Important Convolutional Neural Networks

Deep Learning (ENGG*6600*07)

School of Engineering,
University of Guelph, ON, Canada

Course Instructor: Benyamin Ghojogh
Fall 2023

Introduction

There exist various convolutional neural networks. Some of the the most important and well-known ones are listed below.

- AlexNet (2010) [1]. It was published in 2017 but it competed in the ImageNet Large Scale Visual Recognition Challenge in 2010.
- VGG (2014) [2]
- Inception and GoogLeNet (2015) [3]
- U-Net (2015) [4]
- ResNet (2016) [5]
- DenseNet (2017) [6]

Most of these neural netwoprks were developed by big companies and they often were winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [7] in different years. This is an annual classification challenge in machine learning and image processing.

About ImageNet and ILSVRC

- ImageNet is an image dataset containing 15 million labeled high-resolution images with about 22,000 classes.
- The images of ImageNet were collected from the web and manually labeled by humans using Amazon's Mechanical Turk crowd-sourcing tool [8].
- Since 2010, ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [7] has been held as a part of the Pascal Visual Object Challenge [9].
- ILSVRC is an annual classification challenge in machine learning and image processing [1].
- The dataset in ILSVRC is a subset of the ImageNet with 1000 classes each of which having 1000 images.
- The ILSVRC dataset has about 1.2 million high-resolution images with 1000 classes. The images are 224×224 pixels with three channels of RGB (Red-Green-Blue).

AlexNet

AlexNet

The architecture of AlexNet is as follows.

- Eight layers: five convolutional layers and three fully connected layers
- The network of AlexNet is spread into two GPUs because it was too big to fit in the 3GB memory of one GPU. This also accelerates the training phase.
- The settings of the layers:
  - layer 1 filters $224 \times 224 \times 3$ input images with 96 kernels of size $11 \times 11 \times 3$ with stride 4.
  - layer 2 filters $11 \times 11 \times 3$ feature maps (output of layer 1) with 256 kernels of size $5 \times 5 \times 48$.
  - layer 3 filters $5 \times 5 \times 48$ feature maps (output of layer 2) with 384 kernels of size $3 \times 3 \times 256$.
  - layer 4 filters $3 \times 3 \times 256$ feature maps (output of layer 3) with 384 kernels of size $3 \times 3 \times 192$.
  - layer 5 filters $3 \times 3 \times 192$ feature maps (output of layer 4) with 256 kernels of size $3 \times 3 \times 192$.
  - Layers 6, and 7 are fully connected layers and have 4096 neurons. The last layer is a fully connected layer with 1000 neurons.
- All layers except the last layer have Rectified linear unit (ReLU) activation function [10]. This is because ReLU accelerates training and has been found to be effective.
- The last layer has 1000 neurons with softmax activation function for classification into 1000 classes.

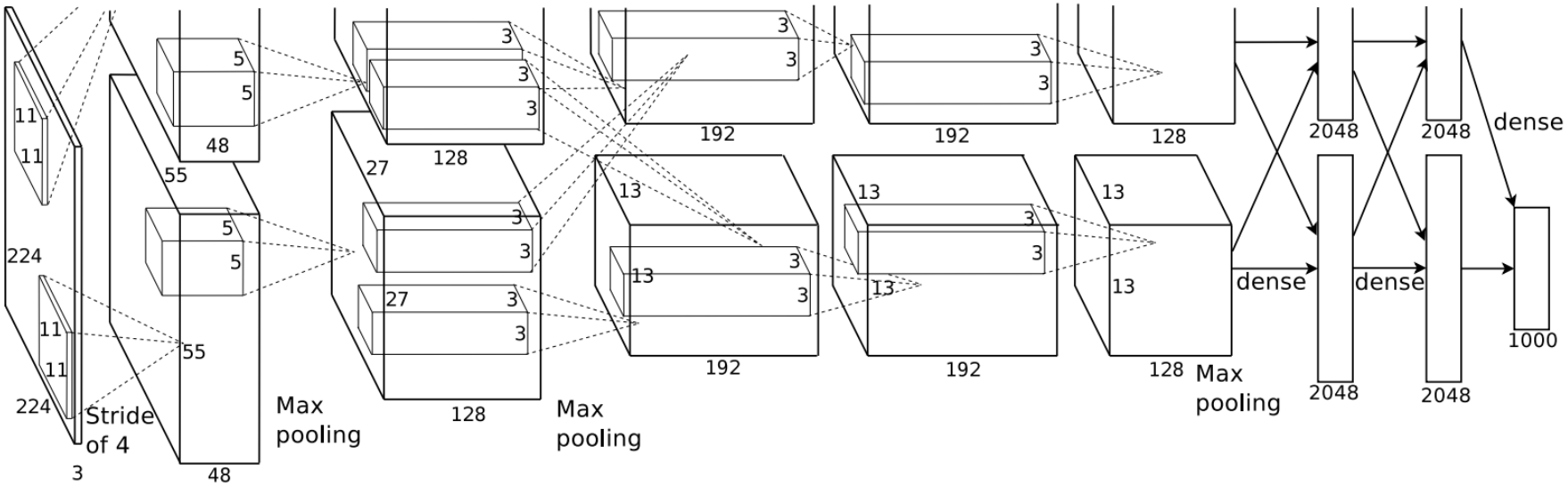
AlexNet

- In the layer 1 and layer 2, a local response normalization is applied after the ReLU activation function. It was empirically found to help generalization in AlexNet. Let $a^i_{x,y}$ denote the output of kernel i at the position (x, y) followed by ReLU activation function. Then, the local response normalization is:

$$b^i_{x,y} = \frac{a^i_{x,y}}{\left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^i_{x,y})^2\right)^\beta}, \tag{1}$$

  where the summation is over n adjacent kernel maps at the same spatial position and N is the total number of kernels in the layer. The hyper-parameters k, α, and β were determined by a validation set and were found to be $k = 2$, $n = 5$, $\alpha = 10^{-4}$, and $\beta = 0.75$.
- There is max pooling after layer 1, layer 2, and layer 5. In layers 1 and 2, this pooling is applied after the local response normalization. The width of pooling kernel is 3 and the stride is 2. Therefore, it uses overlapping pooling.
- AlexNet has 60 million parameters and 650,000 neurons.
- In the ILSVRC 2010, it achieved the top-1 and top-5 error rates of 37.5 % and 17.0%, respectively.
- AlexNet was named after Alex Krizhevsky, the main coauthor of the AlexNet paper [1].

AlexNet



credit of image: [1]

AlexNet

- The loss function of AlexNet is maximization of the multinomial logistic regression. If for a data instance $\boldsymbol{x}_i$, the target class label is ℓ, then it maximizes the ℓ-th output neuron in the softmax:

$$\max_{\theta} \quad \mathbb{P}(y_i = \ell) = \frac{e^{o_\ell}}{\sum_{j=1}^{1000} e^{o_j}} \implies \min_{\theta} \quad -\mathbb{P}(y_i = \ell) = -\frac{e^{o_\ell}}{\sum_{j=1}^{1000} e^{o_j}}, \tag{2}$$

  where o_j denotes the j-th output of network before the softmax activation function.

VGG

VGG

- VGG neural network was proposed by a team named Visual Geometry Group (VGG) at the University of Oxford [2].
- VGG addresses the depth of a neural network. It increased the depth of neural network to a very long depth in order to have more learnable parameters to handle the nonlinear patterns of data better. Of course, after increasing the learnable parameters, it used more training dataset not to have the problem of overfitting.
- For increasing the depth of network, it added many convolutional layers.
- In other words, VGG can be considered as a convolutional neural network with too many convolutional layers.
- VGG uses very small convolutional filter kernels with size 3×3. This small size of kernels was because of avoiding overfitting because having very deep network with large kernels will increase the number of learnable parameters too much which results in overfitting because of not having sufficient training data for learning.

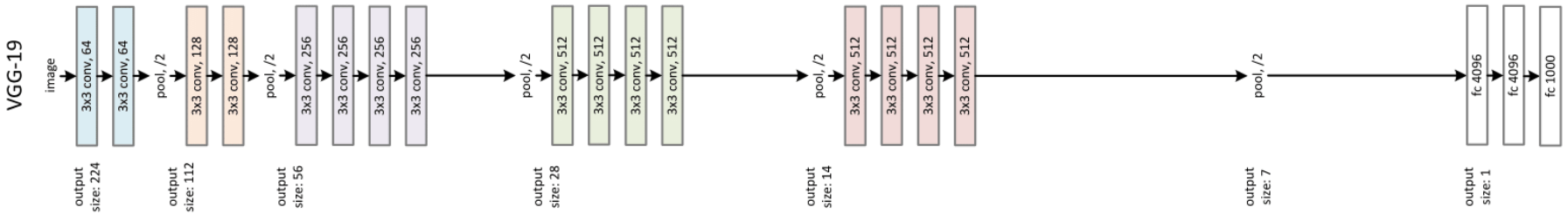
VGG

Network architecture of VGG:

- The means of RGB images of size $224 \times 224 \times 3$ are subtracted from the images as a pre-processing. This makes the mean of data zero.
- Most of the kernels were of size 3×3. A few ones were also of size 1×1 which means a linear transformation of features. The stride of convolution is set to 1. For the 3×3 convolutions, padding size is 1.
- Max pooling is used with width 2 and stride 2 so it is a non-overlapping pooling.
- In various versions of VGG, the number of convolutional layers differ but this number is usually large.
- After the convolutional layers, there are three fully connected layers. The first two fully connected layers have 4096 neurons and the last layer has 1000 layers for classification into 1000 classes.
- All hidden layers have ReLU activation function [10] and the last layer has softmax activation function.
- All versions of VGG, except one of them, do not use the local response normalization used in AlexNet. This is because they empirically found out that having this normalization does not improve the performance on the ILSVRC dataset.

