimodal medical images. Albeit slight improvements in the cases of ideal images, a remarkable gain in performance has been attained for challenging images. We have evaluated our model on five different datasets, each with their own unique challenges, and have obtained a relative improvement in performance of 10.15%, 5.07%, 2.63%, 1.41%, and 0.62% respectively.

Index terms— Convolutional Neural Networks, Medical Imaging, Semantic Segmentation, U-Net

Image Segmentation





Introduction

1 Introduction

Since the introduction of digital medical imaging equipments, significant attention has been drawn towards applying image processing techniques in analyzing medical images. Multidisciplinary researchers have been working diligently for decades to develop automated diagnosis systems, and to this day it is one of the most active research areas [1]. The task of a computer aided medical image analysis tool is twofold: segmentation and diagnosis. In the general Semantic Segmentation problem, the objective is partitioning an image into a set of non-overlapping regions, which allows the homogeneous pixels to be clustered together [2]. However, in the context of medical images the interest often lies in distinguishing only some interesting areas of the image, like the tumor regions [3], organs [4] etc. This enables the doctors to analyze only the significant parts of the otherwise incomprehensible multimodal medical images [5]. Furthermore, often the segmented images are used to compute various features that may be leveraged in the diagnosis [6]. Therefore, image segmentation is of utmost importance and application in the domain of Biomedical Engineering.

⋮

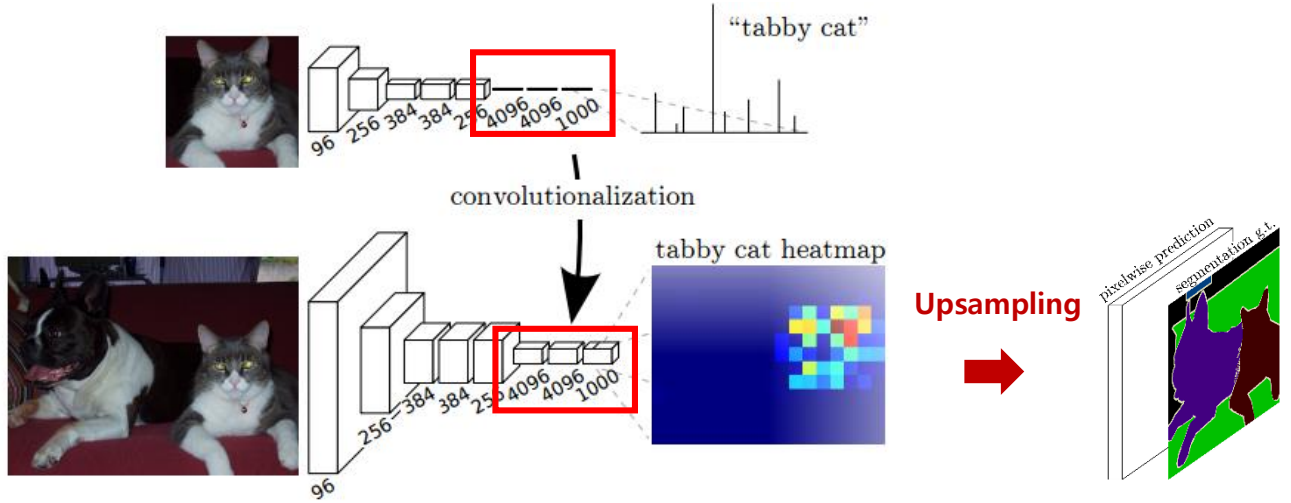
Recent advancements in deep learning [10] have shwon a lot of promises towards solving such issues. In this regard, Convolutional Neural Networks (CNN) [11] have been the most ground-breaking addition, which is dominating the field of Computer Vision. CNNs have been responsible for the phenomenal advancements in tasks like object classification [12], object localization [13] etc., and the continuous improvements to CNN architectures are bringing further radical progresses [14, 15, 16, 17]. Semantic Segmentation tasks have also been revolutionalized by Convolutional Networks. Since CNNs are more intuitive in performing object classification, Ciresan et al. [18] presented a sliding window based pipeline to perform semantic segmentation using CNN. Long et al. [19] proposed a fully convolutional network (FCN) to perform end-to-end image segmentation, which surpassed the existing approaches. Badrinarayanan et al. [20] improved upon FCN, by developing a novel architecture namely, SegNet. SegNet consists of a 13 layer deep encoder network that extracts spatial features from the image, and a corresponding 13 layer deep decoder network that upsamples the feature maps to predict the segmentation masks. Chen et al. [21] presented DeepLap and performed semantic segmentation using atrous convolutions.

In spite of initiating a breakthrough in computer vision tasks, a major drawback of the CNN architectures is that they require massive volumes of training data. Unfortunately, in the context of medical images, not only the acquisition of images is expensive and complicated, accurate annotation thereof adds even more to the complexity [22]. Nevertheless, CNNs have shown great promise in medical image segmentation in recent years [22, 23], and most of the credit go to U-Net [24]. The structure of U-Net is quite similar to SegNet, comprising an encoder and a decoder network. Furthermore, the correspond-

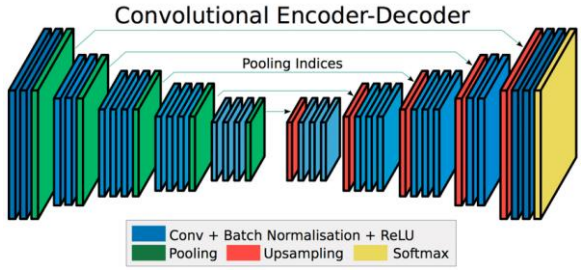
CNN vs FCN

FC layer를 **1x1 convolution layer**로 대체하여 spatial coordinates를 끝까지 유지함

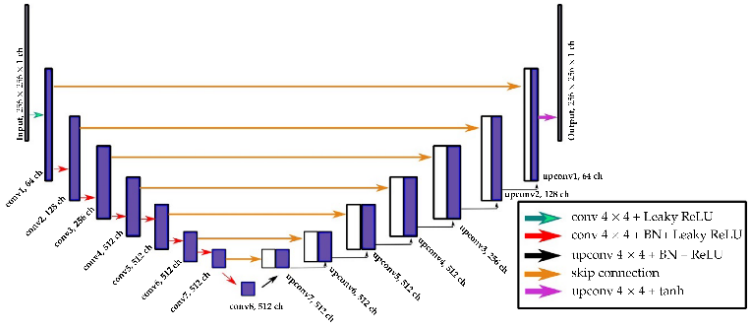
Upsampling을 통해 feature map을 input 크기만큼 다시 늘림



