May 2019

SEMANTIC IMAGE SEGMENTATION VIA A DENSE PARALLEL NETWORK

Jiyang Wang
Syracuse University

ABSTRACT

Image segmentation has been an important area of study in computer vision. Image segmentation is a challenging task, since it involves pixel-wise annotation, i.e. labeling each pixel according to the class to which it belongs. In image classification task, the goal is to predict to which class an entire image belongs. Thus, there is more focus on the abstract features extracted by Convolutional Neural Networks (CNNs), with less emphasis on the spatial information. In image segmentation task, on the other hand, the abstract information and spatial information are needed at the same time. One class of work in image segmentation focuses on “recovering” the high-resolution features from the low resolution ones. This type of network has an encoder-decoder structure, and spatial information is recovered by feeding the decoder part of the model with previous high-resolution features through skip connections. Overall, these strategies involving skip connections try to propagate features to deeper layers. The second class of work, on the other hand, focuses on “maintaining" high resolution features throughout the process.

In this thesis, we first review the related work on image segmentation and then introduce two new models, namely Unet- Laplacian and Dense Parallel Network (DensePN). The Unet-Laplacian is a series CNN model, incorporating a Laplacian filter branch. This new branch performs Laplacian filter operation on the input RGB image, and feeds the output to the decoder. Experiments results show that, the output of the Unet-Laplacian captures more of the ground truth mask, and eliminates some of the false positives. We then describe the proposed DensePN, which was designed to find a good balance between extracting features through multiple layers and keeping spatial information. DensePN allows not only keeping high-resolution feature maps but also feature reuse at deeper layers to solve the image segmentation problem. We have designed the Dense Parallel Network based on three main observations that we have gained from our initial trials and preliminary studies. First,

maintaining a high resolution feature map provides good performance. Second, feature reuse is very efficient, and allows having deeper networks. Third, having a parallel structure can provide better information flow. Experimental results on the CamVid dataset show that the proposed DensePN (with 1.1M parameters) provides a better performance than FCDense56 (with 1.5M parameters) by having less parameters at the same time.

SEMANTIC IMAGE SEGMENTATION VIA A DENSE PARALLEL NETWORK

By

Jiyang Wang
B.S., Anhui University of Science and Technology, Anhui, China, 2017

THESIS

Submitted in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering

Syracuse University
May 2019

ACKNOWLEDGMENTS

First, I would like to thank my thesis advisor Dr. Senem Velipasalar of the Electrical Engineering and Computer Science Department at Syracuse University. She helped me a lot with my research on computer vision as well as this thesis. One year ago, I became interested in computer vision because of her course 'Image and Video Processing'. Whenever I have questions, she always helps me with patience until I fully understand. Following her guidance, I learnt every details of the study of this thesis, including but not limited to reading research papers, thinking about a problem, testing models and writing. Most importantly, she showed to never give up. Hearing "Research is search and re-search" from her helped me to get through the tough times when I encountered a wall on my way. Her suggestions developed my thought opening my mind.

I would also like to thank my committee members Prof. Pramod K. Varshney, Prof. Mustafa C. Gursoy and Prof. Reza Zafarani for their valuable time, help and comments.

I would like to thank the Ph.D. students and my friends in the Smart Vision Systems Lab, namely Burak Kakillioglu, Fatih Altay, Mu Bai and Yantao Lu. They helped me a lot with many things in the past year. Burak, Fatih and Mu also helped me by proofreading my thesis. I can always get new ideas after talking with them. Looking up to them encourages me to learn and push my limits.

Last but not least, I want to thank my family and friends for their continuous support during my years of study. Without them, I could not complete this thesis.

My parents show very much respect for education and research, and they always encourage and support me to continue my studies. Thanks a lot.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

In this chapter, the image segmentation problem will be discussed together with its potential applications. Then, the method proposed in this thesis will be introduced.

1.1 Image Segmentation

Image segmentation has been an important area of study in computer vision. Image segmentation involves processing an image, and labeling each pixel according to the class to which it belongs. When there are only two classes, and each pixel is classified either as foreground or background, it is called binary segmentation. On the other hand, if there are more than two classes, it is called semantic segmentation. Different methods have been proposed for image segmentation, which can be broadly classified into two categories: i) Hand-crafted approaches, ii) neural network-based methods.

1.1.1 Hand-crafted Approaches

Hand-crafted approaches typically rely on features, such as edges, extracted from images by various algorithms. This approach involves understanding and abstracting the problem into a function, and then optimizing this function to reach the desired results. These

methods can be classified into threshold-based, edge-based, block-based, graph-based, and active-contour based methods.

The basic idea behind the threshold-based methods [17–22] is to calculate one or more thresholds, and then to compare the pixel values with these thresholds to decide to which class each pixel belongs. Edge-based methods [23–25], on the other hand, rely on edges to perform the image segmentation task. Edges represent discontinuities in pixel values, and can represent object boundaries. In block-based methods [26–28], similarity rules are used to divide images into blocks, and topology operations are employed to finish the segmentation. Graph-based methods [29–33] build an energy function for an image and minimize it to segment an image into foreground and background. The active-contour based approach [34–36] starts with a manually chosen initial contour, which gradually gets closer to an object's boundary and wraps around the object by minimizing an energy function.

1.1.2 Neural Network-based Approaches

With the availability of increasing amounts of computational power with decreasing prices, neural network-driven approaches have become popular in recent years. Many methods have been proposed, including ones involving deep neural networks. These approaches typically need large amounts of training data, which is fed to the neural network architecture to minimize a loss function, and train the model.

Image segmentation can be considered as a coarse-to-fine classification. Thus, almost every classification model can be modified to support the segmentation task. In recent years, many powerful models have been published to perform image segmentation [3–5, 7–13, 37–39]. The common idea is to use an encoder-decoder structure to segment images. Along the encoder path, also called the down path, model shrinks the resolution of the feature map, but also abstracts high-level features from the data. The decoder path, also called up path, is used to recover the resolution and provide an output, which has the same

resolution as the input image. But due to down-sampling, feature map will loose spatial information, which is problematic for up path's recovery. Methods have been proposed to address this [3, 7–10, 12, 37].

1.2 Image Segmentation Applications

Image segmentation has wide-ranging and important applications. With the development of new neural-network architectures, providing highly accurate results, more application scenarios have resulted.

- **Medical Applications:** Image segmentation methods can be used for different medical applications. Radiologists can use image segmentation to segment areas of interest from medical images including X-rays and CT images. Image segmentation has fund use in dermatological applications, wherein unhealthy skin regions are segmented and classified from RGB images. There has been challenges organized for this application as well [40–42]. Segmentation can also be used to segment each frame of a series of medical images of organs to reconstruct the 3D model inside human body.

- **Autonomous Driving:** Image segmentation plays an important role in autonomous driving applications, wherein it is very important to recognize pedestrians and other vehicles, locate lanes and crosswalks, and detect traffic signs, signals, etc. Correctly understanding the scene and traffic is very important for this application. Performing pixel-wise classification via image segmentation can help self-driving cars to fully understand the traffic scenario.

- **Mapping and Traffic Control:** Image segmentation is also used to process aerial images, and segment roads, buildings etc. to provide traffic information as well as

building detailed maps. Combined with the drone technology, people can get real-time traffic reports and use that information to plan their travels and navigation.

- **Entertainment Industry:** In movie industry, actors are extracted from the background, and then background is changed to any desired scene to create astonishing affects. Image segmentation methods can provide a more precise and better segmentation, and also speed up the process.

1.3 Problem Statement

Semantic image segmentation is a challenging problem, since the algorithm needs to fully understand the difference between targets (foreground) and the background, and label each pixel with a class label. In the past, people needed to understand the internal relationships between images which typically required expert knowledge. Deep learning methods, on the other hand, can learn good features by using large number of labeled images or training data. Thus, they do not require hand-crafted features and provide a way to decrease the need for domain expert's knowledge.

In recent years, with the development of various Convolutional Neural Network (CNN)-based architectures [1, 2, 39, 43, 44], image segmentation accuracy has improved significantly. Methods using CNNs perform convolution and down sampling at each layer, which causes loosing spatial information, which is important to estimate the pixel locations. To address this, methods have been proposed [3, 7–10, 12, 37] using skip connections.

There are works using high-low-high resolution subnetworks in series, also known as auto-encoder/decoder type architecture [3]. This kind of architecture uses subnets that are made up from convolutional layers to down-sample and capture abstract features from input images. Then, up-sampling methods, such as bilinear or trans-convolution [45], are used to recover higher resolution. This kind of methods vary in detail, but the general main idea

is recovering high resolution.

When CNNs become deeper and deeper, the gradient vanishing problem arises, limiting the depth. He et al. [1] introduced a deep residual learning framework, ResNet, to address this problem, which uses skip connections performing identity mapping, and features heavy batch normalization. With this approach, they were able to train a neural network with 152 layers. Huang et al. [2] introduced the Dense Convolutional Network (DenseNet), which connects each layer to every other layer in a feed-forward fashion. This approach was useful to alleviate the vanishing-gradient problem, and strengthen feature propagation. It also reduces the number of parameters.

Overall, these strategies involving skip connections try to propagate features to deeper layers. This strategy can also be found in auto-encoder/decoder type of segmentation methods, such as Unet [6], FCN [3], and deeplab-v3+ [8].

In this thesis, the goal is to find a good balance between extracting features through multiple layers and keeping spatial information. This work has been inspired by the work of Sun et al. [15], which was proposed for human pose estimation. In [15], the high-resolution of feature maps is maintained throughout the process instead of the traditional way of recovering them through a decoder. They start from a high-resolution subnetwork as the first stage, and gradually add high-to-low resolution subnetworks, and connect the multi-resolution subnetworks in parallel. In this thesis, in addition to using adding subnetworks in parallel, we will employ skip connections, as described in [2], to propagate features and encourage feature reuse. We propose a new CNN-based architecture, which will henceforth be referred to as Dense Parallel Network (DensePN). DensePN allows not only keeping high-resolution feature maps but also but also feature reuse at deeper layers to solve the image segmentation problem.

CHAPTER 2
IMAGE SEGMENTATION LITERATURE AND BACKGROUND

In this chapter, the history of image segmentation work is reviewed, and the neural network-based image segmentation methods are introduced.

2.1 Background and History

Object recognition work has its origins in the early 1960s [46]. In the beginning, hand-crafted approaches were introduced and became popular. In these approaches, human knowledge is used to understand the pixel relationships in images, and to guide the building of a model to segment images. The relationships between pixels can be in terms of gray-level intensities, colors, textures etc. In different image modalities, these kinds of relationships can become more complicated. For instance, for medical images, the texture and colors of different pixels or areas are highly similar, which makes it harder for the hand-crafted methods to extract features and segment out the target regions. Thanks to their powerful ability of feature extraction, the deep learning based models are becoming more and more popular for the image segmentation task.

2.1.1 Artificial Neural Networks

Artificial neural networks (ANN) aim to mimic the biological neural networks, which are the building blocks of human brain. The ambition to understand natural intelligence encourages people to build intelligent machines [47]. ANNs are composed of artificial neurons, which are modeled to mimic biological neurons. Signals flowing in biological neurons are analog electrical signals, whereas they are digital values for artificial neurons. As seen In Figure 2.1, the output of each artificial neuron is computed by using a function of the sum of the input signals with some added bias. The connections between artificial neurons, i.e. edges, have an adjustable weight, and are used to simulate the learning process.

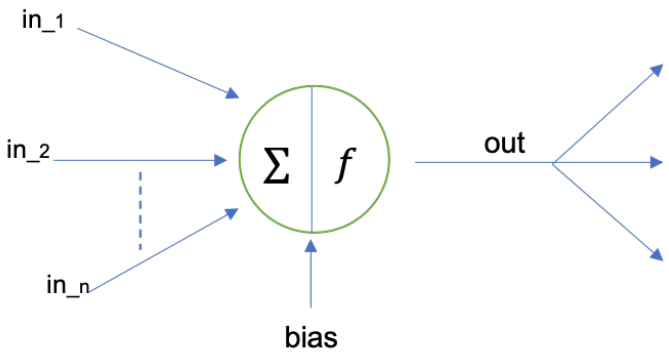

Fig. 2.1: Structure of an artificial neuron

The function mentioned above is called the activation function. There are different choices for the activation function, but there are some more common functions, which have been proven to work well. Such functions include the sigmoid function [48] and ReLU (Rectified Linear Unit) [49]. Since probability of an event is in the range [0,1], sigmoid function is also used to predict the probability in the output of the last layer in classification networks. When the input of the sigmoid function is large, as seen in Figure 2.2, the gradient will be very small, which consequently will influence the learning speed of the model. On the other hand, ReLU is much easier to calculate, and it will not shrink down the gradient when input is large. Therefore, ReLU is one of the most popular activation

functions used in CNNs and deep learning architectures.

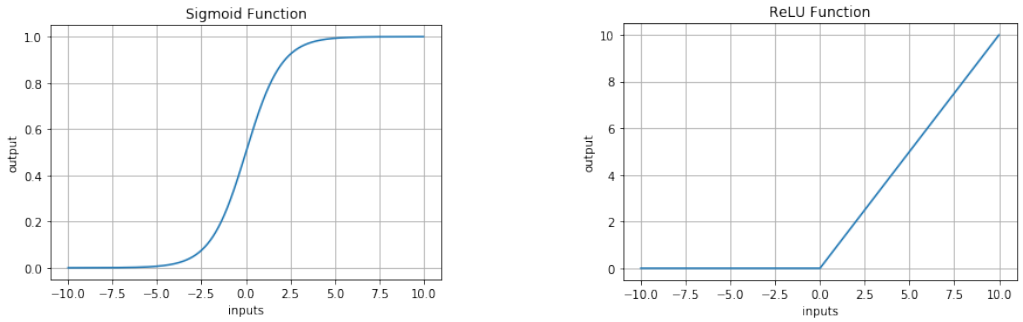

Fig. 2.2: Different activation functions; Sigmoid (on the left) and ReLU (on the right).