VGG

- Mini-batch stochastic gradient descent with momentum was used for training VGG. The batch size was 256. Note that the batch size is usually set to a number which is a power of 2. The momentum parameter was set to 0.9 to have high momentum in VGG. This is set to number close to one not to have much oscillation in training.
- Weight decay with ℓ_2 norm was used for regularization. The regularization parameter of weight decay was set to 5×10^{-4}. Note that the regularization parameter is usually a very small number not to waive the impact of the actual loss function.
- The initial learning rate was 10^{-2} and it was decreased by a factor of 10 (divided by 10) every time the validation accuracy stopped improving. Note that the learning rate should be small enough not to have oscillation but not too small so there is progress in training.



credit of image: [5]

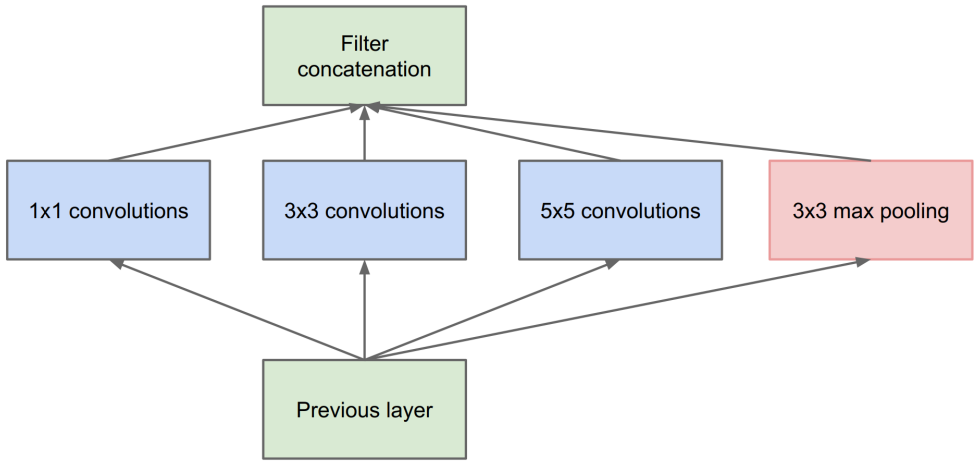
Inception and GoogLeNet

Inception and GoogLeNet

- Inception is a module of neural network proposed in [3].
- The same paper proposed GoogLeNet composed of Inception modules.
- The name of GoogLeNet has two inspirations. The first inspiration is the name of Google as it was proposed by Google employees. The second inspiration is the LeNet [11] network which is one of the first successful convolutional networks.
- In the ILSVRC 2014, GoogLeNet achieved the top result and won the competition.
- There are two ways to increase the number of learnable parameters in a neural network. Either the depth of the network or the number of hidden neurons can be increased or both.
- This increase in the number of learnable parameters has two drawbacks. First, it makes the network prone to overfitting especially if there are not sufficient labeled training data (and labeling data is time and resource consuming). Second, more parameters need more training requiring more computational resource.
- For resolving these two drawbacks, the Inception module was introduced which inserts sparsity to the network. This is also beneficial for the "betting on sparsity principal" [12, 13], the Occam's razor [14],

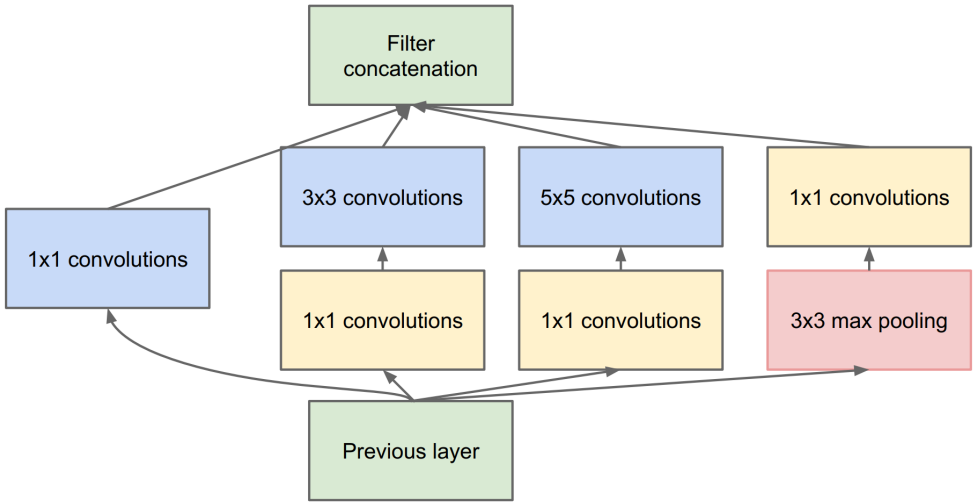
Inception and GoogLeNet

- It is possible to suppose that every neuron in the previous layer of a layer corresponds to some region of the input image.
- On the one hand, in the initial layers of network, the correlated neurons concentrate in a single region. As a result, many clusters of neurons concentrate in a single region. These clusters for a single region can be covered by a layer of 1×1 convolution. Note that a 1×1 convolution can be seen as a linear transformation of features.
- On the other hand, there are a smaller number of more spatially spread clusters of neurons which can be covered together by convolution on larger patches.
- Note that the larger and more spread the regions are, the less the number of the cluster is. Therefore, only small convolutional kernels are used. This explains why the Inception module is sparse as it does not use large convolutional kernels which are denser.
- Therefore, for the above explanations, the Inception module uses convolutions with filter sizes 1×1, 3×3, and 5×5.
- Moreover, because of the benefits of pooling discussed before, 3×3 max pooling is also used.



