Reproducible Machine Learning

Milad Ghanbari

Abstract

The main goal of this project is of reproducing the results of a machine learning paper followed by proposing some possible modifications for improving the performance of the baseline model. The method proposed in the main paper is called Densely Connected Convolutional Network (DensNet). Reducing the vanishing-gradient problem, reinforcing feature propagation and feature reuse are some of the benefits of DenseNet. After introducing the main model and duplicating some of the results from the original paper, the performance of the model is investigated by manipulating some key hyperparameters including learning rate, number of layers and growth rate. Three modifications are proposed for achieving higher performance, and evaluation of these modified models on image databases (CFAR10 and CFAR100) verified the increase in the performance but with the cost of more parameters.

1 Introduction

Image classification, the task of categorizing images into one of several predefined classes, is a fundamental issue in the computer vision area. It is the foundation for other computer vision tasks such as localization, detection, and segmentation [1]. In recent years, convolutional neural networks (CNNs) have become the main method for a wide range of computer vision tasks, e.g., image classification [2], object detection [3], semantic segmentation [4]. Authors in [5] published a seminal paper establishing the modern framework of CNN in 1990, and was improved later in [6]. A multi-layer artificial neural network called LeNet-5 was developed in the mentioned work [6] which was able to classify handwritten digits. LeNet-5 consists of multiple layers like other neural networks, and backpropagation algorithm can be used for training phase [7]. It can obtain effective representations of the original image, which makes it possible to recognize visual patterns directly from raw pixels with little-to-none preprocessing. A parallel study performed in [8] used a shift-invariant artificial neural network (SIANN) to recognize characters from an image. However, due to the lack of large training data and computing power at that time, their networks could not have an appropriate performance on more complex problems, e.g., large-scale image and video classification. Since 2006, many researchers have proposed various approaches to overcome the difficulties encountered in training deep CNNs [9, 10, 11, 12]. Most remarkably, a novel CNN architecture (AlexNet) was developed in [10] and considerable improvements were seen based on this method on the image classification compared to previous approaches.

In order to solve per pixel prediction problems, a natural extension of CNNs called Fully Convolutional Networks (FCNs) was developed in [13, 14]. FCNs add upsampling layers to standard CNNs to recover the spatial resolution of the input at the output layer. Consequently, images of arbitrary size can be processed by FCNs. In order to compensate for the resolution loss induced by pooling layers, FCNs introduce skip connections between their downsampling and upsampling paths. Skip connections help the upsampling path recover fine-grained information from the downsampling layers [15].

Among CNN architectures extended as FCNs for semantic segmentation purposes, Residual Networks (ResNets) [16] make an interesting case. ResNets incorporate additional paths to FCN (shortcut paths) and, thus, increase the number of connections within a segmentation network [15]. Besides ResNets,

Highway Networks [17] and FractalNets [18] pass the information about the input or gradient through many layers as well. The signal from one layer to the next is bypassed via identity connections in Highway Networks [17]. In FractalNets repeatedly several parallel layer sequences with a different number of convolutional blocks are combined to achieve a large nominal depth while keeping many short paths in the network [18]. Recently, a new CNN architecture, called DenseNet, was introduced in [19]. It is worth to mention that this report is developed based on this paper. Creating short paths from early layers to later layers is the key characteristic shared by all recent architectures. The main advantage of the architecture proposed in [19] is that it requires fewer parameters compared to traditional architectures, by having the same performance.

In the rest of the report, the significant works related to the current topic have been investigated followed by introducing the main model proposed in [19]. First, we make an attempt to reproduce some of the results of the DenseNet paper. Then, the key hyperparameters of the main model have been manipulated for analyzing the performance of the model. Finally, we proposed three modifications to the model for achieving better performance (e.g., replacing the pooling layer with a convolution layer in the architecture of the model).

2 Related Work

Various CNN architectures have been proposed for image classification. In this review, we tried to address those with significant performance, starting with AlexNet which was designed for image classification of ImageNet Challenge [23]. It had outstanding performance in ImageNet competition in 2012. AlexNet uses local response normalization, maxpooling with overlapping (window size 3, stride 2), a batch size of 128 examples, the momentum of 0.9 and weight decay of 0.0005 [24]. The next leading network is VGG-Net [25]. This architecture increases the depth while making all filters with at most 3×3 size. The logic behind this architecture is that 2 consecutive 3×3 conv. layers will have an effective receptive field of 5×5, and 3 of such layers an effective receptive field of 7×7 when combining the feature maps [24]. VGG-16 and VGG-19 are two most commonly used versions of this CNN, respectively with 16 weight layers and 19 weight layers.