SegNet



U-Net



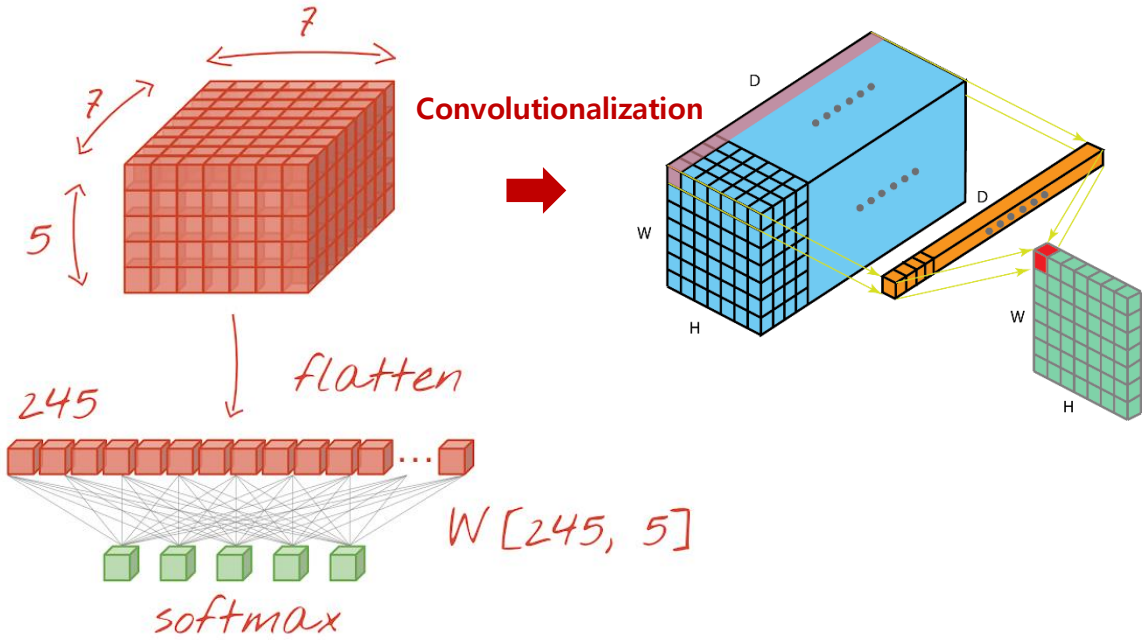
1x1 Convolution

Convolutionalization / Bottleneck / Network In Network

Problem

FC layers throw away spatial coordinates

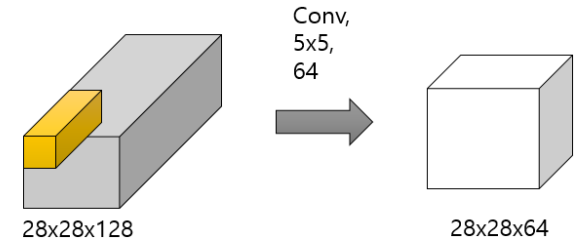
모든 node들이 서로 곱해짐에 따라 feature map의 spatial coordinates를 모두 잃어버림



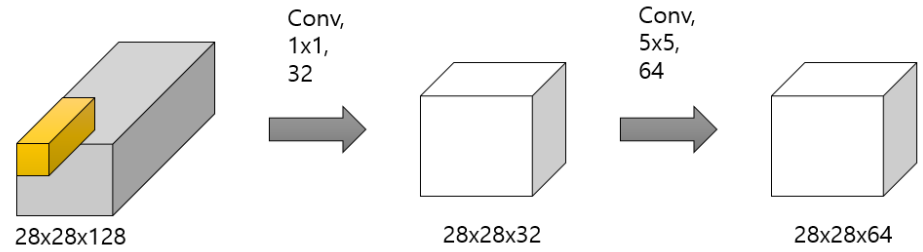
Solution

Replace FC layer with 1x1 convolution layer

- 1x1 convolution layer를 사용하면 feature map의 크기는 그대로 유지하되 channel 수를 자유롭게 조절할 수 있음(**dimension reduction**)
- 모델을 deep하게 만들더라도 memory 문제에 대한 부담을 덜 수 있음



#params = 28x28x64x5x5x128 = 160M



#params = 28x28x32x128x1x1 = 4.8M

#params = 28x28x64x5x5x32 = 40M

#total = 44.8M

[출처] https://hwiyong.tistory.com/45

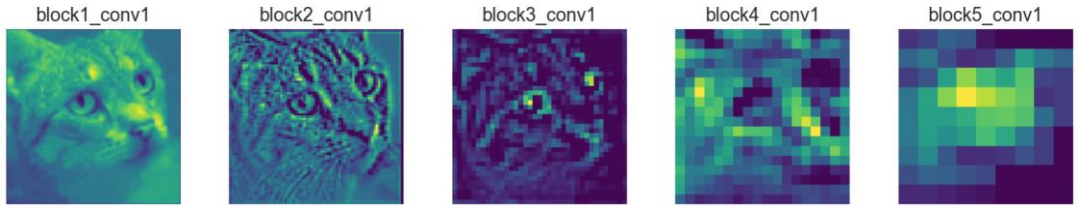
Upsampling & Skip Connection

Shortcut connection / Residual connection

Problem 1

Outputs are coarse rather than dense

subsampling에 의해 feature map의 크기가 줄어들어 detail이 많이 사라짐

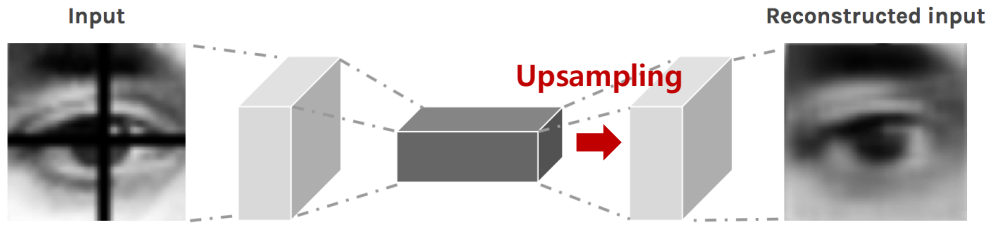


Solution 1

Upsampling

원본 이미지에 가깝게 늘려 coarse map을 dense map으로 만들어 줌

ex) Bilinear interpolation, Transposed convolution, Unpooling, etc.