A basic architecture for an example ANN is shown in Figure 2.3. It has an input layer, a hidden layer and an output layer. Neurons in the neighboring layers are connected by weighted edges. To use this type of ANN for processing (possibly classifying) gray images, we need to first flatten the two-dimensional images to one dimensional vectors. This flattened vectors are then used as the input to the ANN.

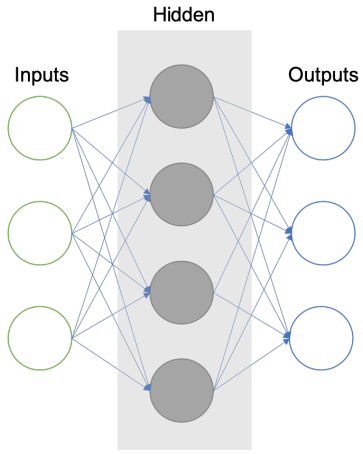

Fig. 2.3: Artificial neural network architecture

2.1.2 Convolutional Neural Networks

Similar to ANNs, Convolutional Neural Networks (CNNs) are also inspired by biological visual cortex structure [50]. CNNs are widely used in deep learning. A CNN includes convolutional layers and fully connected layers. The output of a convolutional layer is called

a feature map, which usually is a 3 dimensional tensor with dimensions width, height and depth. The convolutional layer uses a filter to process inputs or feature maps, similar to the filtering operation in image preprocessing. The main difference is that with traditional image preprocessing, the weights (values) of the filter is hand-crafted, whereas in CNNs, they are learned from data during training. Output of a convolutional layer corresponds to the receptive field in the input feature map and the size of the receptive field is determined by the kernel size and dilation. With CNNs, it is not necessary to flatten the images. However, to feed the features maps, which are the outputs of convolutional layers, to the fully connected layers flattening must be performed. Fully connected layers are the same as ANNs described above, although ANN is the broader name for all artificial neural networks, which include CNNs. Fully connected layers learn the linear and non-linear relationships between extracted features, and classify each sample into two or more abstract classes. Figure 2.4 shows a simple CNN architecture.

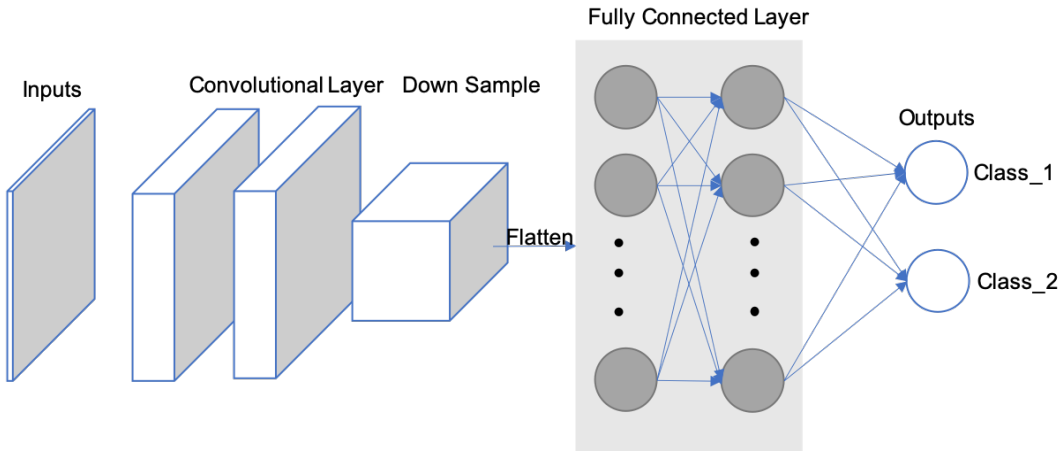

Fig. 2.4: A simple CNN architecture

There are also pooling layers [51] between convolutional layers. They are used to down sample the feature maps and reduce the number of parameters, thus the computational complexity, of the model. Average pooling and max pooling are the two most commonly used types of pooling layers. As their names suggest, average pooling takes the average of the values on the feature map corresponding to a kernel, which slides through the whole fea-

ture map. Max pooling, on the other hand, takes the maximum value in the neighborhood corresponding to the kernel position on the feature map. Figure 2.5 shows the difference between these two pooling types. The pooling layers do not have parameters that need to be learned during training. The stride in the convolutional layer can also be used to replace the pooling layers.

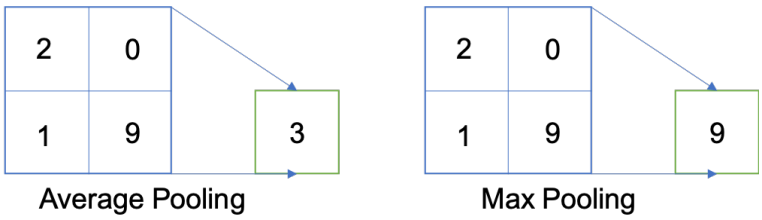

Fig. 2.5: Average Pooling and Max Pooling in a 2x2 neighborhood.

Batch normalization (BN) [52] is another important component of a CNN architecture. Similar to normalizing the inputs images, BN normalizes the feature maps at the output of a convolutional layer. BN subtracts the batch mean and divides the values in the feature map by batch standard deviation so that the distribution of feature maps will be the same. Consequently, BN reduces the value shift in hidden layers. BN provides more freedom on the initialization of kernel weights. In addition, higher learning rates can be used, since BN makes sure that there is no activation diverging to very large values. Furthermore, it also has some regularization effects like a dropout layer, since it adds some noise to the feature maps. Both BN and dropout layers will act differently during training and evaluation.

In deep learning, we want the probability density function of the model $pdf(model)$ to be as close to the true real data distribution $pdf(real)$ as possible. However, we only have limited amount of training data available. Thus, the goal becomes to make $pdf(model)$ as close to $pdf(train)$ as possible, by training on a dataset that represents the real-world application domain. To achieve better representation, a sufficiently large training dataset is needed. When the training dataset is not large enough, there will be a large error between $pdf(real)$ and $pdf(train)$. In other words, even though we achieve to obtain a model pdf close to $pdf(train)$, it will not be able to perform well with $pdf(test)$ which is a sample

Convolutional kernel
Batch Normalization
ReLU
Dropout

Table 2.1: Components of a convolutional layer

of $pdf(real)$. This is called the overfitting problem. When overfitting happens, model will fail to fit future observations. To avoid overfitting, dropout [53] is used, which randomly chooses some neurons and ignores them in the training process. Incoming and outgoing edges to a dropped-out node are also removed. By avoiding to train with all nodes, dropout decreases the problem of overfitting, and also improves the training speed. The components of a classic convolutional layer are shown in Table 2.1.

Most of the trainable parameters are at fully-connected layers in a CNN. Fully connected layers can have outputs as the categories for image classification or as regressions, which represent bounding boxes [54]. We can also use convolutional layer work as fully-connected layers to do the same task in more efficient way [55]. In image segmentation task, the output is a mask with the same resolution as input images instead of a class. We can use 1x1 kernel as the output layer to shrink the depth same as desired classes [3], which will provide the model with pixel-wise classification ability.

Training a CNN involves two steps, namely a forward and a backward step. In forward step, we feed the CNN with input information flow. In each layer, feature maps are calculated with preset weights and bias to get the input for the next layer. In the output layer, a loss value will be calculated by using a loss function on the difference between the calculated output and the ground truth. In backward step, chain rule is used to compute the gradients of loss value for every trainable parameter. Then, these gradients are used to update every parameter for the next iteration. This kind of update is why CNNs can 'learn'. After enough iterations, when the loss value converges to an acceptable small value, the training process can be stopped. The loss function is used to calculate the difference or error between the outputs and ground truth. When loss function is used as the guide,

minimizing loss means minimizing the difference between $pdf(model)$ and $pdf(train)$. The loss function will vary for different applications and goals. Cross-entropy is a very commonly-used loss function, since it can measure the similarity/difference between two distributions.

There are also different optimizer choices to guide how to update the network parameters. Stochastic Gradient Descent (SGD) [56] and Adam [57] are two choices. In general, SGD will provide a better result with a low learning speed. It will take many iterations to reach a good result. SGD maintains a single learning rate, which does not change during training. Adam, on the other hand, is faster to converge, since it uses a vector of learning rates, which are adapted as learning progresses. Keskar and Socher [58] present a hybrid strategy and show that using Adam and then switching to SGD results in better performance, and closes the generalization gap.

2.1.3 Feature Reuse

With the introduction of deeper and deeper neural networks, the general observation has been that increasing the depth of the network also increases the performance [59]. With increasing number of convolutional layers in deeper CNNs, a new problem, called gradient vanishing, emerges.

ResNet [1] and DenseNet [2] address the gradient vanishing problem in different ways. ResNet uses identity connections to pass previous feature map to the next one. Figure 2.6 shows the structure of a residual block. Instead of training stacked convolutional kernels to fit to the desired mapping, let's say $H(x)$, ResNet is designed to make its two-kernel fit to $F(x) = H(x) - x$. The original mapping is recast into $F(x) + x$. They state that the residual mapping, i.e. $F(x)$, is much easier to optimize. Even in the extreme case of the identity being the optimize option, model will lead $F(x)$ to zero. Taking advantage of such structure, networks with over 100 layers could be trained. The ResNet was evaluated [1] on the ImageNet dataset with a depth of up to 152 layers. The authors [1] also presented

an analysis on CIFAR-10 with 100 and 1000 layers.

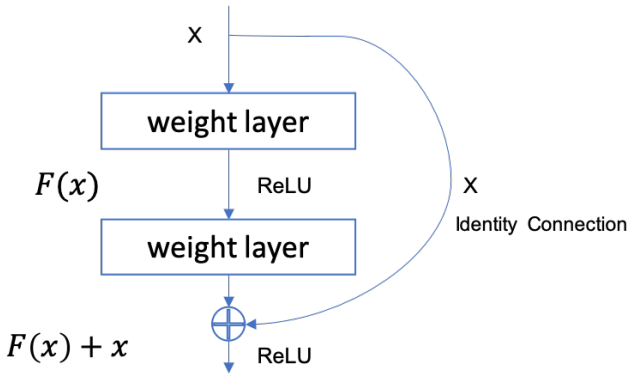

Fig. 2.6: ResNet residual block [1].

Targ et al. [59] introduced a new architecture called ResNet in ResNet (RiR), which is a generalized residual architecture that combines residual networks and standard convolutional networks in parallel residual and non-residual streams.

Xie et al. [60] presented ResNeXt, by adding a new dimension called 'cardinality'. They use a multi-branch approach instead of only one branch as in ResNet. It uses inception [61–63] split-transform-merge strategy on ResNet to achieve higher performance.

DenseNet [2] is different from ResNet [1], which uses skip connections to reuse previous feature maps. Instead, DenseNet connects each layer to every other layer using concatenation. For each layer, the input is all the previous feature maps concatenated together. This is a very straightforward way to reuse features. The input features are accumulated after each convolutional layer, and each convolutional layer outputs a feature map with fixed depth (growth rate). The topology of DenseNet is dense, and the total number of parameters is decreased because of the fixed depth. With the concatenation of all feature maps, it can perform deep supervision like Deeply Supervised Networks [64]. Figure 2.7 shows a basic Dense Block of DenseNet. Here, we did not draw all the feature maps from all the previous layers entering into the concatenate operator. These connections are implicitly represented by the recursive concatenation.

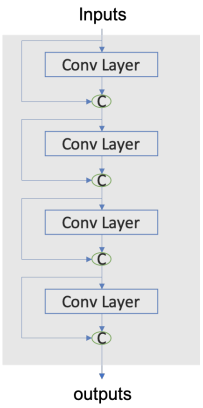

Fig. 2.7: Dense Block [2], the circle with 'C' represents recursive concatenation.

2.2 Related Work

In image classification problem, the goal is to predict the class of the whole image, and thus there is more focus on the abstract feature information from the CNN model, with less emphasis on the spatial information. In image segmentation problem, on the other hand, we care about the abstract information and spatial information at the same time. Abstract information is used to predict which class the input belongs to, and spatial information is used to predict where the corresponding pixel is. As the information flows forward through pooling layers, the spatial information will decrease, and meanwhile, the feature map becomes higher level and more abstract. To perform pixel-wise classification, we need to balance the weights of spatial information and abstract features. Figure 2.8 shows this contradiction between classification and localization.

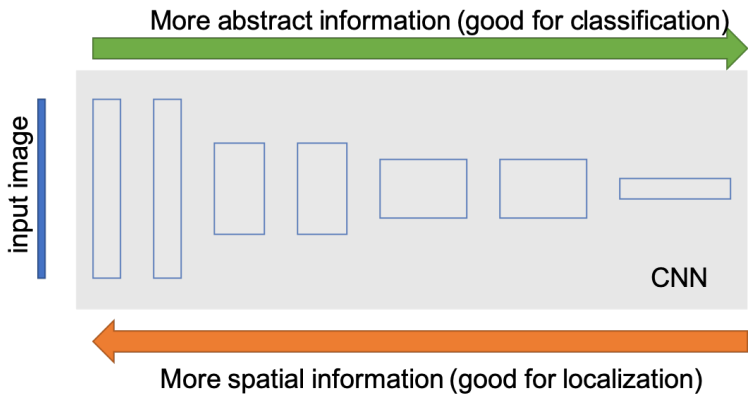

Fig. 2.8: Relation between feature abstraction and spatial information for CNNs.

The most general structure used in the image segmentation models is the encoder-decoder structure. Fully Convolutional Network (FCN) [3] is an extension of CNN, obtained by adding up-sampling layers to normal CNN structure to recover the spatial information (Figure 2.9). It uses skip connections to gradually up-sample the previous features and sums the corresponding low-level features, feeding multi-scale spatial information for recovery. Since FCN uses convolutional layers in the output layer, it does not need independent design for different resolution of images.