ResNet appeared by seeking for the answer to this question: can stacking more layers lead to better performance [26]. Authors of [26] proposed the use of residual blocks for networks with 34 to 152 layers. Those blocks are designed to preserve the characteristics of the original vector x before its transformation by some layer $H_l(x)$ by skipping weight layers and performing the sum $H_l(x) + x$. Due to the fact that the gradient is an additive term, it is not probable to vanish even with many layers. In this architecture, after the last convolution layer, an average pooling is computed followed by the output layer. GoogLeNet proposed in [27] gained high attention due to its unique architecture based on modules named Inception [28]. The most important features of GoogLeNet architecture can be summarized as follows: 1) decrease of representation size from input to output; 2) use of higher dimensional representations per layer (and consequently more activation maps); 3) use of lower dimensional embedding using 1×1 convolutions before spatial convolutions; 4) balance of width (number of filters per layer) and depth (number of layers). Some other notable network architectures with competitive performance are Network in Network (NIN) structure [29], Deeply Supervised Network (DSN) [30], Ladder Networks [31, 32] and Deeply-Fused Nets [33]. All the mentioned network structures have achieved competitive results.

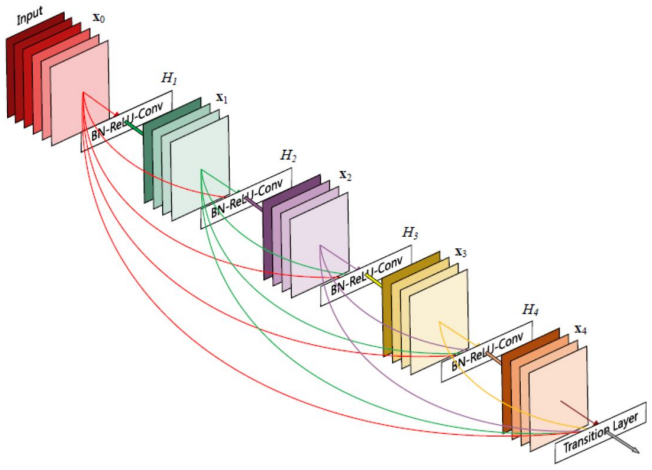

Figure 1: Illustration of a DenseBlock (adapted from [19])

3 Main Model

The core approach used for all the following evaluations is based on Densely Connected Convolutional Network (DensNet) [19]. In a standard feed forward network, the output of the l^{th} layer is computed as $x_l = H_l(x_{l-1})$ where the network input is denoted as x_0 and $H_l(.)$ is a non-linear transformation which can be a composite function of operations such as Batch Normalization (BN) [20], rectified linear units (ReLU) [21], pooling, or convolution. ResNet [19] employs a skip connection which adds an identity mapping of the input to the non-linear transformation:

$$x_l = H_l(x_{l-1}) + x_{l-1} \tag{1}$$

With the inspiration of ideas from both Inception and ResNets, the DenseNet presents the DenseBlock, a sequence of layers where each layer l takes as input all preceding feature maps $x_0; x_1; x_2; ...; x_{l-1}$ concatenated. Thus, each layer produces an output which is a function of all previous feature maps [19], i.e.:

$$x_l = H_l([x_0; x_1; x_2; ...; x_{l-1}]) \tag{2}$$

As a consequence, while regular networks with L layers have L connections, each DenseBlock (shown in Figure 1) has a number of connections following an arithmetic progression, i.e., $L(L+1)/2$ direct connections. The DenseNet is a concatenation of multiple inputs of $H_l(.)$ into a single tensor. Each H_l is composed of three operations, in sequence: batch normalization (BN), followed by ReLU and then a 3×3 convolution. This sequence of operations is called pre-activation unit introduced in [22] and has the property of yielding a simplified derivative for each unit that is unlikely to be canceled out, which in turn would improve the convergence process.