Problem 2

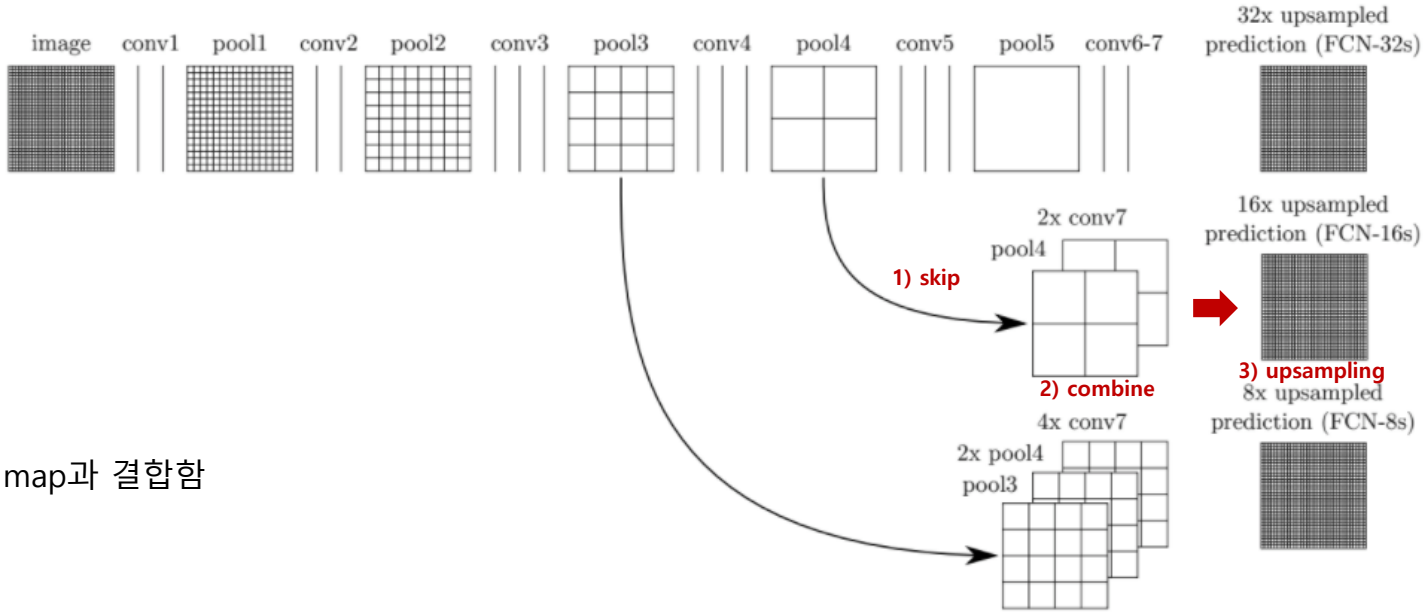
Dense prediction with pixel-wise loss

이미 많은 feature가 소실된 coarse map을 upsampling한 dense map에는 여전히 detail이 누락되어 있음

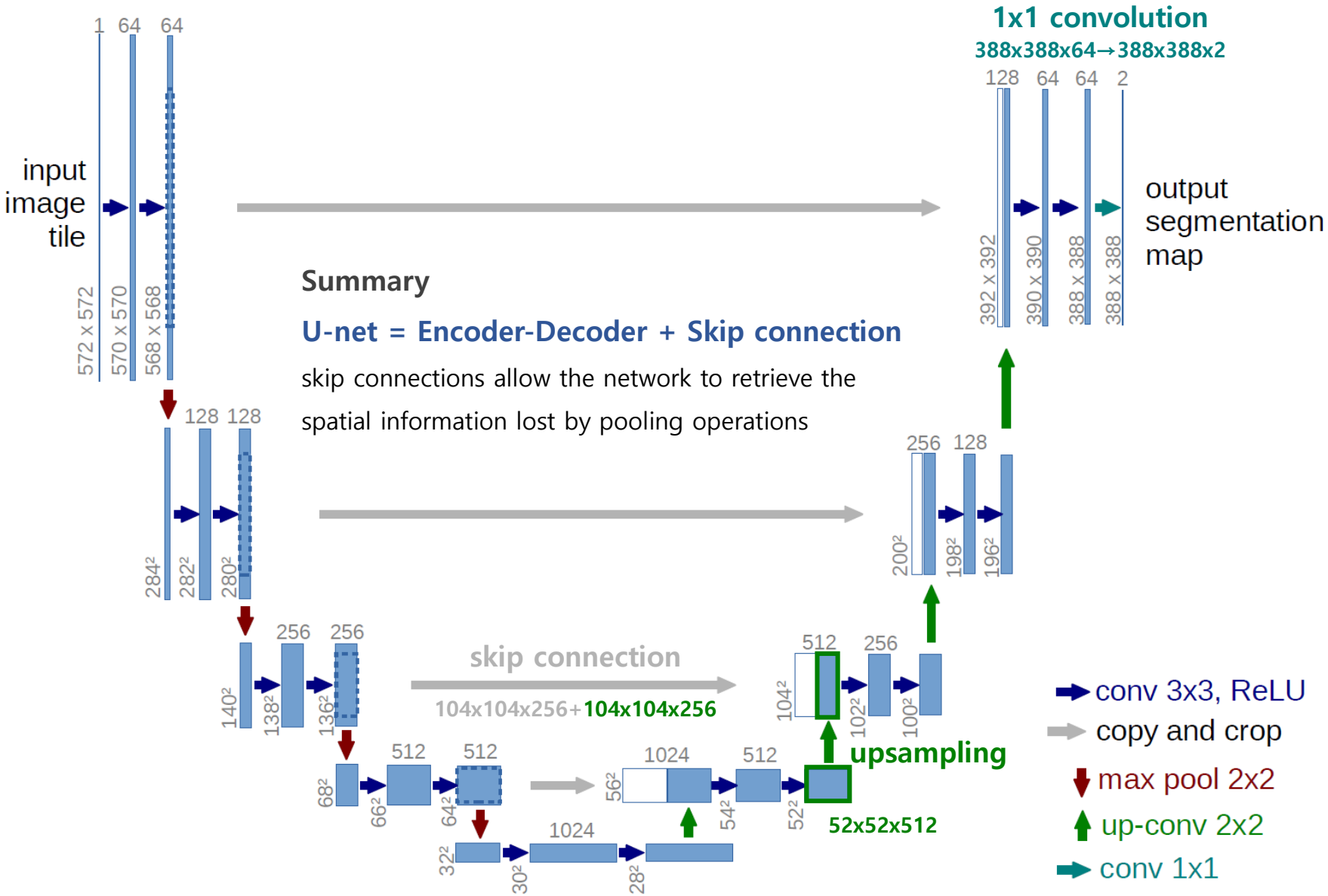
Solution 2

Skip connection

조금 더 dense한 이전 Layer의 feature map을 가져와 coarse map과 결합함



Overview of the UNet Architecture



Motivations and Considerations

3.1 Variation of Scale in Medical Images

In medical image segmentation, we are interested in segmenting cell necluei [36], organs [4], tumors [3] etc. from images originating from various modalities. However, in most cases these objects of interest are of irregular and different scales. For example, in Figure 2 we have demonstrated that the scale of skin lesions can greatly vary in dermoscopy images. These situations frequently occur in different types medical image segmentation tasks.

(a) (b) (c)

⋮

Therefore, a network should be robust enough to analyze objects at different scales. Although this issue has been addressed in several deep computer vision works, to the best of our knowledge, this issue is still not addressed properly in the domain of medical image segmentation. Serre et al. [37] employed a sequence of fixed Gabor filters of varying scales to acknowledge the variation of scale in the image. Later on, the revolutionary Inception architecture [15] introduced Inception blocks, that utilizes convolutional layers of varying kernel sizes in parallel to inspect the points of interest in images from different scales. These perceptions obtained at different scales are combined together and passed on deeper into the network.

In the U-Net architecture, after each pooling layer and transposed convolutional layer a sequence of two 3×3 convolutional layers are used. As explained in [34], this series of two 3×3 convolutional operation actually resembles a 5×5 convolutional operation. Therefore, following the approach of Inception network, the simplest way to augment U-Net with multi-resolutional analysis is to incorporate 3×3, and 7×7 convolution operations in parallel to the 5×5 convolution operation, as shown in Figure 3a.