U-net [37], which was designed for biomedical image segmentation, is similar to FCN in terms of both down-sample up-sample structure. However, U-net uses concatenation, instead of sum, for skip connection (Figure 2.10).

Chen et al. [8] introduces DeepLabv3+ (Figure 2.11) with atrous superable convolution for semantic image segmentation. They include spatial pyramid pooling to segment targets at different scales, and fuse the low and high resolution features together similar to U-net. D-LinkNet [9] uses residual blocks as the encoder part and uses dilated convolution to ensemble multi-scale features together (Figure 2.12). It iteratively up-samples every scale of feature map and adds them together as the decoder part. The encoder part of SegNet [5] is the same as VGG16 [65] with 13 convolutional layers. In decoder part, SegNet computes pooling indices while doing the down sampling by max pooling (Figure 2.9). Then, it uses these indices to do non-linear up-sampling. DeconvNet [4] uses VGG 16 layers as the

encoder. In the decoder part, it uses deconvolution (transpose convolution) and up-pooling.

Fully Combined Convolutional Network (FCCN) [10,38] modifies FCN, by using multiple steps of up-sampling and combining feature maps in pooling layers with corresponding up-pooling layers (Figure 2.13). The aforementioned methods all use low-resolution feature maps to recover the high-resolution final outputs. All of them have down-sampling/up-sampling structure or encoder/decoder structure, usually with a skip connection as well.

FCDense [7] uses dense blocks instead of convolutional layer, and it has the same topology structure as U-net. Thanks to the dense block's structure, it can significantly reduce the number of parameters.

Unlike previous work, Full resolution Convolutional Networks (FrCN) [11] can learn the full resolution features without down-sampling and skip connections (Figure 2.14). Because of the lack of down-sampling, this method will not perform well in multi-class semantic segmentation.

Li et al [12] introduced Y-Net, a Y-shape network architecture, which contains two-arm feature extraction module (Y_1, Y_2) and a branch for feature fusion (Y_3), as seen in Figure 2.15. Y_1 is similar to U-net, Y_2 is similar to FrCN, and Y_3 is a simple series of convolutional layers. Y_1 is a down-samplin/up-sampling structure and its output is a full resolution feature map. Y_2 is a full-resolution convolution structure. Both branches' outputs are concatenated together and fed through Y_3 to provide the final output.

RCNN and its derivatives [39, 66, 67] employ a region-based method for image segmentation. In the beginning, region-based CNN was proposed to solve the object detection problem by predicting the location of bounding boxes of different objects in images. In RCNN, a region of interest (RoI) is calculated, and the class of RoI is predicted. Faster RCNN uses a CNN region proposal sub-network to create RoI, which is not precise enough for image segmentation. Mask RCNN modifies the RoI pooling of faster RCNN. In addition, mask RCNN adds another sub-network to build the mask for the image segmentation task. Comparing Faster RCNN and Mask RCNN, we can observe that they share the same

encoder part, but have different decoders. Moreover, from FCN, U-net, DeepLabv3+ and D-LinkNet (Figures 2.9, 2.10, 2.11, 2.12), we can see that the encoder part topologies are highly similar, and all constructed by series of convolutional layers.

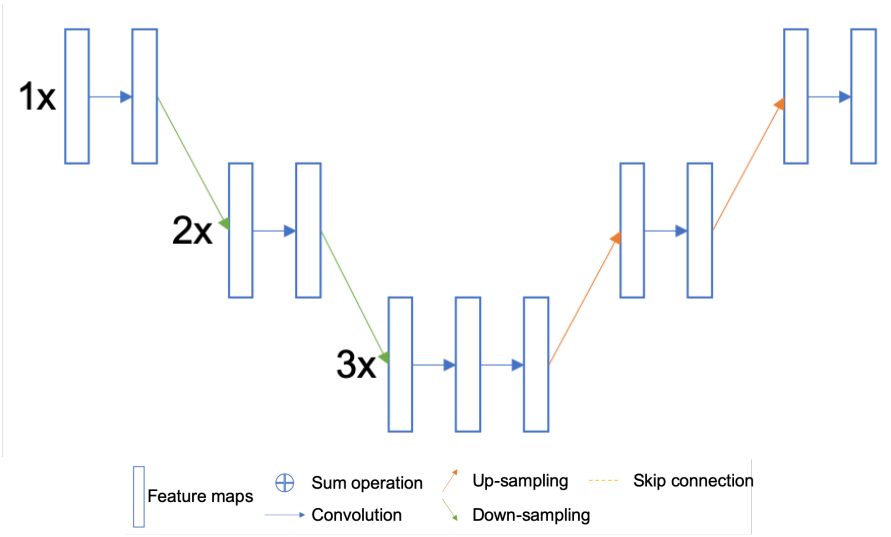

Fig. 2.9: Basics of FCN [3], Deconv [4], and SegNet [5] network architectures.

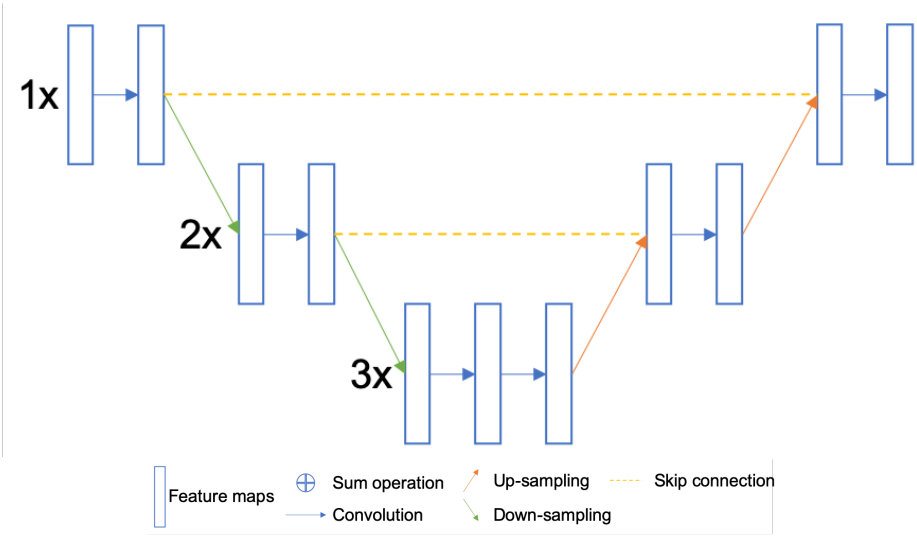

Fig. 2.10: U-net [6] and FCDenseNet [7] main network architectures

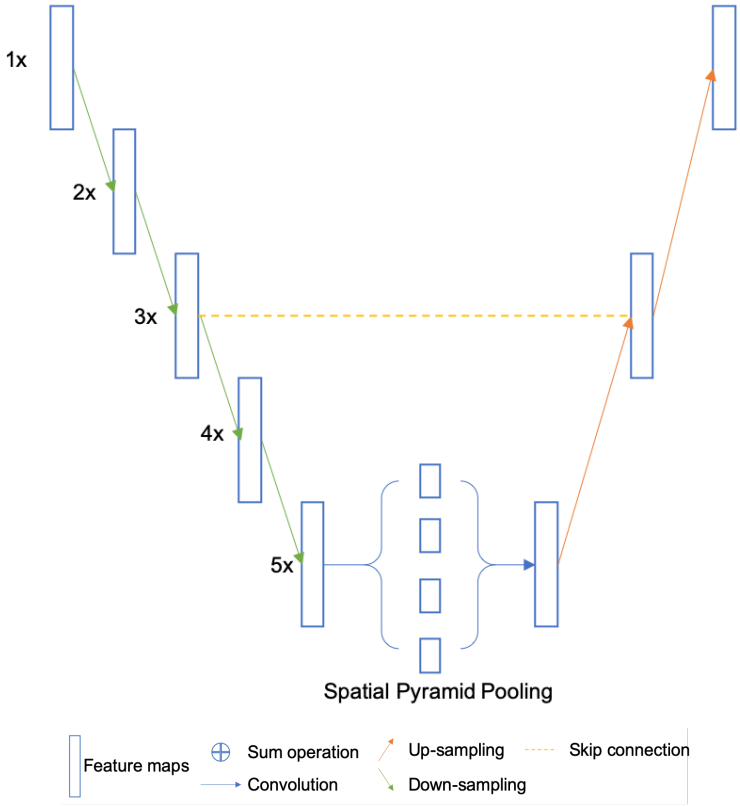

Fig. 2.11: DeepLab-v3p [8] network architecture

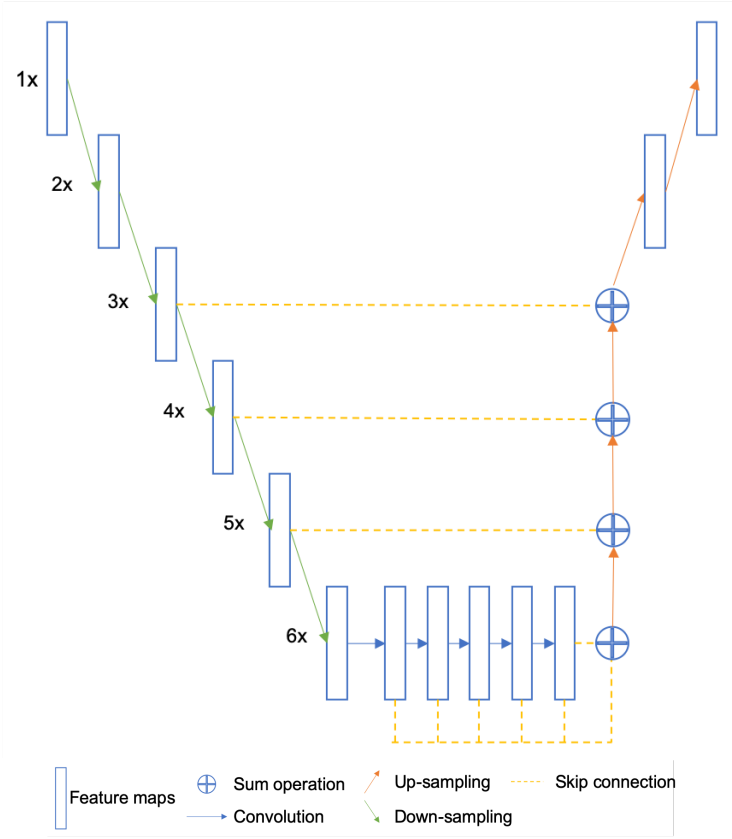

Fig. 2.12: D-LinkNet [9] network architecture

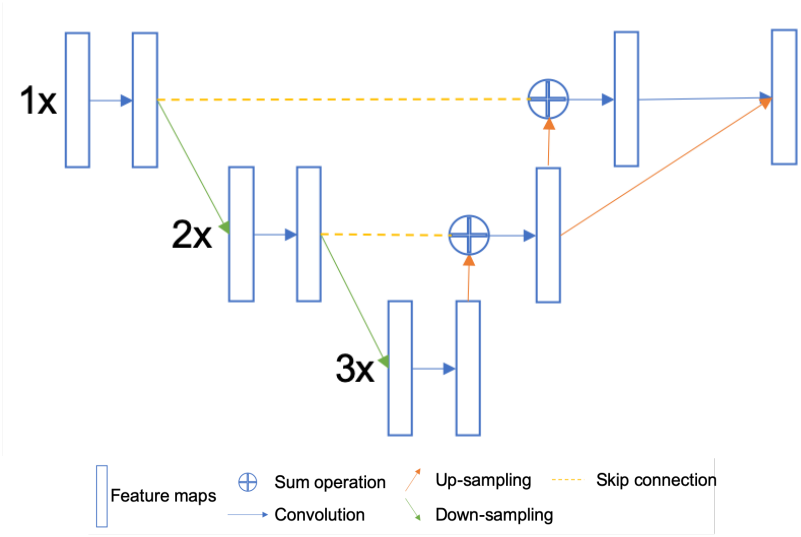

Fig. 2.13: FCCN [10] network architecture

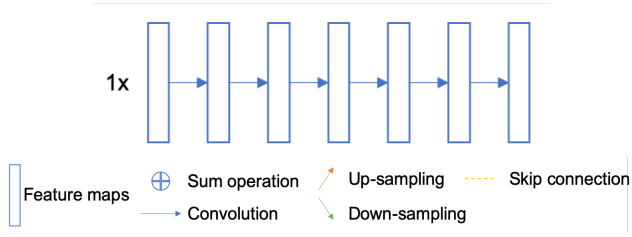

Fig. 2.14: FrCN [11] network architecture

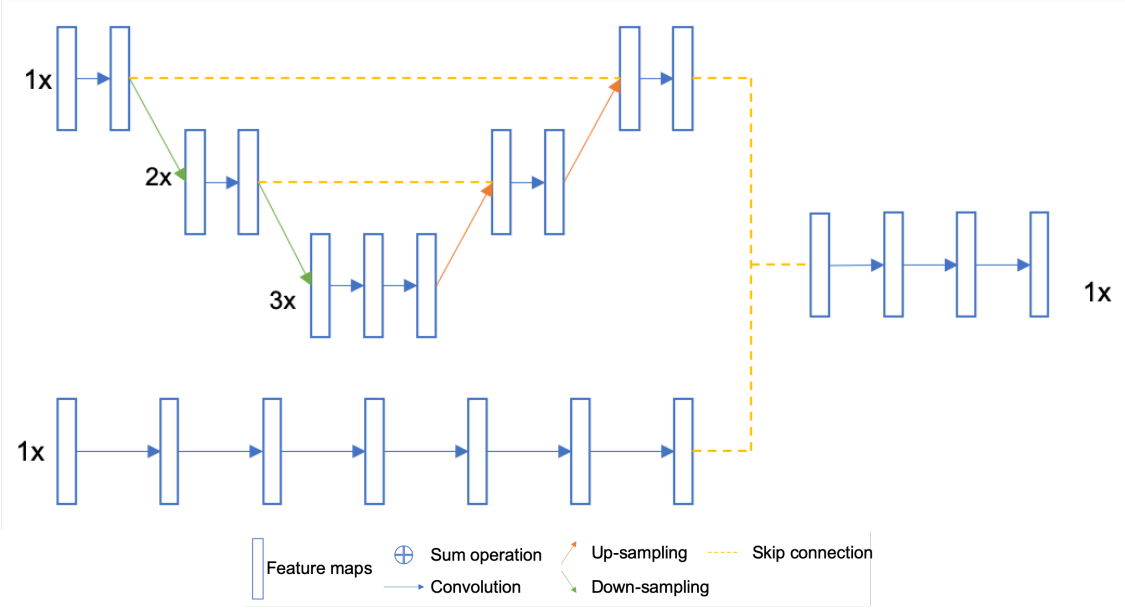

Fig. 2.15: Y-Net [12] network architecture

Architectures in Figures 2.9, 2.10, 2.11, 2.12, 2.13, 2.14, 2.15 all use series of convolutional layers, some with skip connections some not.