Inception and GoogLeNet

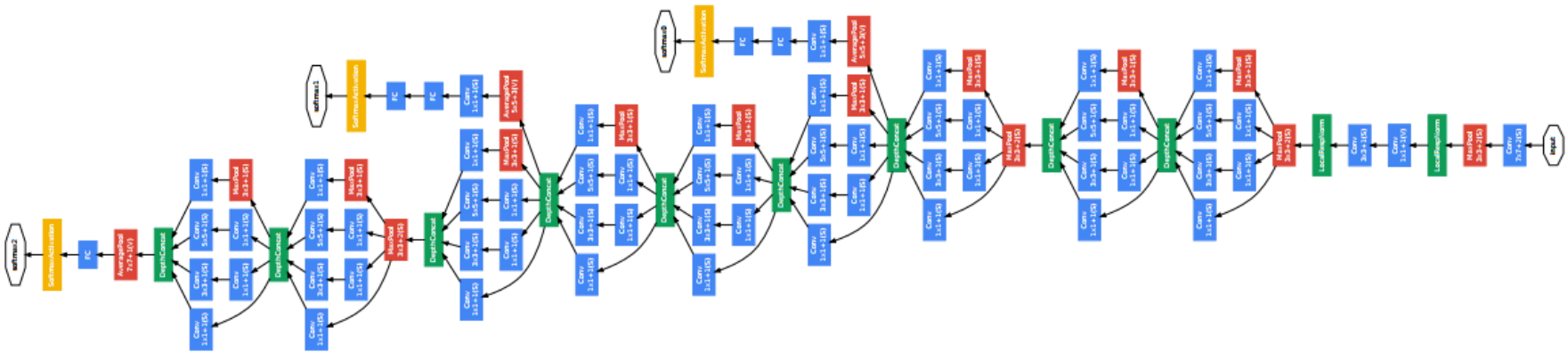
- GoogLeNet stacks the Inception modules as its layers.
- The number of 1×1, 3×3, and 5×5 convolutions can change in the Inception module depending on where it is used in the network.
- In the higher layers close to the output of network, more abstract features are extracted; therefore, the larger convolutional kernels are required more in the higher layers. As a result, the Inception modules in higher layers of network should contain more 3×3 and 5×5 convolutions compared to 1×1 convolution.
- However, having larger convolutional kernels results in too much computational complexity because for example the 5×5 convolution is applied on the result of the 5×5 convolution in the previous layer. Having multiple layers of these large convolutions results in computational blow up. To resolve this problem, dimensionality reduction by 1×1 convolution (linear transformation of features) is added in the Inception module not to have computational blow up in complexity. In other words, 1×1 convolutions are added before the expensive 3×3 and 5×5 convolutions and after the 3×3 max pooling.



Inception and GoogLeNet

Network architecture of GoogLeNet:

- The means of RGB images of size $224 \times 224 \times 3$ are subtracted from the images as a pre-processing. This makes the mean of data zero.
- GoogLeNet is a stack of Inception modules as its layers.
- All the convolutions, including the ones within the Inception modules, use the ReLU action function [10].