Problem 1

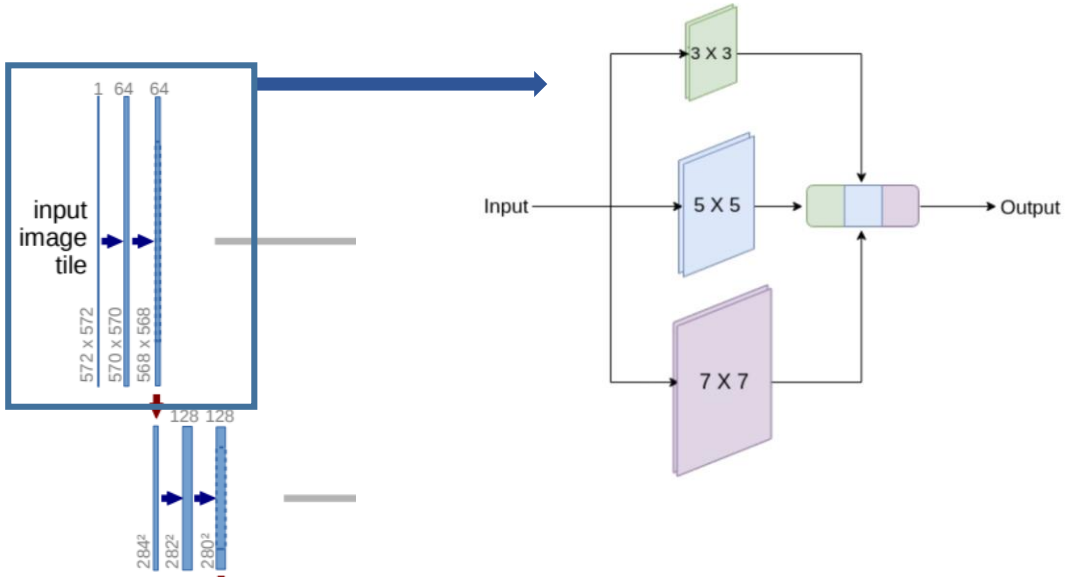
Not robust enough at different scales

대부분의 이미지는 scale이 다양하지만 U-Net은 이를 고려하지 않음

Solution 1

Replace convolution layer with Inception-like blocks

다양한 크기의 이미지에서 학습한 spatial feature가 서로 조화를 이루도록 함



Inception Module

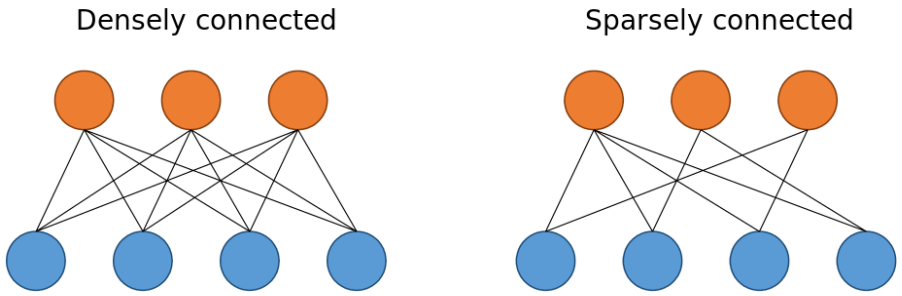
Inception block

Problem

Dense connections are expensive, but sparse matrix are slow

network가 깊어질수록 overfitting, vanishing gradient, memory 등의 문제가 발생함

dropout 등의 기법으로 spare하게 만들어 줄 수 있지만 computer resource가 늘어남

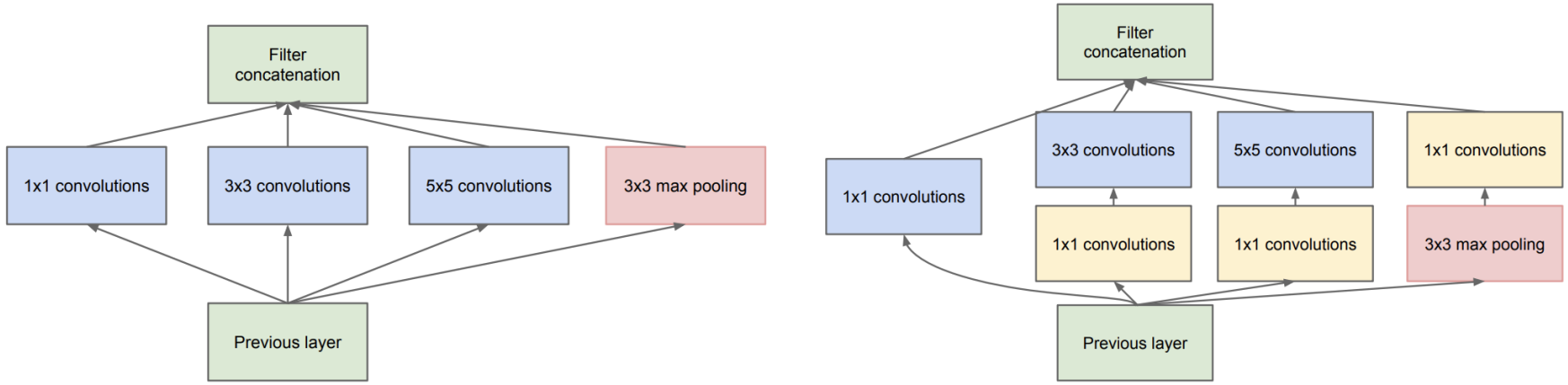


Solution

Concatenate the layers in parallel with 1x1 convolution

convolution layer와 pooling layer를 병렬적으로 쌓음(sparse)

행렬 연산을 다시 합치되(dense) 1x1 convolution을 추가해 균형을 맞춤

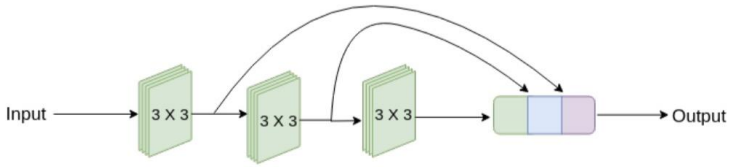


(a) Inception module, naïve version

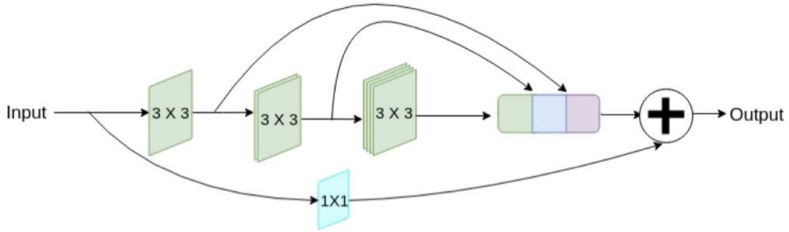
(b) Inception module with dimension reductions

Motivations and Considerations

② increases the memory requirement. Therefore, we improvise with the following ideas borrowed from [34]. We factorize the bigger, more demanding 5×5 and 7×7 convolutional layers, using a sequence of smaller and lightweight 3×3 convolutional blocks, as shown in Figure 3b. The outputs of the 2nd and 3rd 3×3 convolutional blocks effectively approximate the 5×5 and 7×7 convolution operations respectively. We hence take the outputs from the three convolutional blocks and concatenate them together to extract the spatial features from different scales. From our experiments, it was seen that the results of this compact block closely resemble that of the memory intensive Inception-like block decsribed earlier. This outcome is in line with the findings of [34], as the adjacent layers of a vision network are expected to be correlated.