Transition layers are layers between DenseBlocks composed of BN, a 1×1 Conv.Layer, followed by a 2×2 average pooling with stride 2. Variations of DenseBlocks were also experimented in [19] using bottleneck layers, in this case, each DenseBlock is a sequence of operations: BN, ReLU, 1×1 Conv., followed by BN, ReLU, 3×3 Conv. This variation is called DenseNet-B. Finally, a compression method is also proposed to reduce the number of feature maps at transition layers with a factor θ. When bottleneck and compression are combined, they refer to the model as DenseNet-BC. Each layer has many input feature maps; if each H_l produces k feature maps, then the l^{th} layer has $k_0 + k \times (l-1)$ input maps, where k_0 is the number of channels in the input image. The number of filters k is a hyperparameter defined in DenseNet as the growth rate [19].

Table 1: DenseNet Accuracy Results for CFAR10 and CFAR100 datasets. The term BC in 100-12-BC stands for bottleneck and compression

Dataset	Data Augmentation	Layer (l)-Growth Rate (k)	Error rate (%)	
			Ours	Paper
CFAR10	NO	40-12	8.62	7.00
		100-12	7.74	5.77
		100-12-BC	6.88	5.92
	Yes	40-12	7.11	5.24
		100-12	5.58	4.10
		100-12-BC	5.41	4.51
CFAR100	NO	40-12	30.99	27.55
		100-12-BC	29.47	24.15
	Yes	40-12	27.35	24.42
		100-12-BC	25.03	22.27

4 Reproducing the Results of the DenseNet paper

The source code provided by the DenseNet authors is written in LUA [34]. Alternatively, we utilize another code suggested by the authors written in python, and the code is build based on keras platform to implement convolutional neural networks [35].

Table 1 shows the error rates of DenseNet on CIFAR10 and CIFAR100 datasets from DenseNet paper and our implementations. We use mirroring and shifting for data augmentation as suggested by the DenseNet paper. For CIFAR10 and CIFAR100 datasets, the initial learning rate is set to 0.1 and is divided by 10 at 50% and 75% of the total number of training epochs. The DenseNet model is trained for 300 epochs. CIFAR10 and CIFAR100 have 10 and 100 classes respectively. As it can be observed from Table 1, we could not reproduce the exact results provided by the DenseNet paper. However, our results are close to the paper. It is worth mentioning that we achieved different results from the same DenseNet configuration in different runs. As a result, we suspect that the authors of the paper did the same and reported the best results.

5 Analysis of Key Hyperparameters

Here, we evaluate the impact of key hyperparameters on the DenseNet performance. These hyperparameters are *Learning Rate*, *Number of Layers (L)* and *Growth Rate (k)*. Along with the mentioned hyperparameters, we also assess the Adam optimizer performance on the DenseNet.

5.1 Learning Rate

The initial learning rate (as mentioned in the DenseNet paper) is 0.1 and is divided by 10 at 50% and 75% of the total number of training epochs. We try 0.01 and 0.001 as the initial learning rate. Furthermore, we develop an adaptive learning rate reducer that divides the learning rate by 5 whenever it sees no improvement in the accuracy of the validation set over five epochs. We employ Adam optimizer as well. The evaluation is made using DenseNet model with 40 layers and the growth rate of 12 on the CFAR10 dataset with data augmentation. Fig. 2 and Fig. 3 show the accuracy of train and validation sets