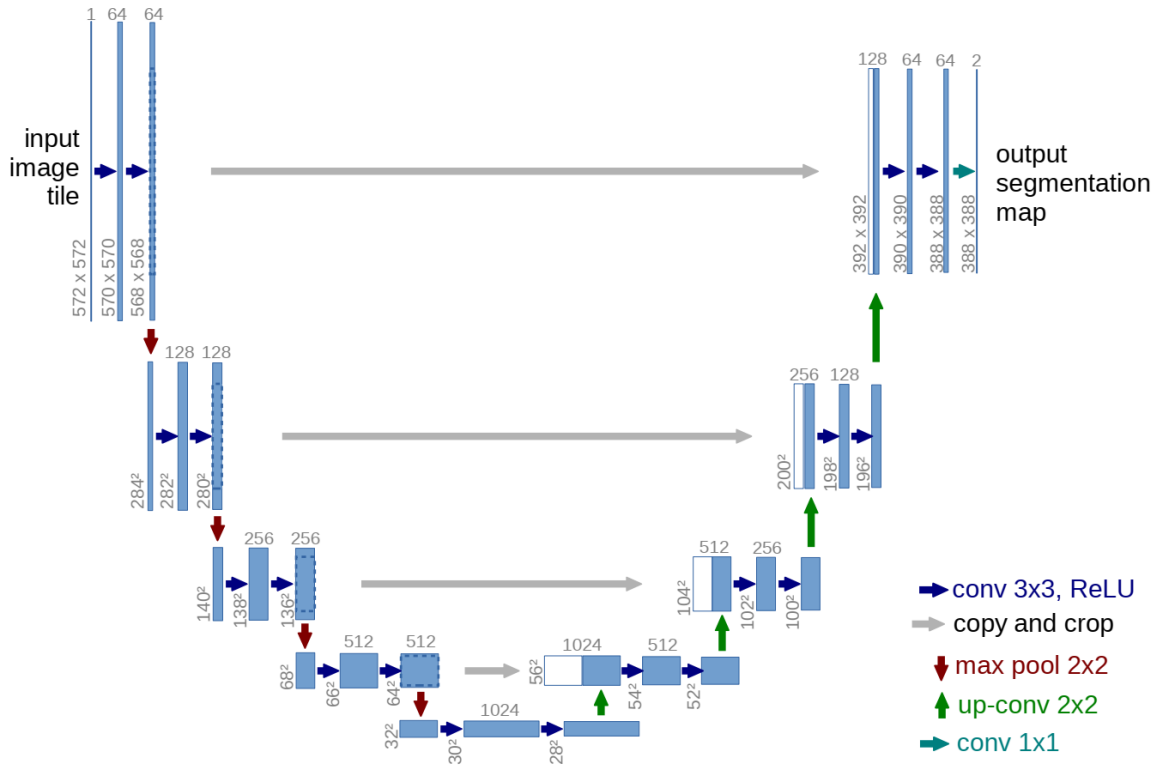


credit of image: [3]

U-Net

U-Net

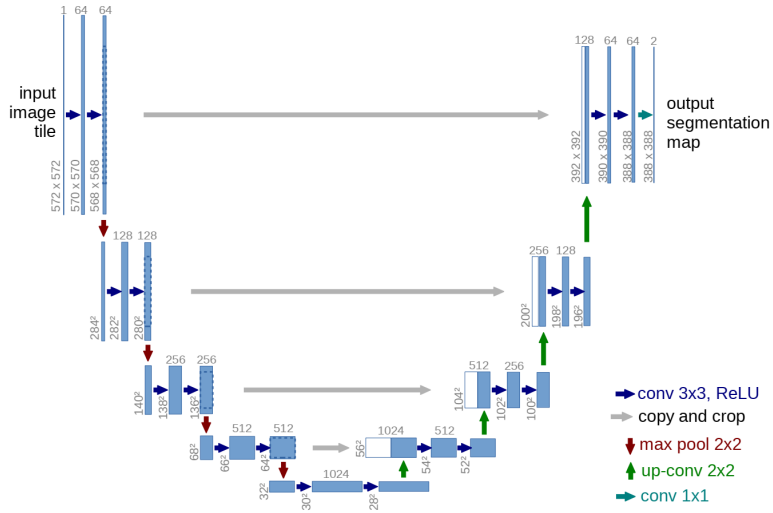
- U-Net was proposed in [4] for a biomedical image processing application which was biomedical image segmentation.
- It was a winning submission for the ISBI challenge for segmentation of neural structures in electron microscopic stacks.
- It is named U-Net because its shape is like the letter U.
- A U-Net has a contracting path (like the downward edge of the letter U) and a symmetric expanding path (like the upward edge of the letter U).
- The contracting path is for capturing the context and the expanding path is for precise localization.



U-Net

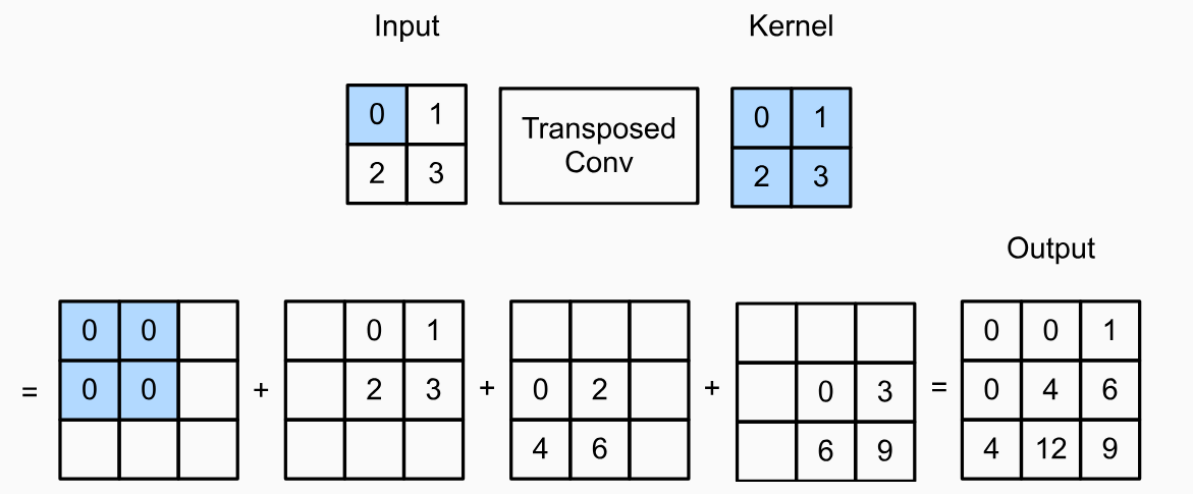
The network architecture of U-Net:

- The contracting path:
  - The layers have 3×3 convolutions followed by ReLU activation functions [10].
  - After every two 3×3 convolution followed by ReLU activation function, max pooling of width 2 and stride 2 is used for down-sampling.
  - At every down-sampling, the number of feature channels is doubled.
- The expansive path:
  - The layers have 3×3 convolutions followed by ReLU activation functions [10].
  - After every two 3×3 convolution followed by ReLU activation function, a 2×2 up-convolution (reverse of convolution) (also called transpose convolution) is used for up-sampling.
  - At every up-sampling, the number of feature channels is halved.
  - After every up-sampling, the result of up-sampling is concatenated with the cropped output of the corresponding layer in the contracting path. Cropping is required because the border pixels are removed at every convolution.



U-Net

- An example of transpose convolution (credit of image: `https://d2l.ai/chapter_computer-vision/transposed-conv.html`):



- U-Net was primarily implemented in the Caffe library for deep learning [15].
- Because of the large input tiles of biomedical images, the batch size was set to one in U-Net.
- Similar to VGG, U-Net also sets the momentum parameter to 0.9 to have high momentum and less amount of oscillation in training.
- The final feature map of U-Net has the same size as the input image. The final feature map has c number of channels, with softmax activation function, where c is the number of classes in image segmentation. The loss function of U-Net is cross entropy for classification of the pixels into one of the classes of segmentation.

ResNet

ResNet

- It has been empirically observed that, in very deep neural networks (e.g., with more than 20 layers), by increasing the depth of network, accuracy saturates and then degrades rapidly. In other words, by adding more layers to very deep networks, the training error increases. This problem is referred to as the **degradation problem**.
- The degradation problem is surprising. This is because a neural network with additional layers is a special case of the shallower network. In other words, if the shallow network is sufficient for learning, the additional layers are supposed to become identical mapping by optimization in backpropagation. However, this problem surprisingly exists.
- Residual Network (ResNet) was proposed in [5] which uses **deep residual learning** for resolving the degradation problem.

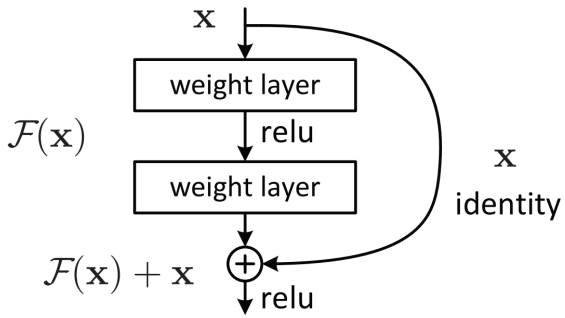
ResNet

- In the regular networks, It is hoped that each few stacked layers directly fit a desired underlying mapping.
- In the deep residual learning, however, each few stacked layers fit a residual mapping rather than the direct mapping.
- Let the desired original underlying mapping be denoted by $\mathcal{H}(x)$ for the input x to the stacked layers. In deep residual learning, the stacked nonlinear layers fit another mapping:

$$\mathcal{F}(x) := \mathcal{H}(x) - x. \tag{3}$$

Therefore, the desired original underlying mapping is:

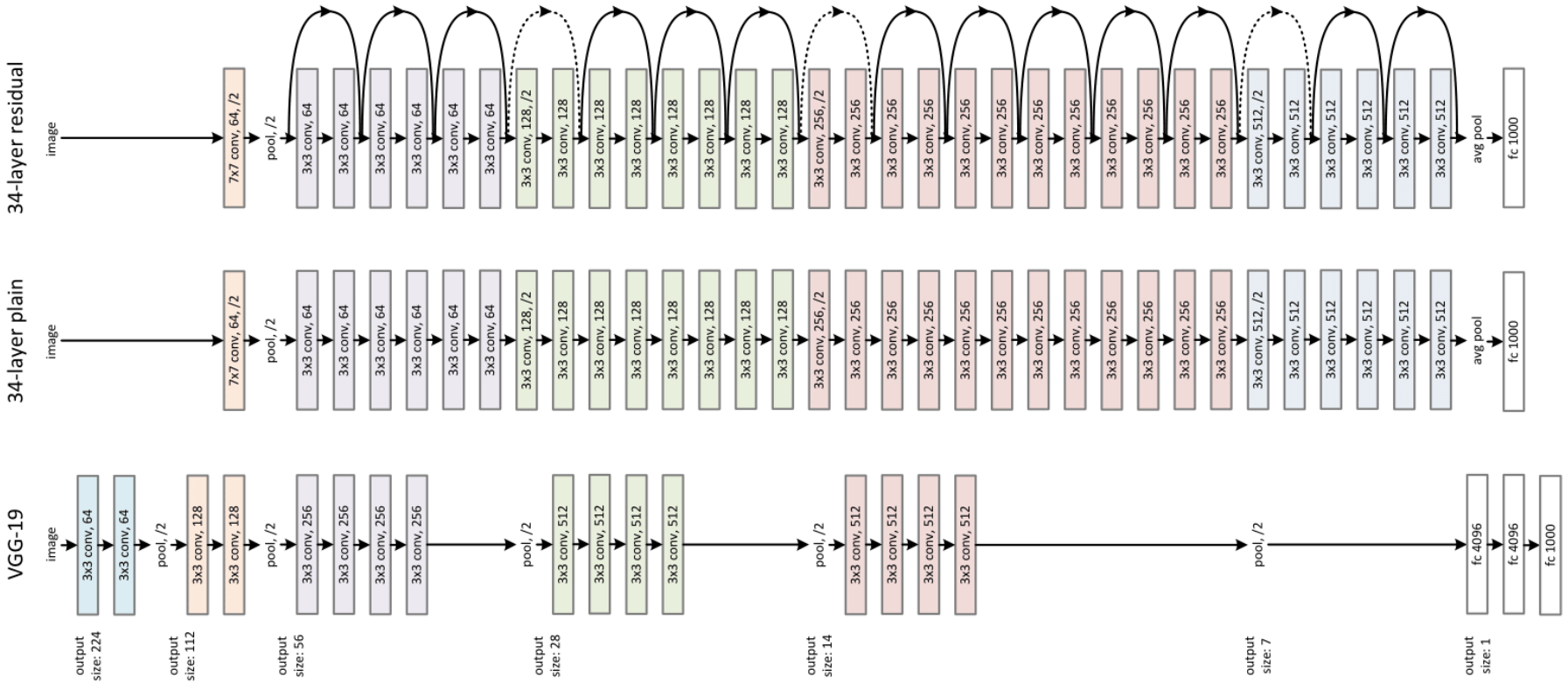
$$\mathcal{H}(x) = \mathcal{F}(x) + x. \tag{4}$$



- Benefit of ResNet: in backpropagation, gradient can flow directly through the identity function from the next layers to the previous layers.
- The deep residual learning postulates that it is easier to optimize the residual mapping than to optimize the desired original underlying mapping which is unreferenced. Empirically, this postulation has been approved to be correct.

ResNet

- Variants of ResNet: ResNet 18, ResNet 34, ResNet 50, ResNet 101. Choose based on your computer, memory, CPU, application, and data.
- Comparison of archtectures of regular net, VGG, and ResNet:

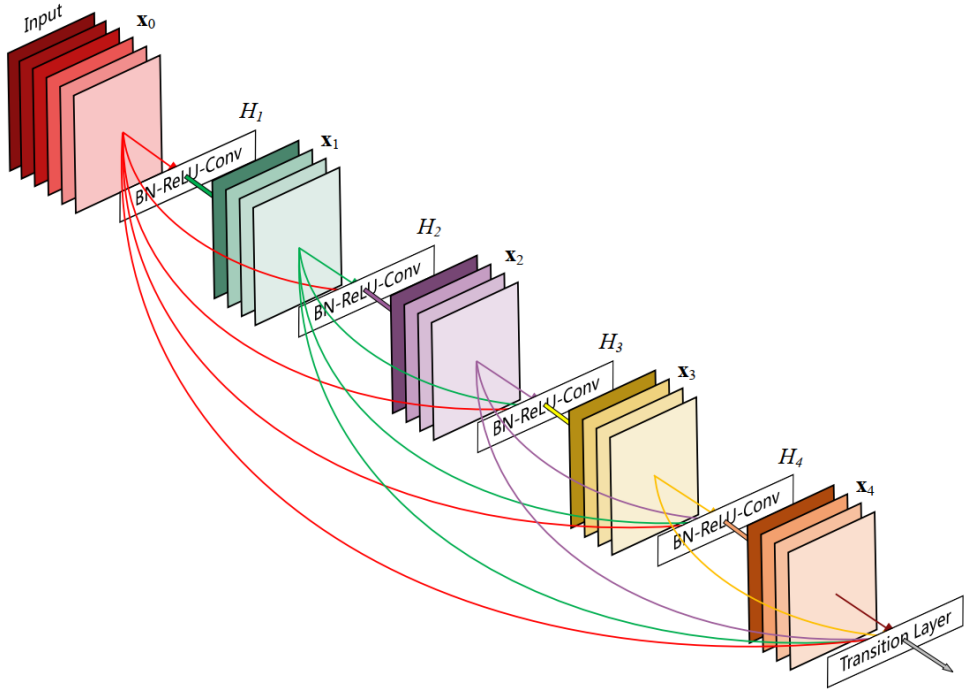


credit of image: [5]