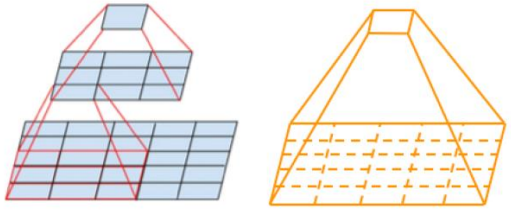
Despite that this modification greatly reduces the memory requirement, it is still quite demanding. This is mostly due to the fact that in a deep network if two convolutional layers are present in a succession, then ③ the number of filters in the first one has a quadratic effect over the memory [15]. Therefore, instead of keeping all the three consecutive convolutional layers of equal number of filters, we gradually increase the filters in those (from 1 to 3), to prevent the memory requirement of the earlier layers from exceedingly propagating to the deeper part of the network. We also add a residual connection because of their efficacy in biomedical image segmentaion [33], and for the introduction of the 1×1 convolutional layers, which may allow us to comprehend some additional spatial information. We call this arrangement a '*MultiRes block*', as shown in Figure 3c.



Problem 2

High memory requirement

filter 크기가 커졌기 때문



Solution 2

Factorize the large filters into a succession of smaller filters

3x3 convolution layer를 쌓아 만든 block으로 5x5, 7x7을 approximate

Problem 3

Quadratic effect over the memory

중첩된 convolution layer로 인해 여전히 memory 문제가 남아 있음

Solution 3

Gradually increase the filters and add residual connection

3x3 filter를 단계별로 조금씩 겹쳐 쌓아 올리고, 1x1 convolutional layer로 dimension을 줄여 memory가 급격하게 늘어나는 것을 막음

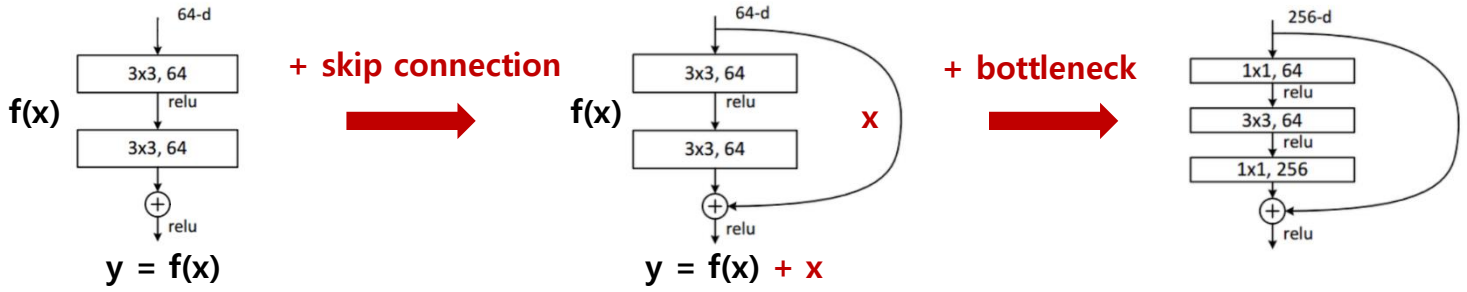
∴ **MultiRes block**

Residual connection

Summary

Train deeper networks by learning residual of input and output

이전 layer의 정보를 연결해 기존에 학습한 feature를 보존함으로써 추가로 학습해야 할 양을 줄임



[MultiResUnet]

```
shortcut = inp
shortcut = conv2d_bn(shortcut, filters, 1, 1,
                     activation=None, padding='same')

out = conv2d_bn(inp, filters, 3, 3, activation='relu', padding='same')

out = add([shortcut, out])
out = Activation('relu')(out)
out = BatchNormalization(axis=3)(out)

for i in range(length-1):

    shortcut = out
    shortcut = conv2d_bn(shortcut, filters, 1, 1,
                         activation=None, padding='same')

    out = conv2d_bn(out, filters, 3, 3, activation='relu', padding='same')

    out = add([shortcut, out])
    out = Activation('relu')(out)
    out = BatchNormalization(axis=3)(out)
```

[Unet]

```
inputs = Input(input_size)
conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(inputs)
conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv1)
pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool1)
conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv2)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool2)
conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv3)
pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool3)
conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv4)
drop4 = Dropout(0.5)(conv4)
pool4 = MaxPooling2D(pool_size=(2, 2))(drop4)

conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool4)
conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv5)
drop5 = Dropout(0.5)(conv5)

up6 = Conv2D(512, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(drop5))
merge6 = concatenate([drop4,up6], axis = 3)
conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge6)
conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv6)
```

Motivations and Considerations

3.2 Probable Semantic Gap between the Corresponding Levels of Encoder-Decoder

An ingenious contribution of the U-Net architecture was the introduction of shortcut connections between the corresponding layers before and after the max-pooling and the deconvolution layers respectively. This enables the network to propagate from encoder to decoder, the spatial information that gets lost during the pooling operation.

⋮

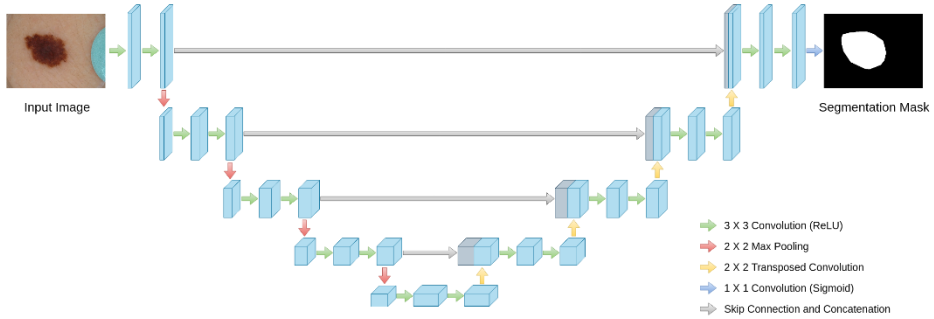
before the first pooling with the decoder after the last deconvolution operation. Here, the features coming from the encoder are supposed to be lower level features as they are computed in the earlier layers of the network. On the contrary, the decoder features are supposed to be of much more higher level, since they are computed at the very deep layers of the network. Therefore, they go through more processing. Hence, we observe a possible semantic gap between the two sets of features being merged. We conjecture that the fusion of these two arguably incompatible sets of features could cause some discrepancy throughout the learning and thus adversely affect the prediction procedure. It may be noted that, the amount of discrepancy is likely to decrease gradually as we move towards the succeeding shortcut connections. This can be attributed to the fact that, not only the features from the encoder are going through more processing, but also we are fusing them with decoder features of much juvenile layers.