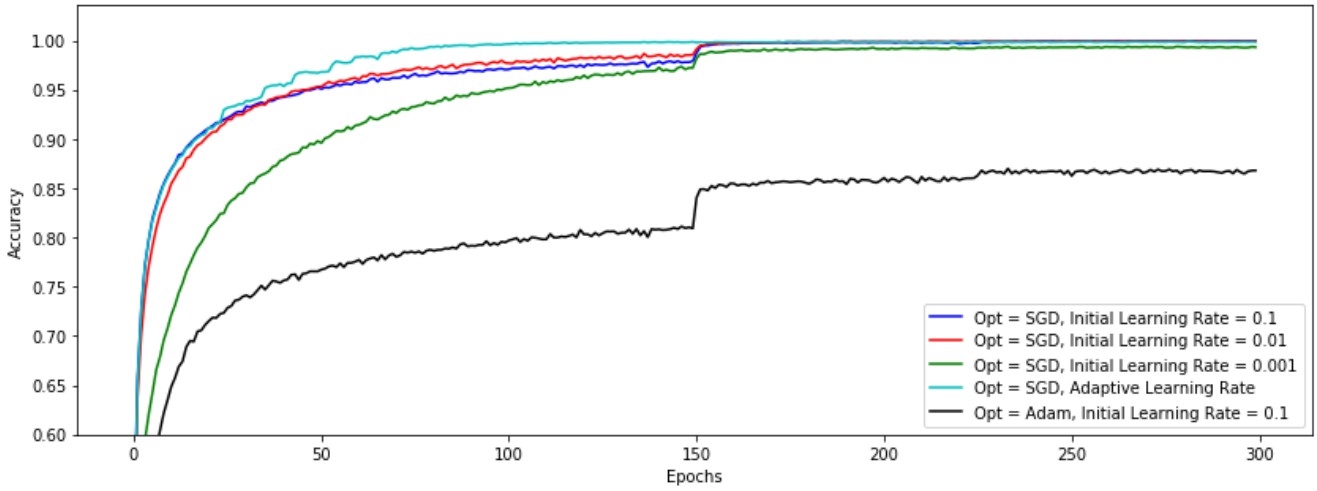

Figure 2: Accuracy over different learning rates and optimizer on train set.

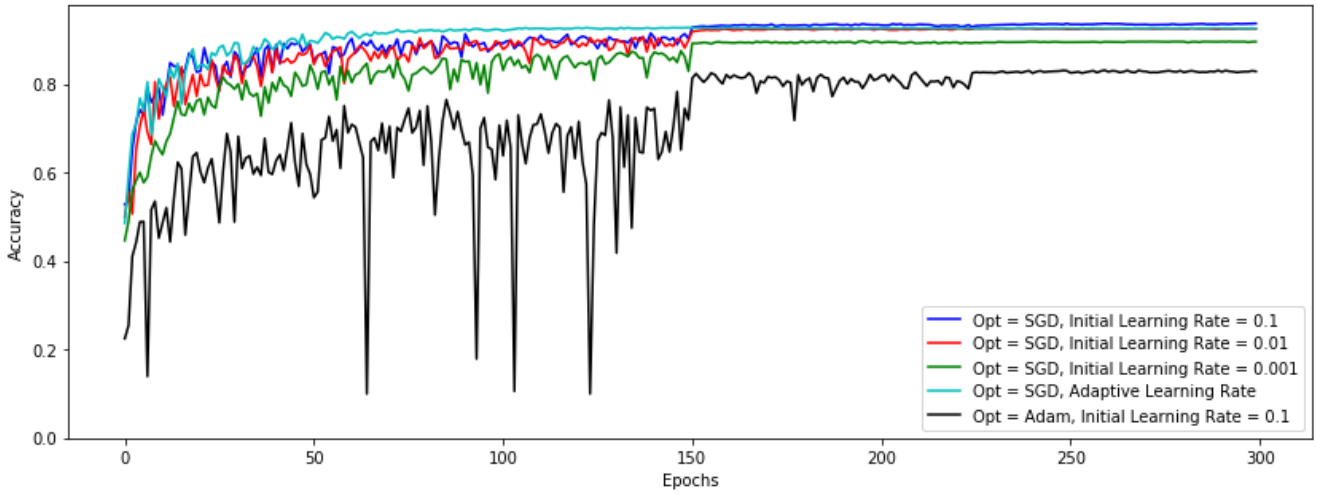

Figure 3: Accuracy over different learning rates and optimizer on validation set.

respectively. We can see that both learning rates of 0.1 and 0.01 behave similarly. Moreover, it is evident that the value of 0.001 is very small and it convergences slower than other learning rates. According to the results, the DenseNet model has the fastest convergence when using the adaptive learning rate reducer and using Adam optimizer has the worst performance among all. Adam optimizer has the most fluctuations on the validation set as well.

5.2 Number of Layers and Growth Rate

It is fair to claim that the number of layers (l) and growth rate (k) are the most important hyperparameters of the DenseNet. Increasing these two hyperparameters yields a higher accuracy on both train and validation sets. In order to assess their impact on DenseNet performance, we change the value of each hyperparameter and keep the other one constant. Five different l and k values are evaluated. It is worth mentioning that we tried our best to choose hyperparameters wisely so that the number of trainable

Table 2: DenseNet Accuracy Results for CFAR10 over different Growth Rate Values (k)

Layer (l)-Growth Rate (k)	**Parameters (M)**	**Accuracy (%)**
40-06	0.270	90.71
40-12	1.059	92.89
40-18	2.365	93.29
40-24	4.189	93.62
40-30	6.530	93.73

Table 3: DenseNet Accuracy Results for CFAR10 over different Number of Layers (l)

Layer (l)-Growth Rate (k)	**Parameters (M)**	**Accuracy (%)**
22-12	0.290	90.91
40-12	1.059	92.89
58-12	2.307	93.30
76-12	4.034	93.27
94-12	6.241	92.98

parameters becomes close for each pair of l and k values. For this experiment, we use the simple DenseNet model without bottleneck and compression on CIFAR10 dataset with data augmentation.