There are also parallel information flow architectures for image segmentation. UNet++ [13], shown in Figure 2.16, is a nested version of the original UNet, in which encoder and decoder sub-networks are connected through the dense skip connections. The dense skip connection path fills the gap between the encoder and decoder. Convolutional neural fabrics [14], seen in Figure 2.17, connect multi-scale convolutional layers and a sparse homogeneous local connectivity pattern. The ensemble network can fuse all features together, and provides a prediction. High-Resolution Net (HRNet) [15], shown in Figure 2.18, starts with a full resolution layer and gradually adds high-to-low resolution sub-networks one by

one to form more stages and to connect multi-scale feature maps in parallel. Interlinked CNN [68] consists of multi-scale CNNs and uses and interlink layer to allow the CNNs exchange information. GridNet [69] follows a grid pattern allowing multiple interconnected streams to work at different resolutions. Multi-scale DenseNet [16], shown in Figure 2.19, uses multi-scale convolutions and progressively generates new feature maps used in following layers.

Our proposed approach (Dense Parallel Network), which is introduced in Section 4.2, is different from the DenseNet in the following ways: (i) DenseNet has been proposed for the object recognition task; and (ii) DenseNet structure does not have an up-sampling path, and cannot keep full resolution feature maps. Figures 2.16, 2.17, and 2.18 show the topology of parallel flow networks.

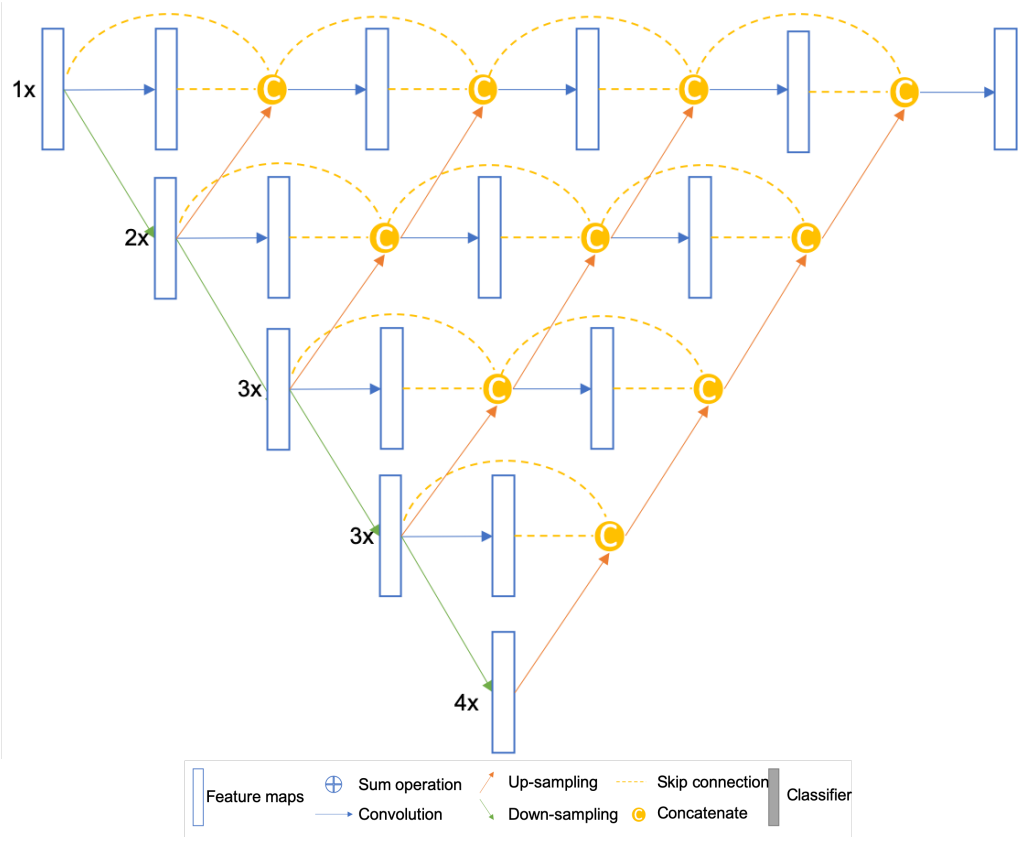

Fig. 2.16: Unet++ [13] network architecture

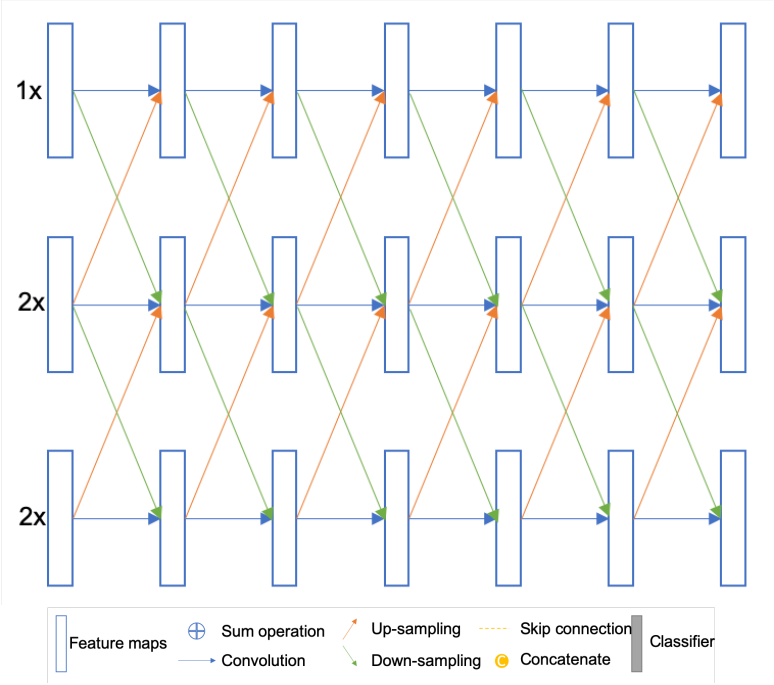

Fig. 2.17: Convolutional neural fabric [14] network architecture

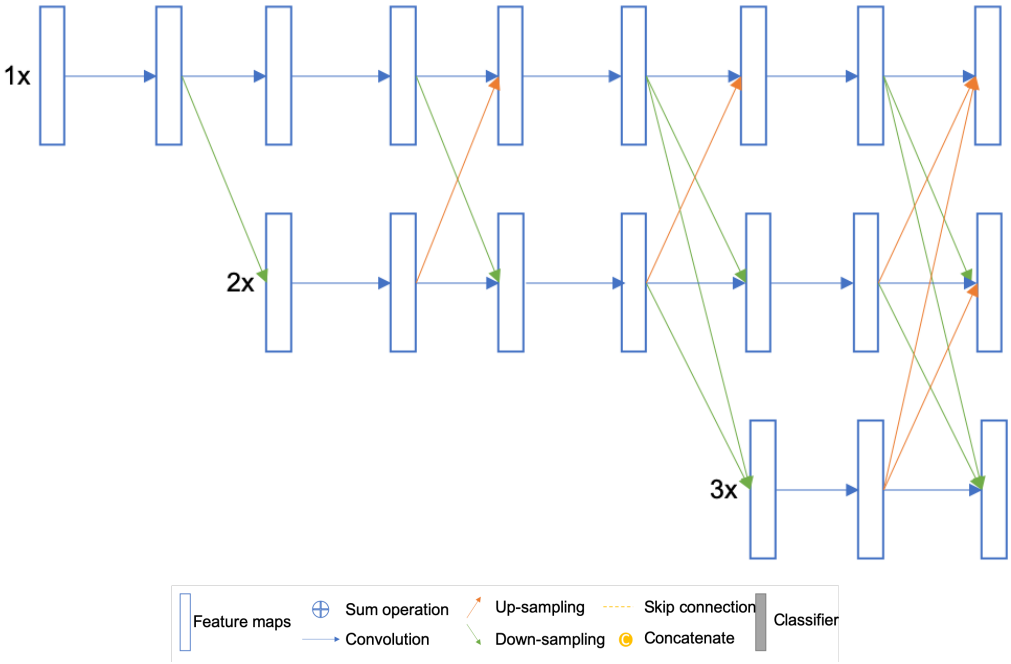

Fig. 2.18: HRNet [15] architecture

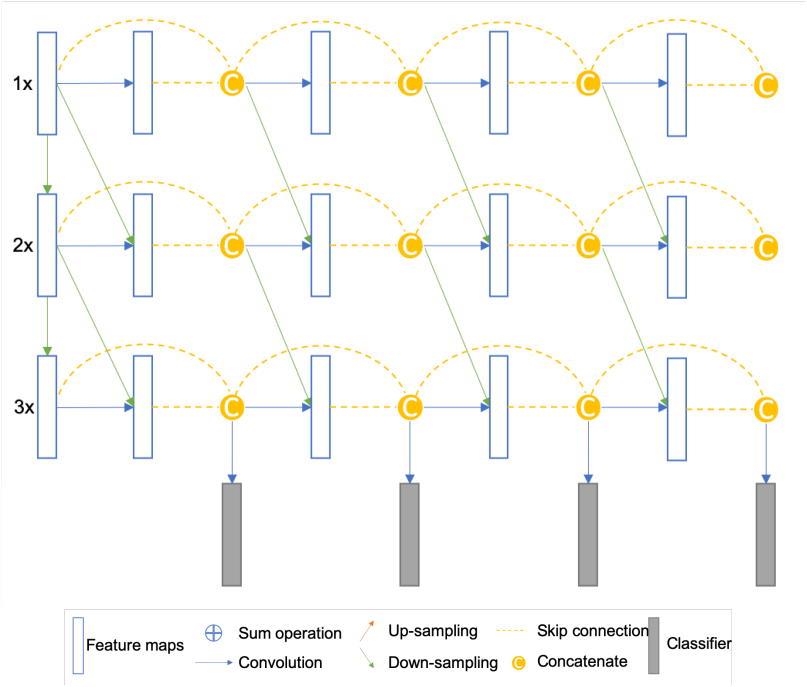

Fig. 2.19: Multi-scale DenseNet [16] architecture

We have designed the Dense Parallel Network based on three main observations that we have gained from our initial trials and preliminary studies. First, maintaining a high-resolution feature map provides good performance. Second, feature reuse is very efficient, and allows having deeper networks. Third, having a parallel structure can provide better information flow. We have obtained very promising results presented in Section 4.2.

CHAPTER 3
UNET-LAPLACIAN

3.1 Overview

In this chapter, we present the network architecture that we developed based on U-net [37]. As briefly mentioned above, U-net reuses the previous high-resolution maps by concatenating them to Up path to enhance the spatial information, which is very important for the decoder part to do the segmentation. In our network, we incorporate edge features/information to this network. The idea is that the edge information is useful for the decoder just like the previous high-resolution features.

Laplacian filter has been used to enhance edge information and to detect edges. In our network, which will henceforth be referred to as Unet-Laplacian, we add a new Laplacian branch. This new branch performs Laplacian filter operation on the input RGB image, and uses multi-scale max pooling to make the resolution match the corresponding decoder features. The Laplacian filter uses a constant weight convolution kernel for implementation shown in Figure 3.1. Experimental results show that Unet-Laplacian provides slight improvement over the original U-net. The output of the Unet-Lapl closely follows that of the original U-net, but the output of Unet-Lapl captures more of the ground truth mask.

0	-1	0
-1	4	-1
0	-1	0

Fig. 3.1: The Laplacian filter used in the proposed Unet-Laplacian.

3.1.1 Model Details

The architecture of the Unet-Laplacian is shown in Figure 3.2. We use an extra, almost computationally free, branch called Laplacian branch to enhance the edge information compared to U-net. The architecture can be divided into four parts: Down path, Bottleneck Conv, Up path and Laplacian path. The details of each block we used in Unet-Laplacian are provided in Table 3.1.

Conv Block
3x3 same convolution + ReLu
3x3 same convolution + ReLu

(a) Conv Block

Down Block
2x2 MaxPool
Conv Block

(b) Down Block

Up Block
2x2 Up-sampling nearest
3x3 same convolution + ReLu

(c) UpSample

Table 3.1: Blocks of Unet-L

The Down path includes 4 Down blocks. We use two 3x3 same convolution layer followed by ReLU as a basic Convolution block. The Down blocks include a 2x2 MaxPool layer followed by basic Conv blocks. Up path includes 4 Up blocks.

The up-sample layer includes a nearest up-sampling layer followed by a 2x2 convolutional layer and ReLU. A single Up block includes an up-sample layer. Every concatenation operation combines 3 different features together along depth dimension. The output is given by a 1x1 convolutional layer followed by a soft-max layer. For given output, if the probability is larger than 0.5, we classify it as an object and the rest as background.