Therefore, to alleviate the disparity between the encoder-decoder features, we propose to incorporate some convolutional layers along the shortcut connections. Our hypothesis is that these additional non-linear transformations on the features propagating from the encoder stage should account for the further processing done during the by decoder stage therein. Furthermore, instead of using the usual convolutional layers we introduce residual connections to them as they make the learning easier [17] and is proven to be having great potential in medical image analysis [33]. This idea is inspired from the image to image conversion using convolutional neural networks [39], where pooling layers are not favorable for the loss of information. Thus, instead of simply concatenating the feature maps from the encoder stage to the decoder stage, we first pass them through a chain of convolutional layers with residual connections, and then concatenate with the decoder features. We call this proposed shortcut path '*Res path*', illustrated in Fig. 4. Specifically, 3×3 filters are used in the convolutional layers and 1×1 filters accompany the residual connections.

Problem 4

Semantic gap between encoder and decoder features

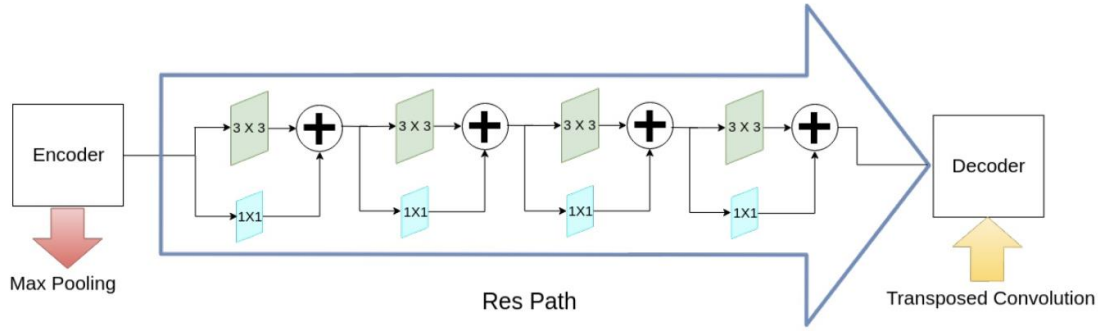
U-net의 skip connection은 서로 다른 level의 feature를 결합함



Solution 4

Incorporate non-linear operations to skip connections

encode의 feature map을 residual connection을 포함한 convolution layer에 통과시킨 뒤에 decoder의 feature map과 결합함



∴ **Res Path**

Proposed Architecture

4 Proposed Architecture

In the MultiResUNet model, we replace the sequence of two convolutional layers with the proposed *MultiRes* block as introduced in Section 3.2. For each of the *MultiRes* blocks, we assign a parameter W, which controls the number of filters of the convolutional layers inside that block. To maintain a comparable relation between the numbers of parameters in original U-Net and the proposed model, we compute the value of W as follows:

$$W = \alpha \times U \tag{1}$$

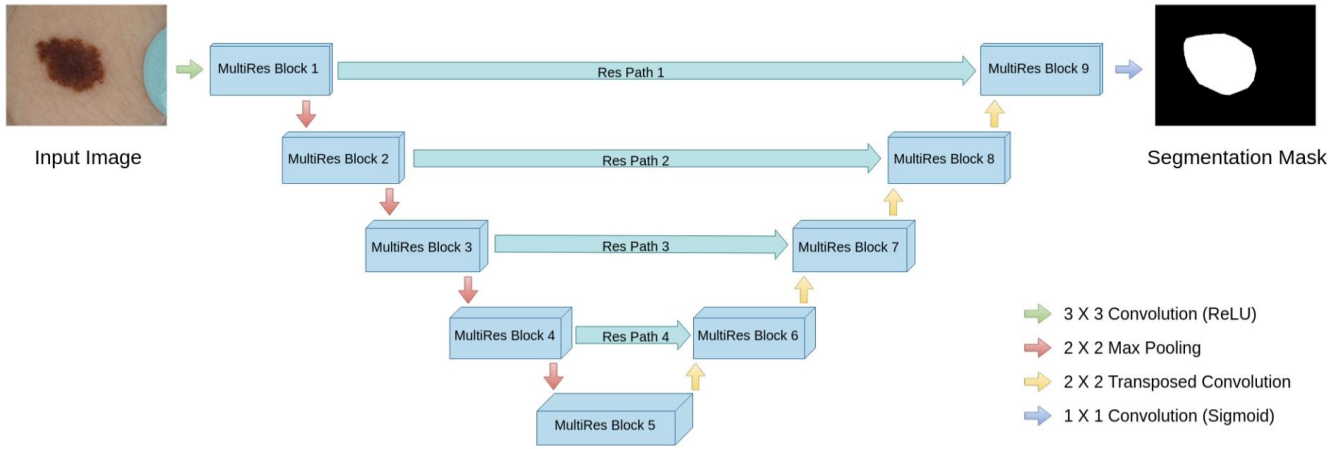
Here, U is the number of filters in the corresponding layer of the U-Net and α is a scaler coefficient. Decomposing W to U and α provides a convenient way to both controlling the number of parameters and keeping them comparable to U-Net. We compare our proposed model with an U-Net, having $\#filters = [32, 64, 128, 256, 512]$ along the levels, which are also the values of U in our model. We selected $\alpha = 1.67$ as it keeps the number of parameters in our model slightly below that of the U-Net.

In Section 3.2, we pointed out that it is beneficial to gradually increase the number of filters in the successive convolutional layers inside a *MultiRes* block, instead of keeping them the same. Hence, we assign $\lfloor \frac{W}{6} \rfloor$, $\lfloor \frac{W}{3} \rfloor$ and $\lfloor \frac{W}{2} \rfloor$ filters to the three successive convolutional layers respectively, as this combination achieved the best results in our experiments. Also it can be noted that similar to the U-Net architecture, after each pooling or deconvolution operation the value of W gets doubled.

In addition to introducing the *MultiRes* blocks, we also replace the ordinary shortcut connections with the proposed *Res* paths. Therefore, we apply some convolution operations on the feature maps propagating from the encoder stage to the decoder stage. In Section 3.1, we hypothesized that the intensity of the semantic gap between the encoder and decoder feature maps are likely to decrease as we move towards the inner shortcut paths. Therefore, we also gradually reduce the number of convolutional blocks used along the *Res* paths. In particular, we use $4, 3, 2, 1$ convolutional blocks respectively along the four *Res* paths. Also, in order to account for the number of feature maps in encoder-decoder, we use $32, 64, 128, 256$ filters in the blocks of the four *Res* paths respectively.

All the convolutional layers except for the output layer, used in this network is activated by the *ReLU* (Rectified Linear Unit) activation function [10], and are batch-normalized [35]. Similar to the U-Net model, the output layer is activated by a *Sigmoid* activation function. We present a diagram of the proposed MultiResUNet model in Fig. 5. The architectural details are described in Table 1.