First, we vary k values and set the value of l to constant value of 40. We evaluate the DenseNet model with $k = \{6, 12, 18, 24, 30\}$. The results are shown in Table 2. Next, we try different l values ($l = \{22, 40, 58, 76, 94\}$) and assign a constant of 12 to the l value as shown in Table 3.

Fig. 4 shows the accuracy (with respect to the number of parameters) of the aforementioned experiment. According to Fig. 4, it is evident that increasing the l and k values improves the DenseNet performance with respect to the number of trainable parameters. As a result, one can increase either l or k values separately or increase both l and k values simultaneously to improve the DenseNet model. However, it should be noticed that the accuracy starts dropping for $l > 58$.

6 Modification on the Architecture

As mentioned previously, the transition layer of the DenseNet model consists of a 1×1 convolution followed by a 2×2 pooling layer. It is shown that convolution layers can supersede the pooling layer in CNNs [36]. In this regard, we make an attempt to replace the pooling layer with a convolution layer in three different approaches.

Approach A: First, we only replace the pooling layer with a 2×2 convolution layer with a stride of 2.

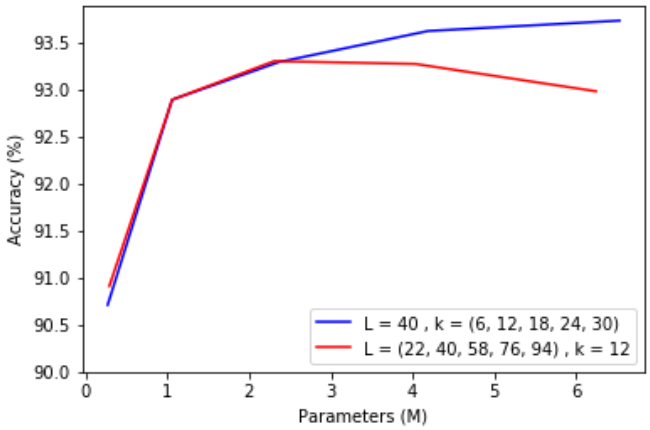

Figure 4: Accuracy over different number of layers and growth rates with respect to the number of parameters.

We also keep the previous layer's number of channels.

Approach B: In another approach, we replace both the 1×1 convolution and the 2×2 pooling layer with a 2×2 convolution layer with a stride of 2.

The number of channels grows at each layer of each dense block by concatenating the channels of previous layers. In another attempt, we use addition to grow the number of channels.

Approach C: In this approach, we grow the number of channels at the end of each dense block. Furthermore, instead of concatenating, we add the previous channels to the next layer of each dense block. In each dense block of DenseNet, the number of channels at each layer grows by k. Also, there are $(l-4)/3$ layers in each dens block. Therefore, the total number of channels added by the dense block is $k \times (l-4)/3$. In approach C, we only grow the number of channels at the end of the dense block by the rate of $k \times (l-4)/3$.

We evaluate these three modified models on CIFAR10 dataset with data augmentation. We use the DenseNet model with 40 layers and the growth rate of 12 as the baseline for this experiment. The results are shown in Table 4. Although all modified models slightly improved accuracy, they are more expensive than the baseline model. For instance, approach C has almost 12 times more parameters than the baseline.