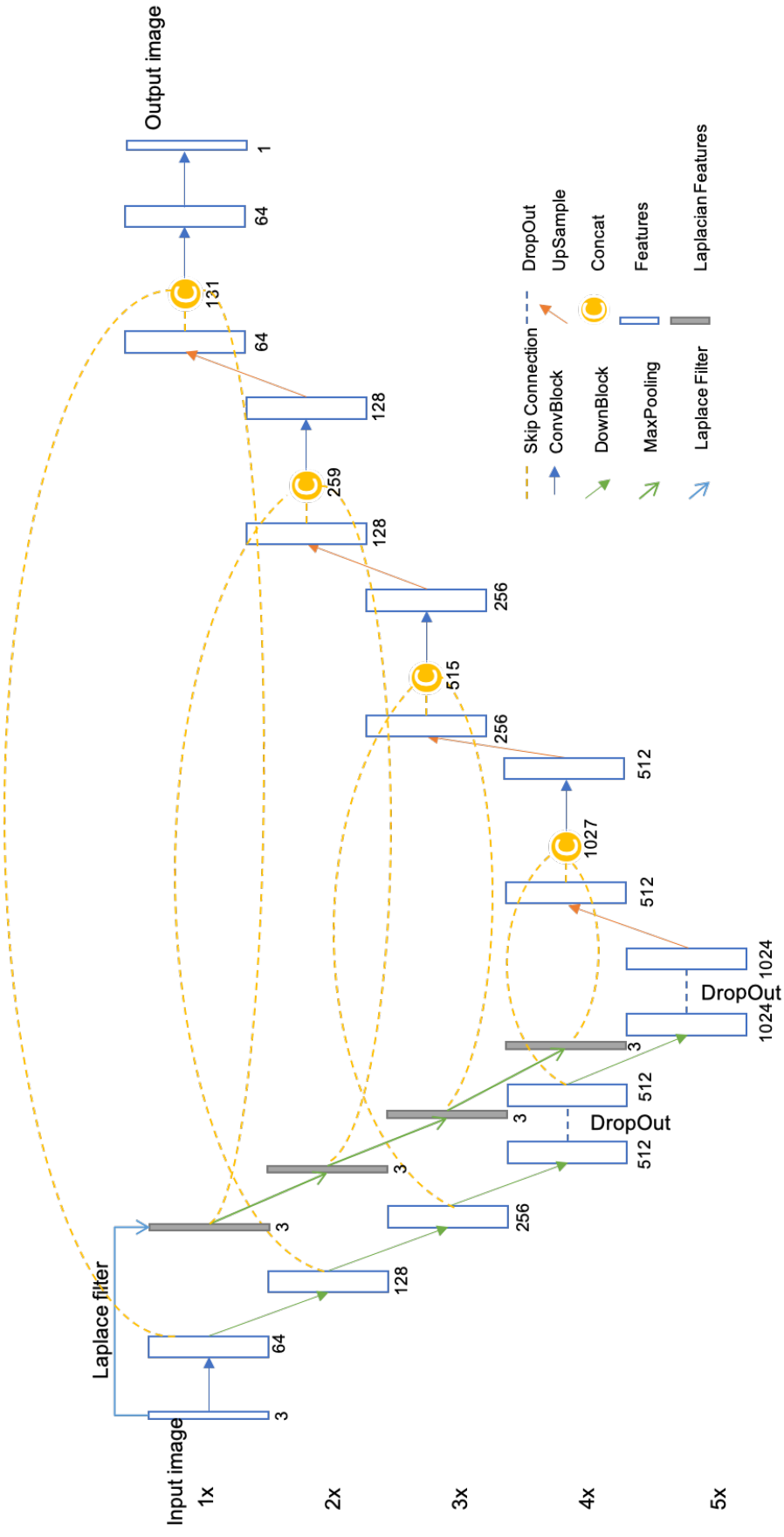

Fig. 3.2: Unet-Laplacian architecture, the leftmost side of figure shows the resolution scale of feature maps. 1x means full resolution. 2x scale is getting from down-sampling the 1x scale. The little number below features response to depth of the feature map. This Unet-L architecture is based on Unet architecture which is already discussed in Section 2.2, Network architecture figures in Chapter 2 is basic concept representation, which lose details in some parts. Figures in this Chapter will contain more details.

The Laplace path is a simple one that uses a MaxPool layer, to sample the inputs to proper size, followed by a Laplacian filter. The summary of Unet-Laplacian is shown in Table 3.2. Since the Laplacian filter is only a constant convolutional kernel, it doe not add much to the computational budget.

Main Branch
Input, c=3
Conv Block 1, c=64
Down Block 2, c=128
Down Block 3, c=256
Down Block 4, c=512
Dropout,p(0.5), c=512
Up Block 4, c=256
Concat(Up4, Lap4, Down4), c=1027
Conv Block, c=512
Up Block 3, c=256
Concat(Up3, Lap3, Down3), c=515
Conv Block, c=256
Up Block 2, c=128
Concat(Up2, Lap2, Down2), c=259
Conv Block, c=128
Up Block 1, c=64
Concat(Up1, Lap1, Down1), c=121
Conv Block, c=64
3x3 convolution layer + ReLu, c=2
Softmax, c=1

(a) Architecture of the main branch. c means num of channels. Concat layer describe the concatenate rule.

Laplace Branch	
Input, c=3	
–	3x3 Laplace1 c=3
2x2 Maxpool, c=3	3x3 Laplace2 c=3
4x4 Maxpool, c=3	3x3 Laplace3 c=3
8x8 Maxpool, c=3	3x3 Laplace4 c=3

(b) Architecture of the Laplace Branch, c means num of channels. Laplace 1 get from Input. Laplace 2, 3 and 4 get from correspoding maxpool layer. All maxpool layers connect to Input layer

Table 3.2: Architecture of Unet-L, there are two branches, main branch and laplace branch. Check Figure 3.2

3.2 Experiments

The cost of building annotated image segmentation datasets is very high, since the ground truth needs pixel wise annotation. Especially in a special purpose area, such as biomedical imaging, the requirement for expert knowledge makes it even harder to build and maintain large annotated datasets.

We have trained and evaluated our proposed Unet-Laplacian and compared it with U-net on ISIC and IDRiD datasets. Below, we will first describe these two datasets.

3.2.1 ISIC Skin Cancer Dataset

Regular image datasets do not rely on expert's knowledge too much. They can use a web server asking people with common knowledge to help annotate/segment the images. However, for medical images, the ground truth mask can only be provided by experts who have worked on or have expertise with a specific disease. In addition, the protection of patient privacy, results in limited number of medical images.

Skin cancer is the most common malignancy diagnosed in United States [70]. Melanoma, a kind of skin disease, develops from the pigment-containing cells. An estimated number of 96,480 new cases of melanoma and 7,230 deaths due to melanoma have been reported in the United States in 2019 [70]. A good segmentation method predicting the boundary of skin lesion can help dermatologists make early diagnosis.

The International Skin Imaging Collaboration (ISIC) is an international effort to improve melanoma diagnosis. The ISIC2018: Skin Lesion Analysis Towards Melanoma Detection [41, 71] Task1: Lesion Boundary Segmentation dataset includes 2594 images of skin lesion and response masks for training. It also includes 100 pairs of images for validation and 1000 images for testing. Since pigment-containing cells occur on the surface of the skin, all images in ISIC2018 dataset were captured by dermoscopy, which is an image technology that eliminates the surface reflection of skin, which will help to improve the

visualization of skin lesion samples. Figure 3.3 shows example images together with the ground truth masks from the dataset.

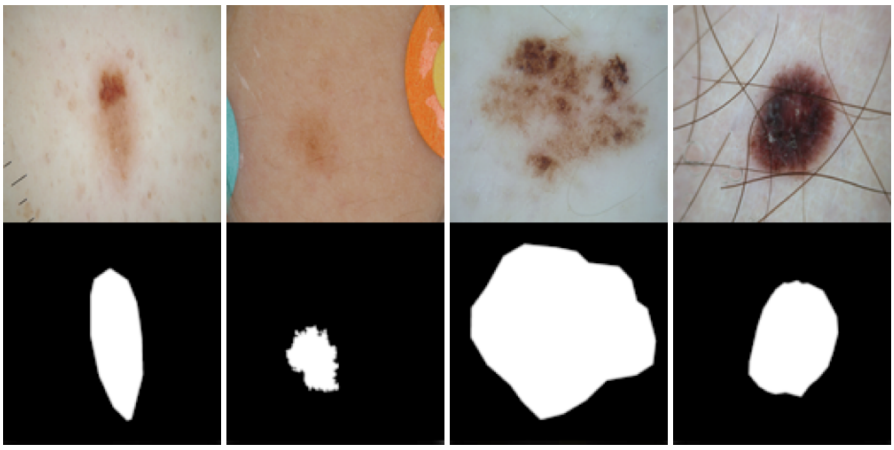

Fig. 3.3: Example images from the ISIC2018 dataset. Top row: input RGB images; bottom row: the ground truth mask.

3.2.2 IDRiD Diabetic Retinopathy Images

IDRiD dataset [42] provides data with the aim of segmenting the retinal lesions from diabetic retinopathy images. There is a total of 81 RGB images with pixel-wise ground truth. We randomly split the dataset, and use 54 images for training and 27 for validation. The images are pixel-wise annotated for optic-disk (OD), microaneurysms (MA), soft exudates (SE), hard exudates (EX) and hemorrhages (HE). The corresponding binary mask is individually provided for each class. Figure 3.4 shows example images from the IDRid dataset.

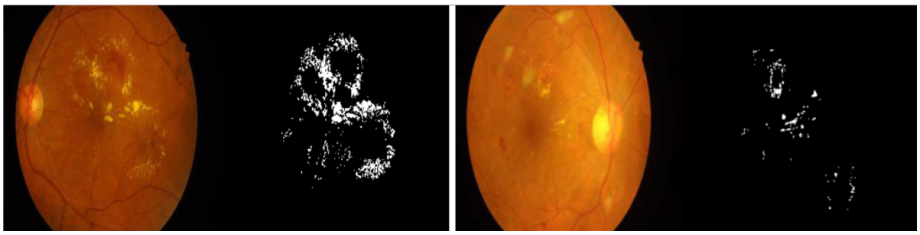

Fig. 3.4: Exampleimages from the IDRiD dataset, together with the ground truth for hard exudates.

3.2.3 Training

We have trained and evaluated our proposed Unet-Laplacian and compared it with U-net on the ISIC and IDRiD datasets described above. Both models were trained from scratch without any extra data or pre-training process. We evaluated the results by using Intersection over Union (IoU) and pixel-wise accuracy. IoU is described by using the test statistics shown in Table 3.3. $TP(c_i)$, $FP(c_i)$, $FN(c_i)$ and $TN(c_i)$ stand for true positive, false positive, false negative and true negative, respectively, for class c_i.

		GT	
		Present	Absent
P	positive	true positive	false positive
	negative	false negative	true negative

Table 3.3: Test statistics terms for class c. GT stands for ground truth, P stands for Prediction.

Let $IoU(c)$ denote the IoU for class c. Mean IoU, $mIoU$, is the average of IoU for all classes. Equations below described how to calculate IoU and $mIoU$.

We can describe IoU of class c using $IoU(c)$. Mean IoU i.e. $mIoU$ is the average of IoU along class. Equations (3.1) and (3.2) show how IoU and $mIoU$ are calculated, respectively.

$$IoU(c_i) = \frac{TP(c_i)}{TP(c_i) + FP(c_i) + FN(c_i)} \qquad i = 1...|c| \tag{3.1}$$

$$mIoU = \frac{\sum_1^{|c|} IoU(c_i)}{|c|} \tag{3.2}$$

We implemented original U-net and compared it with our proposed Unet-Laplacian model. Since the original U-net does not apply batch normalization, we also implemented U-net with batch normalization (Unet-bn) and also Unet-Laplacian with batch normalization (Unet-Lapl-bn).

Since masks of test images are unavailable for the ISIC dataset, we have compared the results on the validation data. We initialized the Unet-Laplacian model by xavier normal and using Adam [57] optimizer with constant learning rate of 1e-4. The drop rate of Dropout layer was set as 0.5. We monitored IoU on validation split every 5 epochs and saved the best (highest) one. For all experiments, we used the same training rules and same dataset split. We resized the input RGB images to 256x256 resolution. We also applied random vertical and horizontal flip for training data augmentation. The Laplacian filter with kernel size 3 is applied on R, G, B channels of the input image. The loss function we used with U-net and Unet-Laplacian is $Cross\ enctropy + 1 - iou$.

3.2.4 Results and Discussion

As mentioned above, since masks of test images are unavailable for the ISIC dataset, we have compared the results on the validation data. Table 3.4 shows the results. As can be seen, the proposed Unet-Lapl provides some improvement over the original U-net. Compared to original U-net, Unet-Lapl increases the IoU by 0.26%. We can also see that for both U-net and Unet-Lapl, using batch normalization provides improvement.

Model	IoU
Unet	0.80827
Unet-bn	0.82393
Unet-Lapl	**0.81037**
Unet-Lapl-bn	**0.82399**

Table 3.4: Results of Unet-Laplacian model on the ISIC dataset.

Since better results were obtained with batch normalization on the ISIC dataset, we only trained the batch normalization version of both models on the IDRiD dataset. Table 3.5 shows the results on the IDRiD dataset. Except for the class of SoftExudates, the proposed Unet-Lapl increases the IoU for all classes. But due to having very limited data in this

dataset (only 54 for training and 27 for validation), Microaneurysms are barely detected, and we cannot say that the results on this dataset are convincing.

Model	IoU					mIoU
	Optic Disc	Hard Exudates	Microaneurysms	Haemorrhages	SoftExudates	
Unet-bn	0.93146	0.35268	0.02143	0.27916	**0.45861**	0.40866
Unet-Lap-bn	**0.93271**	**0.38351**	**0.03064**	**0.30494**	0.42585	**0.41552**

Table 3.5: Results of Unet-Laplacian model on IDRiD dataset.