MultiResUNet					
Block	Layer(filter size)	#filters	Path	Layer(filter size)	#filters
MultiRes Block 1	Conv2D(3,3)	8	Res Path 1	Conv2D(3,3)	32
	Conv2D(3,3)	17		Conv2D(1,1)	32
MultiRes Block 9	Conv2D(3,3)	26		Conv2D(3,3)	32
	Conv2D(1,1)	51		Conv2D(1,1)	32
MultiRes Block 2	Conv2D(3,3)	17		Conv2D(3,3)	32
	Conv2D(3,3)	35		Conv2D(1,1)	32
MultiRes Block 8	Conv2D(3,3)	53		Conv2D(3,3)	32
	Conv2D(1,1)	105		Conv2D(1,1)	32
MultiRes Block 3	Conv2D(3,3)	35	Res Path 2	Conv2D(3,3)	64
	Conv2D(3,3)	71		Conv2D(1,1)	64
MultiRes Block 7	Conv2D(3,3)	106		Conv2D(3,3)	64
	Conv2D(1,1)	212		Conv2D(1,1)	64
MultiRes Block 4	Conv2D(3,3)	71		Conv2D(3,3)	64
	Conv2D(3,3)	142		Conv2D(1,1)	64
MultiRes Block 6	Conv2D(3,3)	213	ResPath 3	Conv2D(3,3)	128
	Conv2D(1,1)	426		Conv2D(1,1)	128
MultiRes Block 5	Conv2D(3,3)	142		Conv2D(3,3)	128
	Conv2D(3,3)	284		Conv2D(1,1)	128
	Conv2D(3,3)	427	Res Path 4	Conv2D(3,3)	256
	Conv2D(1,1)	853		Conv2D(1,1)	256



Experiments

6.3 Training Methodology

The task of semantic segmentation is to predict the individual pixels whether they represent a point of interest, or are merely a part of the background. Therefore, this problem ultimately reduces to a pixel-wise binary classification problem. Hence, as the loss function of the network we simply took the binary cross-entropy function and minimized it.

Let, for an image X, the ground truth segmentation mask is Y, and the segmentation mask predicted by the model is $\hat{Y}$. For a pixel px, the network predicts $\hat{y}_{px}$, whereas, the ground truth value is y_{px}. The binay cross-entropy loss for that image is defined as:

$$Cross\ Entropy(X, Y, \hat{Y}) = \sum_{px \in X} -(y_{px} \log(\hat{y}_{px}) + (1 - y_{px}) \log(1 - \hat{y}_{px})) \quad (2)$$

For a batch containing n images the loss function J becomes,

$$J = \frac{1}{n} \sum_{i=1}^{n} Cross\ Entropy(X_i, Y_i, \hat{Y}_i) \quad (3)$$

We minimized the binary cross-entropy loss, hence trained the model using the Adam optimizer [49]. Adam adaptively computes different learning rates for different parameters from estimates of first and second moments of the gradients. This idea, in fact combines the advantages of both AdaGrad [50] and RMSProp [51]; therefore Adam has been often used in benchmarking deep learning models as the default choice [52]. Adam has a number of parameters including β_1 and β_2 which control the decay of first and second moment respectively. However, in this work we used Adam with the parameters mentioned in the original paper. The models were trained for 150 epochs using Adam optimizer. The reason of selecting 150 as the number of epochs is due to the fact that after 150 epochs neither of the models were showing any further improvements.

6.4 Evaluation Metric

In semantic segmentation, usually the points of interest comprise a small segment of the entire image. Therefore, metrics like precision, recall are inadequate and often lead to false sense of superiority, inflated by the perfection of detecting the background. Hence, Jaccard Index has been widely used to evaluate and benchmark image segmentation and

Loss function

Binary Cross-entropy loss

foreground vs background

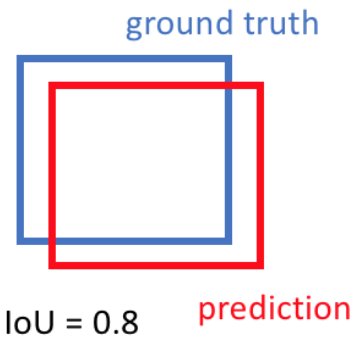
Optimizer

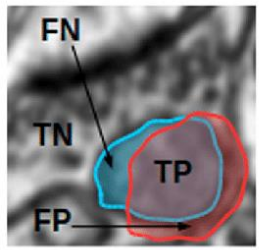
Adam

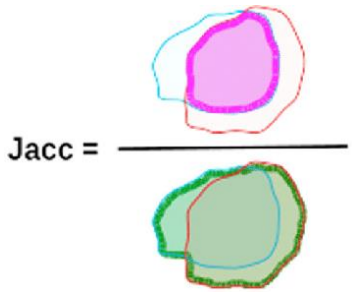
150 epochs

Evaluation metric

Jaccard index(Intersection over Union)







5-fold Cross validation

Results

7 Results

7.1 MultiResUNet Consistently Outperforms U-Net

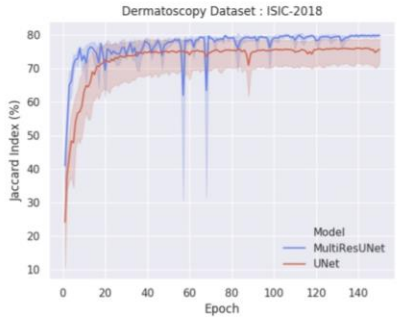
As described in Section 5 and Section 6 to evaluate the performance of the proposed architecture, we performed experiments with diversified classes of medical images, each with a unique challenge of its own. In particular, we have performed 5-fold cross validation and observed the performance of our proposed MultiResUNet and the baseline, U-Net. In each run the best results obtained on the validation set through the 150 epochs performed was noted and they were summarised from the 5 runs to obtain the final result.

Modality	MultiResUNet (%)	U-Net (%)	Relative Improvement (%)
Dermoscopy	80.2988 ± 0.3717	76.4277 ± 4.5183	5.065
Endoscopy	82.0574 ± 1.5953	74.4984 ± 1.4704	10.1465
Fluorescence Microscopy	91.6537 ± 0.9563	89.3027 ± 2.1950	2.6326
Electron Microscopy	87.9477 ± 0.7741	87.4092 ± 0.7071	0.6161
MRI	78.1936 ± 0.7868	77.1061 ± 0.7768	1.4104

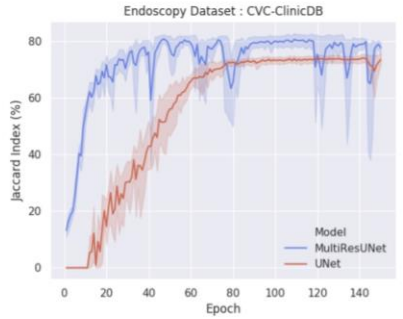
7.2 MultiResUNet can Obtain Better Results in Less Number of Epochs

In addition to analyzing the best performing models from each run, we also monitored how the model performance progressed with epochs. In Figure 6, the performance on the validation data on each epoch is shown, for all the datasets. We have presented the band of Jaccard Index values at a certain epoch in the 5-fold cross validation. It can be noted that for all the cases our proposed model attains convergance much faster. This can be attributed to the synergy between residual connections and batch normalization [33]. Moreover, apart from Fig. 6d in all other cases the MultiResUNet model consistently outperformed the classical U-Net model. In spite of lagging behind the U-Net at the beginning for the electron microscopy images (Fig. 6d), eventually the MultiResUNet model converges at a better accuracy than U-Net. Another remarkable observation from the experiments is that except for some minor flactuations, the standard deviation of the performance of the MultiResUNet is much smaller; this indicates the reliability and the robustness of the proposed model.