Table 4: Modified DenseNet Accuracy Results

Model	**Parameters (M)**	**Accuracy (%)**
Baseline	1.059	92.89
Approach A	1.561	93.02
Approach B	1.436	93.30
Approach C	12.428	93.24

7 Discussion and Conclusion

In this work, we tried to reproduce some of the DenseNet results provided in [19]. We evaluated the DenseNet performance on CFAR10 and CFAR100 datasets. We employed data augmentation on the datasets as well. Furthermore, the key hyperparameters of the DenseNet such as learning rate, number of layers and growth rate were analyzed. We utilized an adaptive learning rate reducer and showed its efficiency for the DenseNet which caused the DenseNet model to have the fastest convergence. Moreover, running experiments with different growth rates (k) and layers (l), we obtained interesting findings. First, one can always achieve higher accuracy by increasing the growth rate. On the other hand, growing the number of layers improved the accuracy by a specific number of layers which is 58 so that the accuracy started to drop after this number of layers. As a result, through selecting the growth rate and the number of layers wisely, one is able to achieve a trade-off between reasonable accuracy and number of parameters.

Finally, we proposed three modifications on DenseNet model which improved the performance on CIFAR dataset. To be more specified, in approach A, substituting the pooling layer with a convolution layer resulted in 0.13 % improvement in the accuracy. In approach B, replacing both convolution and pooling layers with a new convolution layer not only increased the accuracy 0.41 % but also had the least number of parameters compared to proposed approaches. And in approach C, although growing the number of channels and adding the previous channels to the next layer caused the accuracy to improve 0.35 %, the performance gain came with the increase in cost (i.e., number of parameters).

References

[1] A. Karpathy, L. Fei-Fei, (2016). Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3128–3137). Red Hook, NY: Curran.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, pages 1097–1105, 2012.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, pages 580–587, 2014.

[4] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, pages 3431– 3440, 2015.

[5] B. B. Le Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Handwritten digit recognition with a back-propagation network, in: Proceedings of the Advances in Neural Information Processing Systems (NIPS), pp. 396–404, 1989.

[6] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of IEEE 86 (11), 2278–2324, 1998.

[7] R. Hecht-Nielsen, Theory of the backpropagation neural network, Neural Networks 1 (Supplement-1), 445–448, 1988.

[8] W. Zhang, K. Itoh, J. Tanida, Y. Ichioka, Parallel distributed processing model with local space-invariant interconnections and its optical architecture, Applied optics 29 (32), 4790–4797, 1990.

[9] X.-X. Niu, C. Y. Suen, A novel hybrid cnn–svm classifier for recognizing handwritten digits, Pattern Recognition 45 (4), 1318–1325, 2012.

[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, International Journal of Conflict and Violence (IJCV) 115 (3), 211–252, 2015.

[11] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Proceedings of the International Conference on Learning Representations (ICLR), 2015.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9, 2015.

[13] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[14] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In In- ternational Conference on Medical Image Computing and Computer-Assisted Intervention (MICAI), 2015.

[15] S. Jegou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation, arXiv preprint arXiv:1611.09326, 2016.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015.

[17] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In NIPS, 2015.

[18] G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. arXiv preprint arXiv:1605.07648, 2016.

[19] G. Huang, Z. Liu, and K. Q.Weinberger. Densely connected convolutional networks. CoRR, abs/1608.06993, 2016.

[20] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in Proc. ICML, 2015.

[21] X. Glorot, A. Bordes, and Y. Bengio, Deep sparse rectifier neural networks, in Proc. AISTATS, 2011.

[22] K. He, X. Zhang, S. Ren, and J. Sun, Identity Mappings in Deep Residual Networks. Cham: Springer International Publishing, pp. 630–645, 2016.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database, in CVPR09, 2009.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks, in Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems., 2012, pp. 1106–1114.

[25] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, CoRR, vol. abs/1409.1556, 2014.

[26] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, CoRR, vol. abs/1512.03385, 2015.

[27] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, Rethinking the inception architecture for computer vision, CoRR, vol. abs/1512.00567, 2015.

[28] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, Rethinking the inception architecture for computer vision, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826, 2016.

[29] M. Lin, Q. Chen, and S. Yan. Network in network. In ICLR, 2014

[30] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeplysupervised nets. In AISTATS, 2015

[31] M. Pezeshki, L. Fan, P. Brakel, A. Courville, and Y. Bengio. Deconstructing the ladder network architecture. In ICML, 2016.

[32] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. Semi-supervised learning with ladder networks. In NIPS, 2015.

[33] J. Wang, Z. Wei, T. Zhang, and W. Zeng. Deeply-fused nets. arXiv preprint arXiv:1605.07716, 2016.

[34] https://github.com/liuzhuang13/DenseNet

[35] https://github.com/titu1994/DenseNet

[36] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. arXiv:1409.4842, 2014.