Experimental results show that Unet-Laplacian provides slight improvement over the original U-net. Some example outputs on the ISIC dataset and the IDRiD Hard Exudates dataset are presented in Figures 3.5 and 3.6, respectively. As can be seen, the output of the Unet-Lapl closely follows that of the original U-net, but the output of Unet-Lapl captures more of the ground truth mask. From the results, we can conclude that the Laplacian filter carries input edge information to the decoder to help model to predict a better segmentation mask.

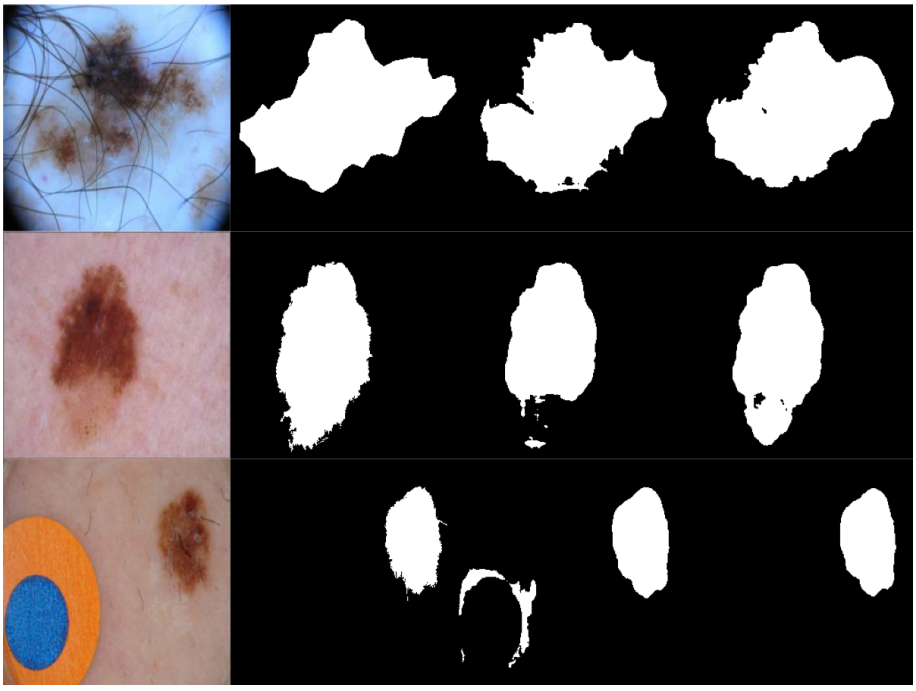

Fig. 3.5: The results of ISIC dataset, from left to right are original input, ground truth, Unet-bn, Unet-Lap-bn respectively

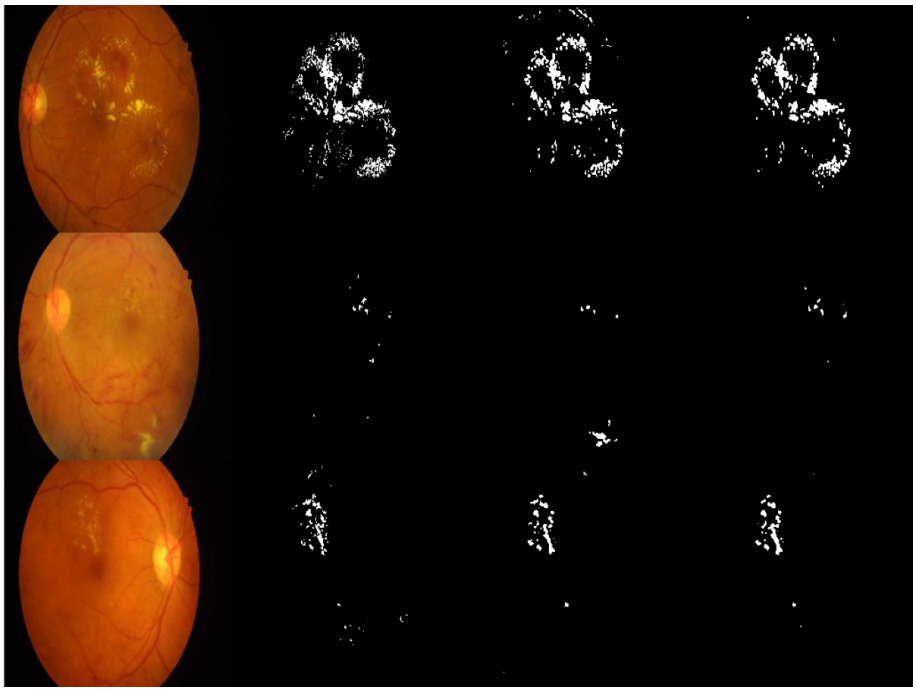

Fig. 3.6: The results of IDRiD Hard Exudates dataset, from left to right are original input, ground truth, Unet-bn, Unet-Lap-bn respectively

CHAPTER 4
DENSE PARALLEL NETWORK

4.1 Overview

In this chapter, we will describe our proposed Dense Parallel Network (DensePN) for image segmentation. DensePN combines ideas from feature reuse architectures [1–3, 7], HRNet [15] and Multi-scale DenseNet [16]. As we concluded at the end of Section 2.2, feature reuse and parallel convolution are the main ideas that guided us through the design of the DensePN. HRNet maintains the full resolution throughout the whole process without the need of recovering the high resolution from low resolution. In addition, fusing multi-scale features helps the model to get a reliable high-resolution representation.

In DensePN, in addition to using adding subnetworks in parallel, we employ skip connections to propagate features and encourage feature reuse. DensePN allows not only keeping high-resolution feature maps but also feature reuse at deeper layers to solve the image segmentation problem. We designed the DensePN to reuse features, and perform parallel convolution, for image segmentation task, while being parameter efficient at the same time. DensePN has a similar structure as multi-scale DenseNet [16], but there are two main differences between them. First, multi-scale DenseNet does growth convolution then concatenates the multi-scale features. However, DensePN involves multi-scale features into

growth convolution. Second, multi-scale DenseNet is designed for image classification, and does not contain the up-sample path.

In this chapter, we compare the DensePN with FCDenseNet [7], since it is the best performing network with the smallest parameter size. FCDenseNet uses dense blocks instead of common convolutional layers to build a U-shape Net. Thanks to the power of DenseNet, FCDenseNet has very few parameters for a relatively deep Network (1.5M parameters for FC-DenseNet56 (56 filters) and 9.4M parameters for FC-DenseNet103 (103 filters)). Experimental results have shown that our proposed DensePN provides better $mIoU$ than FC-Dense56 under the same split conditions on the CamVid dataset. In addition, our proposed DensePN has only 1.1M parameters, compared to the 1.5 M parameters of FCDense56. This corresponds to 26.6% reduction in the number of the parameters.

4.1.1 Review of DenseNet, HRNet and FCDenseNet

DenseNet

Let x_l represent the output of l^{th} layer, and let H_l represent a non-linear transformation at l^{th} layer. In a common CNN layer, the input of H_l is the output x_l of the previous layer. In other words, we can describe the output of l^{th} layer as $x_l = H_l(x_{l-1})$. In traditional CNNs, the non-linear transformation $H_l(x_{l-1})$ is defined as $H_l(x_{l-1}) = Bn(Drop(ReLU(w_l x_{l-1} + b_l)))$, where $Bn()$ denotes batch normalization [52] operation, $Drop()$ is the dropout [53] operation and $ReLU()$ is the ReLU [49] activation function.

In order to overcome the vanishing gradient problem [72], happening with very deep networks, ResNet [1] introduced a residual block, which adds the identity connection to the output of a layer. This structure can be represented as $x_l = H_l(x_l l - 1) + x_{l-1}$. This structure allows feature reuse, and the gradient can flow through from beginning to end.

DenseNet [2] recursively concatenates all previous features to current features, which, in each dense block, can be represented as $x_l = H_l([x_{l-1}, x_{l-2}, ..., x_0])$. Each layer l in a dense block has k output features by growth convolution and the k named growth rate.

This connection pattern strongly enhances feature reuse, and layers all under a directly deep supervision [64]. Because of the relatively small k ($k = 16$ in our experiments), the dense block is more parameter efficient than ResNet and traditional CNNs.

HRNet

A series CNN model usually processes the multi scale features in series rather than reusing them by skip connections [3, 5, 6, 9, 11, 65]. In a series pattern, information flows through all layers in current scale then flows to the next scale. This means that multi-scales features are only fused at skip connection at the decoder part.

HRNet, on the other hand, is a parallel model. It uses a feature exchange layer to fuse multi-scales features and feeds it back to following multi-scale layers. Its architecture is shown in Figure 2.18. Different from HRNet, our DensePN fuses the multi-scales features by skip connections used in DenseNet. The multi-scales features are fused by growth convolution.

FCDenseNet

FCDenseNet is built from dense blocks used in DenseNet. Its network structure is similar to U-net. The difference is that FCDenseNet uses dense blocks instead of the traditional convolutional layers used in U-net. Considering its small number of parameters, FCDenseNet is currently the-state-of-the-art architecture on the CamVid dataset (which will be described below). In the original paper [7], the authors proposed different versions of FCDenseNet, namely FCDense56, FCDense67 and FCDense103. These versions have 1.5M, 3.5M and 9.4M parameters, respectively.

In [7], they first used low resolution images with a relatively large learning rate. Next, they retrained models with smaller learning rate with high resolution to refine the results. In our study, we perform comparison with FCDense56 and train both networks with low resolution images and compare the results. It is worth to mention that after fine tuning,

the small model FCDense56 already outperforms popular architectures, which have at least 100 times more parameters. Our proposed DensePN only has 73.4% of the parameters of FCDense56. Models for the segmentation task and their corresponding number of parameters are shown in Table4.1.

Model	SegNet	Bayesian SegNet	DeconvNet	FCN8	DeepLab-LFOV	Unet	FCDense56	FCDense67	FCDense103	DensePN
Parameters	29.5M	29.5M	252M	134.5M	37.3M	28.9M	1.5M	3.5M	9.4M	1.1M

Table 4.1: Parameters of image segmentation models

4.2 Model Details of the Proposed DensePN

The details of our proposed DensePN architecture are shown in Figure 4.1. Table4.2 lists details of the each block shown in Figure 4.1. The down stage contains 3 down Blocks. We set the output of first conv block as 32. The growth rate used in DensePN is 16. Each row can be seen as a Dense Block in DenseNet [2]. In forward process, the information flows through the down stage and then recursively follows the up path to the output layer.

Conv Block
3x3 same convolution
Batch Normalization
ReLU
Dropout

(a) Conv Block, the basic convolutional block

Down Block
3x3 stride=2 convolution
Batch Normalization
ReLU

(b) Down Block, using kernel size 3x3 and stride equal to 2 instead of MaxPool to do the downsample

Up Block
2x2 Up-sampling nearest
3x3 same convolution
Batch Normalization

(c) UpSample, using nearest up-sample method followed by a 3x3 convolutional layer to up-sample features

First Conv
3x3 same convolution
Batch Normalization
ReLU

(d) First Conv, extend the input feature in depth and as the stem of the network

Table 4.2: Blocks of DensePN

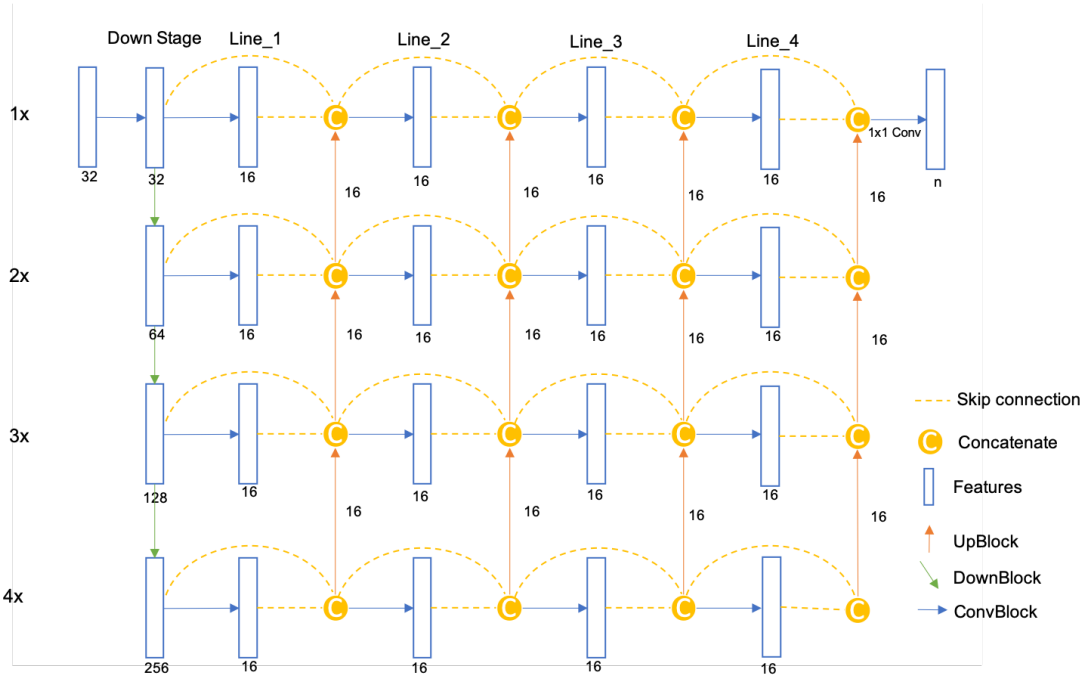

Fig. 4.1: Dense-Parallel-Network architecture. The most left side shows the resolution scales of feature maps. 1x scale: full resolution. 2x scale: half the height and width. The little number below features or at right side of up path arrows are depth of feature map

We defined the traditional convolutional blocks used in DensePN. It contains a 3x3 convolutional block with padding equal to 1, followed by batch normalization, a ReLU activation function and a dropout operation. Each common convolutional block outputs 16 channel features (growth rate equal to 16 i.e k=16). The down block is composed of a 3x3 convolutional layer with stride 2 instead of pooling layer to do down sampling, batch normalization is followed by ReLU. The up path is composed of 2x2 nearest up-sample, 3x3 convolution with padding 1 and batch normalization. The first conv is used as a stem and it is composed of 3x3 convolution with padding 1, batch normalization and ReLU.