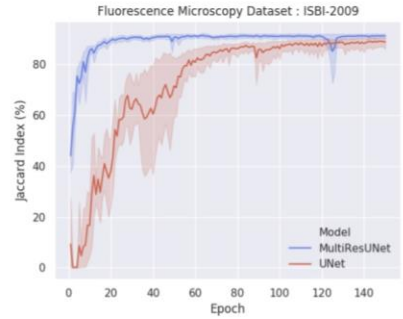
These results, therefore, suggest that using the proposed MultiResUNet architecture, we are likely to obtain superior results in less number of training epochs as compared to the classical U-Net architecture.



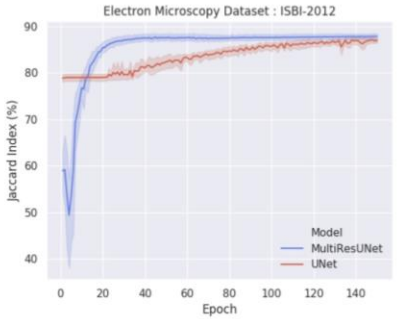
(a) Dermatoscopy



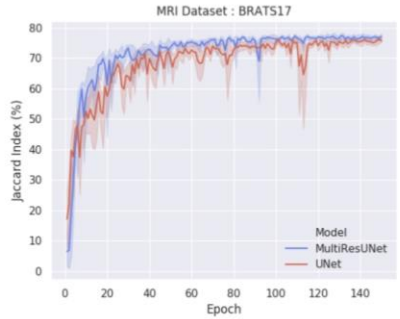
(b) Endoscopy



(c) Fluorescence Microscopy



(d) Electron Microscopy

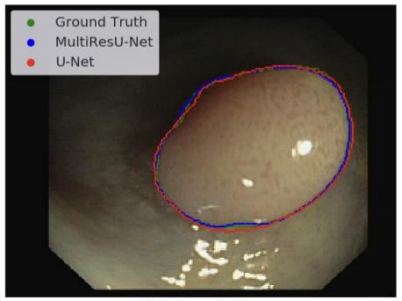


(e) MRI

Results

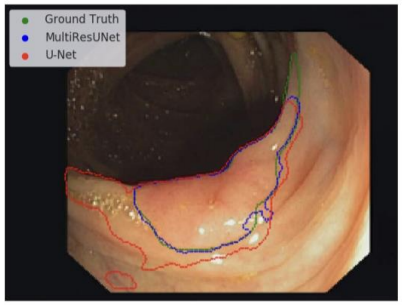
7.3 MultiResUNet Delineates Faint Boundaries Better

Clear boundaries



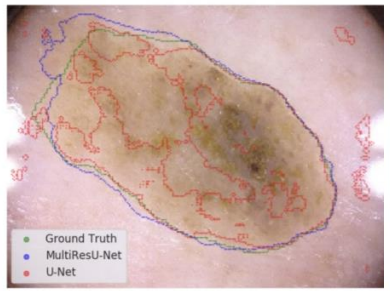
Comparison

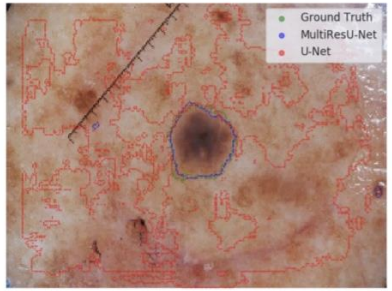
Faint boundaries

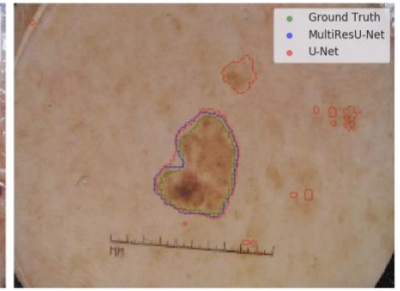


7.4 MultiResUNet is More Immune to Perturbations

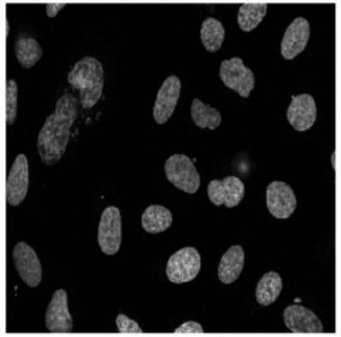
Irregular or plain foreground/background



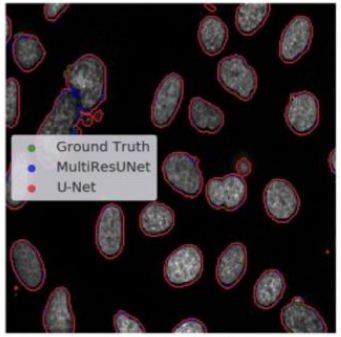




7.5 MultiResUNet is More Reliable Against Outliers

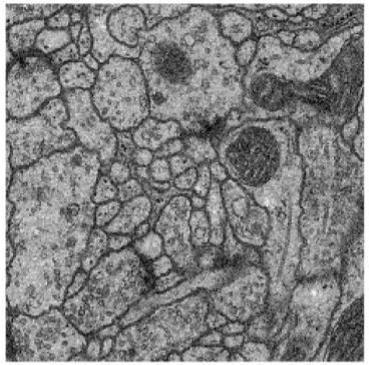


Input Image

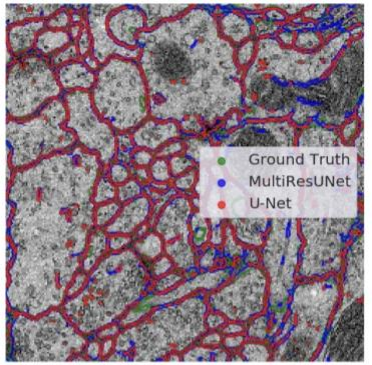


Comparison

7.6 Note on Segmenting the Majority Class



(a) Input Image



(e) Comparison

Conclusion

- We analyze the U-Net model architecture in depth, and conjecture some potential opportunities for further enhancements
- Based on the probable scopes for improvement, we propose MultiResUNet, which is an enhanced version of the standard U-Net architecture.
- We experiment with different public medical image datasets of different modalities, and MultiResUNet shows superior accuracy.
- We also experiment with a 3D version of MultiResUNet, and it outperforms the standard 3D U-Net as well.
- Particularly, we examine some very challenging images and observe a significant improvement in using MultiResUNet over U-Net.

References

- **Dense map vs Coarse map**

Visualizing and Understanding Convolutional Networks, Matthew D. Zeiler, Courant Institute, New York University, 2013

- **1x1 convolution**

Network In Network, Min Lin, National University of Singapore, 2013

- **Inception module(GoogleNet)**

Going Deeper with Convolutions, Christian Szegedy, Google Inc, 2014

- **Residual block(ResNet)**

Deep Residual Learning for Image Recognition, Kaiming He, Microsoft Research, 2015

- **FCN**

Fully Convolutional Networks for Semantic Segmentation, Jonathan Long, University of Berkeley, 2014

- **U-net**

U-Net: Convolutional Networks for Biomedical Image Segmentation, Olaf Ronneberger, University of Freiburg , 2015