DenseNet

DenseNet

- Empirically, it is observed that convolutional neural networks can become deeper, more accurate, and efficient to train if there exist shorter connections between the initial layers and the last layers.
- In Dense Convolutional Network (DenseNet) [6], every layer is connected to every other layer.
- In a regular network with l layers, the number of weight matrices is l as every layer is connected to its next subsequent layer.
- In DenseNet, however, there are $l(l+1)/2$ weight matrices between the layers as every layer is connected to all its next layers and not merely its next subsequent layer.



credit of image: [6]

DenseNet

- The benefits of DenseNet:
  - alleviate the gradient vanishing problem. This is because, in backpropagation, gradient can flow directly through all layers to every other previous layers.
  - strengthen feature propagation
  - encourage feature reuse
  - reduce the number of learnable parameters. This is because DenseNet requires less number of layers than regular networks as it already has enough learnable parameters between the layers.
- Let $\mathcal{F}_\ell(x)$ denote the composite function of batch normalization [16], convolution, ReLU activation function [10], and possibly pooling at the layer ℓ.
- ResNet uses:

$$x_\ell = \mathcal{F}_\ell(x_{\ell-1}) + x_{\ell-1}. \tag{5}$$

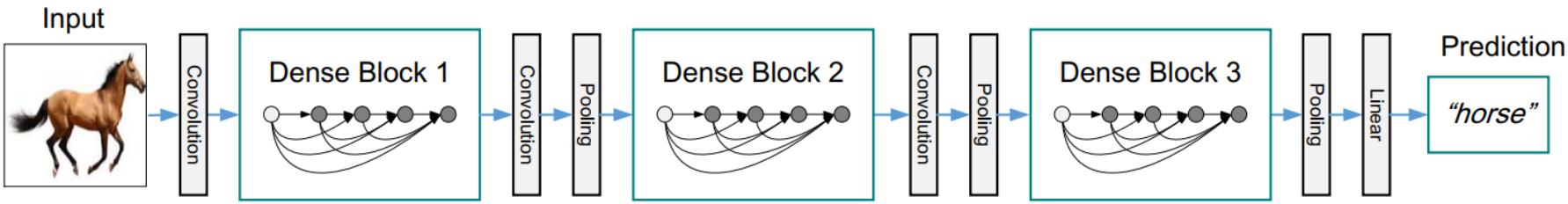
However, DenseNet uses the concatenation of outputs of all previous layers at every layer:

$$x_\ell = \mathcal{F}_\ell\left([x_0, x_1, \ldots, x_{\ell-1}]^\top\right). \tag{6}$$

In other words, it uses direct connections from each layer to all subsequent layers.

DenseNet

- DenseNet is composed of several **dense blocks** where in each dense block, every layer is connected to all its subsequent layers.
- There are **transition layers** between the dense blocks which perform convolution followed by pooling.
- In the dense blocks, the composite function $\mathcal{F}_\ell(x)$ specifically consists of batch normalization [16], 3×3 convolution, and ReLU activation function [10].
- Every transition layer consists of batch normalization [16], 1×1 convolution, and average pooling with width 2. The last transition layer, however, is an average pooling followed by a linear layer (with softmax activation function) whose number of neurons is the number of classes.
- In the dense block, every convolution has some number of channels. This number of channels is named **growth rate** in the paper of DenseNet [6]. The ℓ-th layer in a dense block has $k_0 + k(\ell - 1)$ input feature maps where k_0 is the number of channels in the input data and k is the growth rate in all previous layers inside the block.
- It is empirically observed that a small growth rate (e.g., $k = 12$) is sufficient for DenseNet [6].



References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

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

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241, Springer, 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[6] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

References (cont.)

[7] 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 computer vision*, vol. 115, pp. 211–252, 2015.

[8] K. Crowston, "Amazon mechanical turk: A research tool for organizations and information systems scholars," in *Shaping the Future of ICT Research. Methods and Approaches: IFIP WG 8.2, Working Conference, Tampa, FL, USA, December 13-14, 2012. Proceedings*, pp. 210–221, Springer, 2012.

[9] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *International journal of computer vision*, vol. 88, pp. 303–338, 2010.

[10] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[12] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning: Data Mining, Inference, and Prediction*, vol. 2.
Springer series in statistics, New York, NY, USA, 2009.

References (cont.)

[13] R. Tibshirani, M. Wainwright, and T. Hastie, *Statistical learning with sparsity: the lasso and generalizations*.
Chapman and Hall/CRC, 2015.

[14] P. Domingos, "The role of Occam's razor in knowledge discovery," *Data mining and knowledge discovery*, vol. 3, no. 4, pp. 409–425, 1999.

[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 675–678, 2014.

[16] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, pmlr, 2015.