4.3 Experiments

We have trained and evaluated the FCDense56 and our proposed DensePN on the Cambridge-driving Labeled Video Database (CamVid) [73, 74]. It is the first collection of videos of a city scene with object class semantic segmentation masks. CamVid dataset includes 11

different classes including ‘building’, ‘tree’, ‘sky’, ‘car’ and ‘pedestrian’. Figure 4.2 shows example images from the CamVid Dataset.

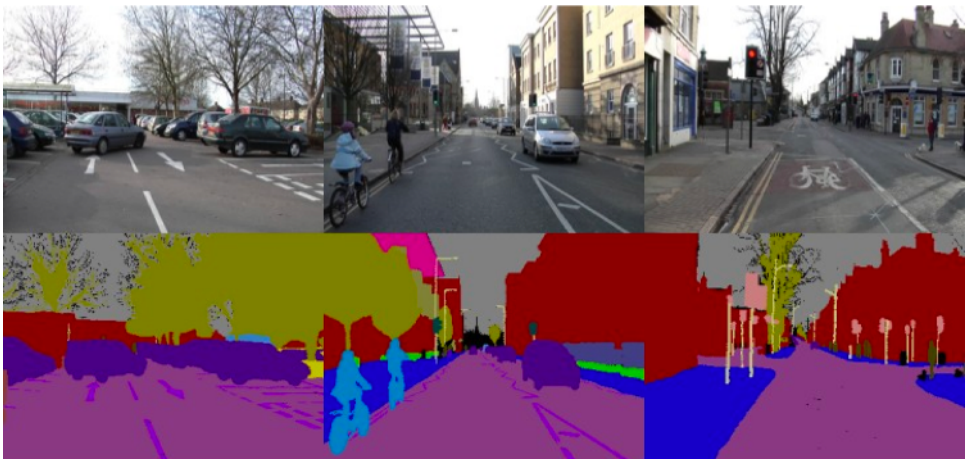

Fig. 4.2: Example images from the CamVid dataset.

We used the same split of the dataset as in [5] (same as FCDenseNet [7]). More specifically, we used 367 images for training, 101 images for validation and 233 images for testing. We resized the input images to 256x256 resolution and used vertical and horizontal flips for data augmentation. We evaluated the results by using Intersection over Union (IoU) and mIoU along classes, and monitored the variation of mIoU to save the weights for testing.

We trained the FCDense56 using the same training rules as the original paper [7], wherein they use two different scales of images to train the model. First they train the model with 224x224 resolution and then retrain it with 360x480 resolution. Because of the time constraint, we trained the FCDense56 only one time with 256x256 resolution (same as DensePN).

For DensePN, Xavier normal method is used to initialize the weights. Adam optimizer was used with initial learning rate of 1e-3, and decay half at every 100 epochs. For regularization, we set weight decay as 2e-4 and drop rate as 0.2.

4.3.1 Results and Discussion

The results comparing FCDense56 and our proposed DensePN on the CamVid dataset are shown in Table 4.3. From this table it can be seen that the number of parameters of DensePN is 26.7% less than those of FCDense56. Moreover, DensePN provides 0.5105 of mIoU compared to the 0.4655 mIoU of FCDense56. The DensePN outperforms the FCDense56 on seven of the 11 classes, namely the classes of 'sky', 'building', 'tree', 'sign_symbol', 'fence', 'car' and 'pedestrian'.

Modle	Parameters	IoU											mIoU
		sky	building	pole	road	pavement	tree	sign_symbol	fence	car	pedestrian	bicyclist	
FCDense56	1.5M	0.9029	0.6383	**0.2533**	**0.905**	**0.7163**	0.5872	0.14	0.1346	0.4071	0.2457	**0.1897**	0.4655
DensePN	**1.1M**	**0.9153**	**0.729**	0.1832	0.8788	0.6778	**0.6565**	**0.1903**	**0.3147**	**0.6203**	**0.2763**	0.173	**0.5105**

Table 4.3: The results of FCDense56 and DensePN.

The example segmentation outputs are shown in Figures 4.3 and 4.4.

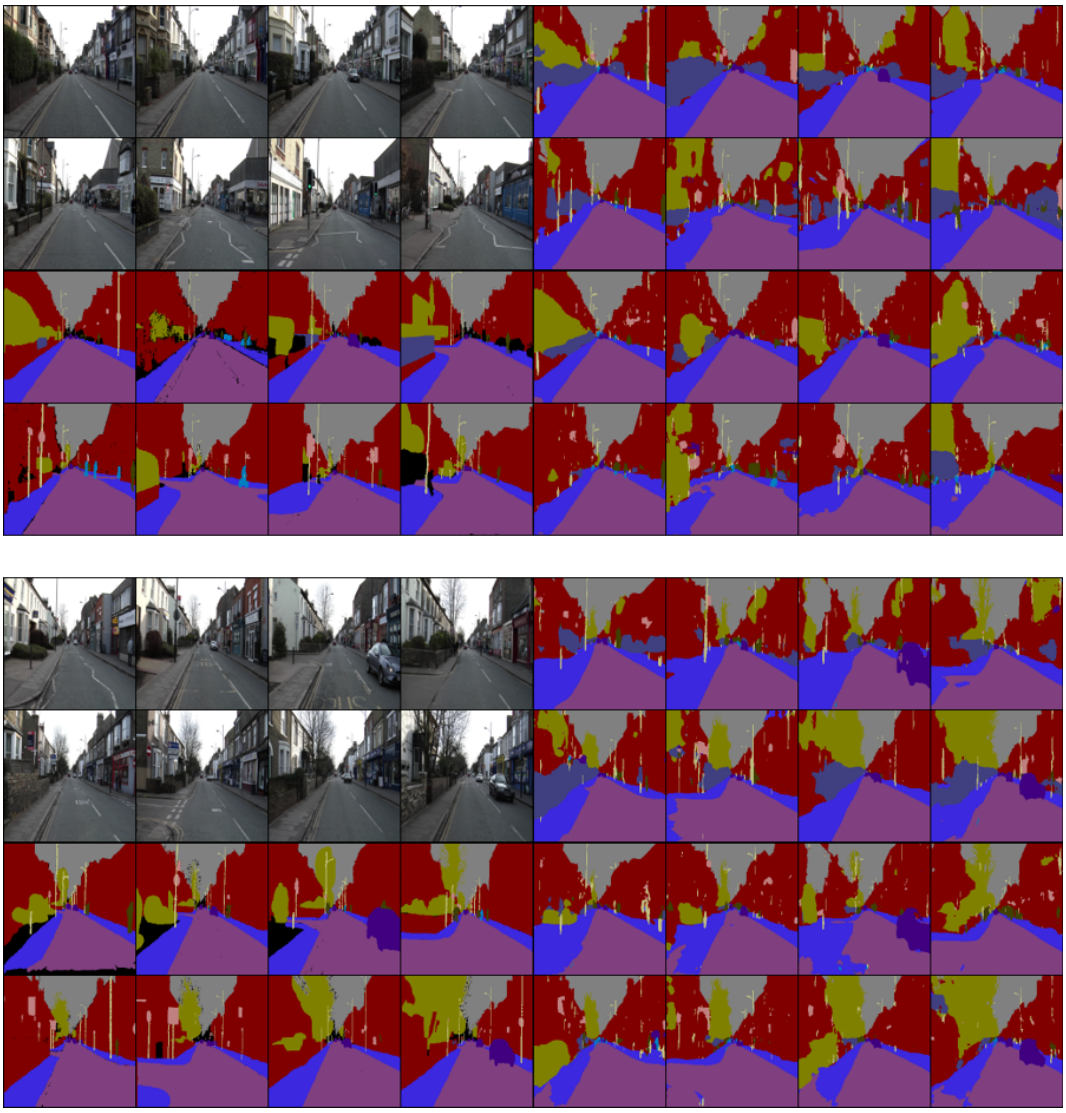

Fig. 4.3: The outputs of DensePN and FCDense56 on the CamVid dataset. The left half from top to bottom are the input images and ground truth segmentation. The upper right half are the results of FCDense56 and the lower right half are the results of DensePN.

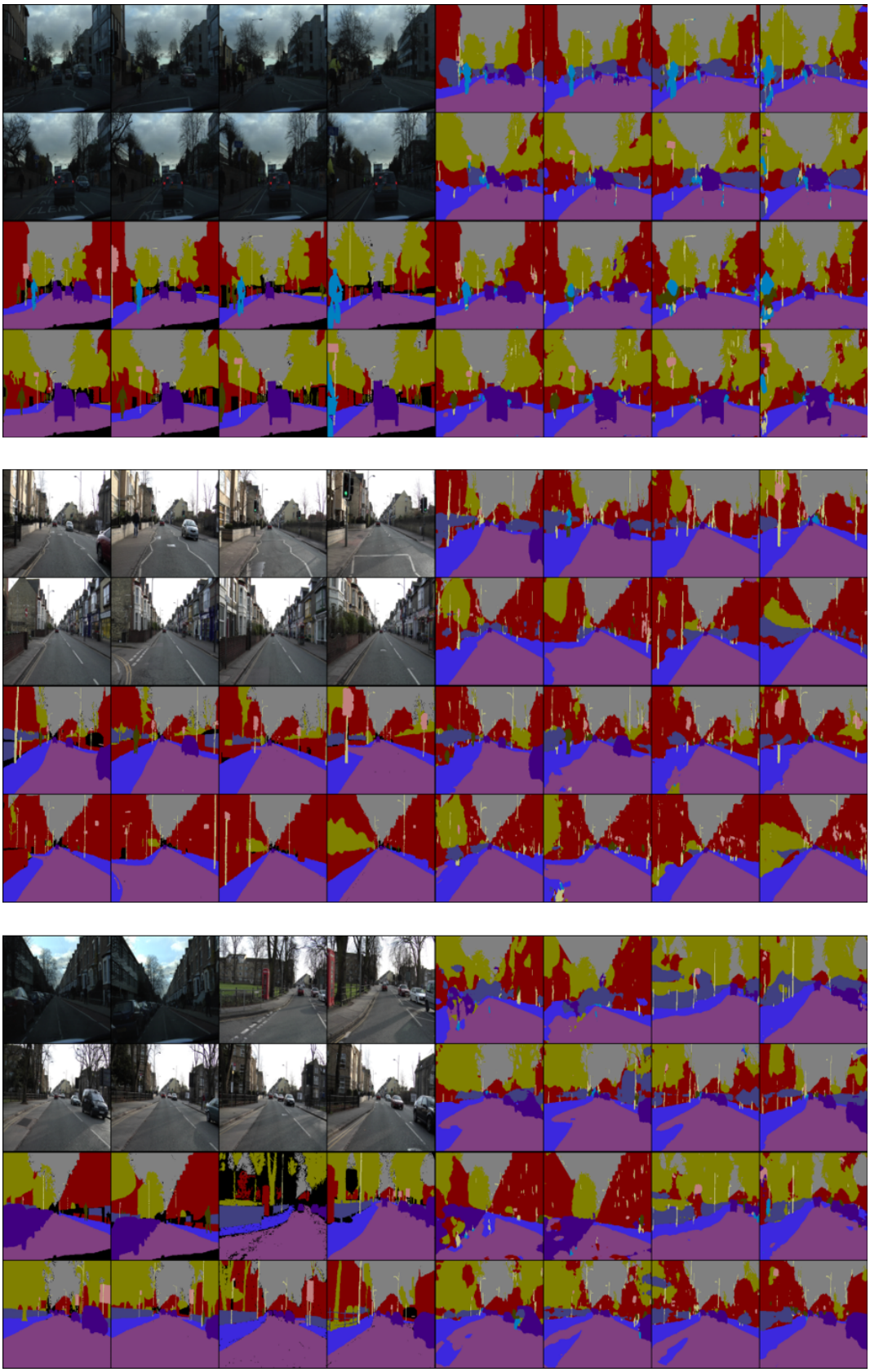

Fig. 4.4: The outputs of DensePN and FCDense56 on the CamVid dataset. The left half from top to bottom are the input images and ground truth segmentation. The upper right half are the results of FCDense56 and the lower right half are the results of DensePN.

Our DensePN architecure implicitly contains the benefits of DenseNet, namely feature reuse, deep supervision and parameter efficiency. In addition, it maintains the high resolution features throughout whole process. The parallel structure recursively feeds the abstract information from multi-scale features to high resolution features. These advantages allow the DensePN to handle the semantic segmentation task with relatively lower number of parameters.

We calculated the training loss after every epoch, and graphed the losses for FCDense56 and DensePN on left side of Figure 4.5. We also calculated the losses during validation after every five epochs, and presented the result on the right hand side of Figure 4.5. As can be seen, the training loss of FCDense56 is lower than the DensePN. However, the loss of DensePN is lower than FCDense56 during validation. This means that DensePN has less overfitting.

In a similar way, we obtained the graphs for the mIoU for both networks and show the results in Figure4.6. Same conclusions can be arrived from this figure, i.e. DensePN has less overfitting than FCDense56.

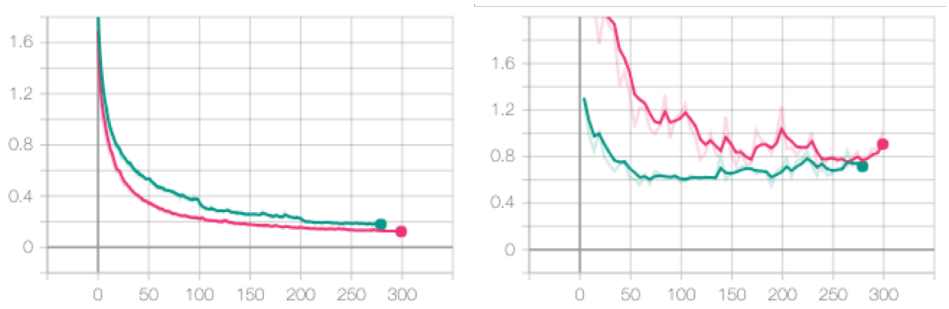

Fig. 4.5: Losses of FCDense56 and DensePN, Left: train losses. Right:validation losses. Red:FCDense56, Green:DensePN

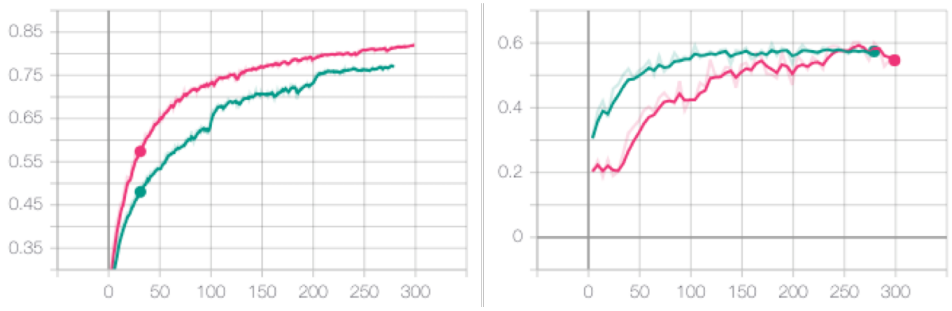

Fig. 4.6: mIoU of FCDense56 and DensePN, Left:train mIoU, Right:validation mIoU. Red:FCDense56, Green:DensePN

CHAPTER 5
CONCLUSION

In this thesis, we have studied different network models and approaches to perform image segmentation. More specifically, we have worked on, and proposed two different networks, namely Unet-Laplacian and DensePN.

The series network model we proposed, referred to as UNet-Laplacian, was inspired by U-net [6]. UNet reuses the previous high-resolution maps by concatenating them to Up path to enhance the spatial information, which is very important for the decoder part to do segmentation. Since object boundaries correspond to edges, we have proposed to include edge information in this framework in addition to the high-resolution feature maps. Laplacian filter is commonly used for edge detection. In our proposed UNet-Laplacian, we have added a new Laplacian branch, which performs the Laplacian filter operation on the input RGB image, and uses multi-scale max pooling to make the resolution same as the corresponding decoder features. The Laplacian filter can be implemented as a constant weight convolution kernel, and thus does not add any significant computational cost. Experimental results show that the proposed Unet-Laplacian provides slight improvement over the original U-net. The output of the Unet-Lapl closely follows that of the original U-net, but the output of Unet-Lapl captures more of the ground truth mask. We have observed that the CNN model is a high dimensional optimization problem, and we need ensemble

optimization to get better results.

We have also proposed a parallel flow network referred to as Dense Parallel Network (DensePN), which provides the ability for feature reuse, and has fewer parameters compared to FCDense56 [7]. DensePN incorporates ideas from HRNet [15] and DenseNet [2]. We have designed the Dense Parallel Network based on three main observations that we have gained from our initial trials and preliminary studies. First, maintaining a high-resolution feature map provides good performance. Second, feature reuse is very efficient, and allows having deeper networks. Third, having a parallel structure can provide better information flow. Experimental results show that the proposed DensePN provides a better performance than FCDense56 by having less parameters at the same time.

As future work, we will fine tune the Dense Parallel Network by retraining the model with higher resolution images. We will also try to expand the filters. Results show that Dense Parallel Network has a very large potential, and provides promising results. Currently, it has 46 filters, and we will add more filters to see how it performs. —

REFERENCES

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[2] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[3] Jonathan Long, Evan Shelhamer, and Trevor Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[4] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1520–1528.

[5] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[7] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 11–19.

[8] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig 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.

[9] Lichen Zhou, Chuang Zhang, and Ming Wu, "D-linknet: Linknet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2018, pp. 182–186.

[10] Tao Yang, Yan Wu, Junqiao Zhao, and Linting Guan, "Semantic segmentation via highly fused convolutional network with multiple soft cost functions," *Cognitive Systems Research*, vol. 53, pp. 20–30, 2019.

[11] Mohammed A Al-Masni, Mugahed A Al-antari, Mun-Taek Choi, Seung-Moo Han, and Tae-Seong Kim, "Skin lesion segmentation in dermoscopy images via deep full resolution convolutional networks," *Computer methods and programs in biomedicine*, vol. 162, pp. 221–231, 2018.

[12] Ye Li, Lele Xu, Jun Rao, Lili Guo, Zhen Yan, and Shan Jin, "A y-net deep learning method for road segmentation using high-resolution visible remote sensing images," *Remote Sensing Letters*, vol. 10, no. 4, pp. 381–390, 2019.

[13] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang, "Unet++: A nested u-net architecture for medical image segmentation," in

Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, pp. 3–11. Springer, 2018.

[14] Shreyas Saxena and Jakob Verbeek, "Convolutional neural fabrics," in *Advances in Neural Information Processing Systems*, 2016, pp. 4053–4061.

[15] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang, "Deep high-resolution representation learning for human pose estimation," *arXiv preprint arXiv:1902.09212*, 2019.

[16] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger, "Multi-scale dense convolutional networks for efficient prediction," *arXiv preprint arXiv:1703.09844*, vol. 2, 2017.

[17] Jesús Lázaro, José Luis Martín, Jagoba Arias, Armando Astarloa, and Carlos Cuadrado, "Neuro semantic thresholding using ocr software for high precision ocr applications," *Image and Vision Computing*, vol. 28, no. 4, pp. 571–578, 2010.

[18] Zhaojun Xue, Dong Ming, Wei Song, Baikun Wan, and Shijiu Jin, "Infrared gait recognition based on wavelet transform and support vector machine," *Pattern recognition*, vol. 43, no. 8, pp. 2904–2910, 2010.

[19] Shu-Kai S Fan and Yen Lin, "A multi-level thresholding approach using a hybrid optimal estimation algorithm," *Pattern Recognition Letters*, vol. 28, no. 5, pp. 662–669, 2007.

[20] Xiangyang Xu, Shengzhou Xu, Lianghai Jin, and Enmin Song, "Characteristic analysis of otsu threshold and its applications," *Pattern recognition letters*, vol. 32, no. 7, pp. 956–961, 2011.

[21] Kuo-Liang Chung and Chia-Lun Tsai, "Fast incremental algorithm for speeding up the computation of binarization," *Applied Mathematics and Computation*, vol. 212, no. 2, pp. 396–408, 2009.

[22] Deng-Yuan Huang and Chia-Hung Wang, "Optimal multi-level thresholding using a two-stage otsu optimization approach," *Pattern Recognition Letters*, vol. 30, no. 3, pp. 275–284, 2009.

[23] Marek Brejl and Milan Sonka, "Object localization and border detection criteria design in edge-based image segmentation: automated learning from examples," *IEEE Transactions on Medical imaging*, vol. 19, no. 10, pp. 973–985, 2000.

[24] Jayaram K Udupa, Supun Samarasekera, and William A Barrett, "Boundary detection via dynamic programming," in *Visualization in Biomedical Computing'92*. International Society for Optics and Photonics, 1992, vol. 1808, pp. 33–40.

[25] Alexander Toshev, Ben Taskar, and Kostas Daniilidis, "Object detection via boundary structure segmentation," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 950–957.

[26] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto, "Class segmentation and object localization with superpixel neighborhoods," in *2009 IEEE 12th international conference on computer vision*. IEEE, 2009, pp. 670–677.

[27] Thomas Leung and Jitendra Malik, "Contour continuity in region based image segmentation," in *European Conference on Computer Vision*. Springer, 1998, pp. 544–559.

[28] Iman Avazpour, Ros Roslan, Peyman Bayat, M Saripan, Abdul Nordin, and Raja Abdullah, "Segmenting ct images of bronchogenic carcinoma with bone metastases using pet intensity markers approach," *Radiology and Oncology*, vol. 43, no. 3, pp. 180–186, 2009.

[29] Pedro F Felzenszwalb and Daniel P Huttenlocher, "Efficient graph-based image segmentation," *International journal of computer vision*, vol. 59, no. 2, pp. 167–181, 2004.

[30] Yuri Boykov and Vladimir Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, , no. 9, pp. 1124–1137, 2004.

[31] Paul P Wyatt and J Alison Noble, "Map mrf joint segmentation and registration of medical images," *Medical Image Analysis*, vol. 7, no. 4, pp. 539–552, 2003.

[32] Anthony M Ungar, *Normalization, Cut-elimination, and the Theory of Proofs*, CSLI Stanford, 1992.

[33] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," in *ACM transactions on graphics (TOG)*. ACM, 2004, vol. 23, pp. 309–314.

[34] Bing Li and Scott T Acton, "Active contour external force using vector field convolution for image segmentation," *IEEE transactions on image processing*, vol. 16, no. 8, pp. 2096–2106, 2007.

[35] Kaihua Zhang, Huihui Song, and Lei Zhang, "Active contours driven by local image fitting energy," *Pattern recognition*, vol. 43, no. 4, pp. 1199–1206, 2010.

[36] Tony F Chan and Luminita A Vese, "Active contours without edges," *IEEE Transactions on image processing*, vol. 10, no. 2, pp. 266–277, 2001.

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[38] Yan Wu, Tao Yang, Junqiao Zhao, Linting Guan, and Jiqian Li, "Fully combined convolutional network with soft cost function for traffic scene parsing," in *International Conference on Intelligent Computing*. Springer, 2017, pp. 725–731.

[39] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[40] Kenneth Clark, Bruce Vendt, Kirk Smith, John Freymann, Justin Kirby, Paul Koppel, Stephen Moore, Stanley Phillips, David Maffitt, Michael Pringle, et al., "The cancer imaging archive (tcia): maintaining and operating a public information repository," *Journal of digital imaging*, vol. 26, no. 6, pp. 1045–1057, 2013.

[41] Noel CF Codella, David Gutman, M Emre Celebi, Brian Helba, Michael A Marchetti, Stephen W Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, et al., "Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic)," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. IEEE, 2018, pp. 168–172.

[42] Prasanna Porwal; Samiksha Pachade; Ravi Kamble; Manesh Kokare; Girish Deshmukh; Vivek Sahasrabuddhe and Fabrice Meriaudeau, "Indian diabetic retinopathy image dataset (idrid)," 2018.

[43] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[44] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[45] Vincent Dumoulin and Francesco Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[46] Alexander Andreopoulos and John K Tsotsos, "50 years of object recognition: Directions forward," *Computer vision and image understanding*, vol. 117, no. 8, pp. 827–891, 2013.

[47] Marcel van Gerven and Sander Bohte, *Artificial neural networks as models of neural information processing*, Frontiers Media SA, 2018.

[48] José Manuel Benítez, Juan Luis Castro, and Ignacio Requena, "Are artificial neural networks black boxes?," *IEEE Transactions on neural networks*, vol. 8, no. 5, pp. 1156–1164, 1997.

[49] Abien Fred Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[50] Kunihiko Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[51] Dominik Scherer, Andreas Müller, and Sven Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *International conference on artificial neural networks*. Springer, 2010, pp. 92–101.

[52] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[53] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[54] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[55] Min Lin, Qiang Chen, and Shuicheng Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[56] Matthew D Zeiler and Rob Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *arXiv preprint arXiv:1301.3557*, 2013.

[57] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[58] Nitish Shirish Keskar and Richard Socher, "Improving generalization performance by switching from adam to sgd," *arXiv preprint arXiv:1712.07628*, 2017.

[59] Sasha Targ, Diogo Almeida, and Kevin Lyman, "Resnet in resnet: Generalizing residual architectures," *arXiv preprint arXiv:1603.08029*, 2016.

[60] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[61] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[62] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[63] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[64] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu, "Deeply-supervised nets," in *Artificial Intelligence and Statistics*, 2015, pp. 562–570.

[65] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[66] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[67] Ross Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[68] Yisu Zhou, Xiaolin Hu, and Bo Zhang, "Interlinked convolutional neural networks for face parsing," in *International symposium on neural networks*. Springer, 2015, pp. 222–231.

[69] Damien Fourure, Rémi Emonet, Elisa Fromont, Damien Muselet, Alain Tremeau, and Christian Wolf, "Residual conv-deconv grid network for semantic segmentation," *arXiv preprint arXiv:1707.07958*, 2017.

[70] V Cokkinides, J Albano, A Samuels, ME Ward, and JM Thum, "American cancer society: Cancer facts and figures," *Atlanta: American Cancer Society*, 2005.

[71] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler, "The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Scientific data*, vol. 5, pp. 180161, 2018.

[72] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al., "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[73] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla, "Segmentation and recognition using structure from motion point clouds," in *ECCV (1)*, 2008, pp. 44–57.

[74] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. xx, no. x, pp. xx–xx, 2008.

Jiyang Wang
Master student of EECS at Syracuse University
Smart vision Systems lab

May 8, 2019
jwang127@syr.edu

Education

- **Anhui University of Science and Technology** Huainan, Anhui, Province China
  Bachelor of Engineering *2013 - 2017*
  - Major: Automation
- **Syracuse University** Syracuse, NY, United States
  Master of Science *2017 - 2019*
  - Major: Electrical Engineering

Master Thesis

- **Semantic Image Segmentation Via a Dense Parallel Network** 2019

Skills

- Python, c/c++, assembly, c#, MATLAB
- Tensorflow, Keras, Pytorch