

Convolutional Neural Network (CNN): Basics and Recent Advancements

Mohammad Pirhooshyaran and Mertcan Yetkin

Department of Industrial and Systems Engineering
Lehigh University

OptML meetings
18 September 2019

1 Neural Network basics

- Definition
- Computer vision tasks handled via CNN

2 What is convolution?

- The mathematical definition of convolution
- What is convolutional layer?

3 Recent developments

- GoogleNet
- VGG nets
- Highway networks
- ResNet
- DenseNet

4 Open questions

Neural networks

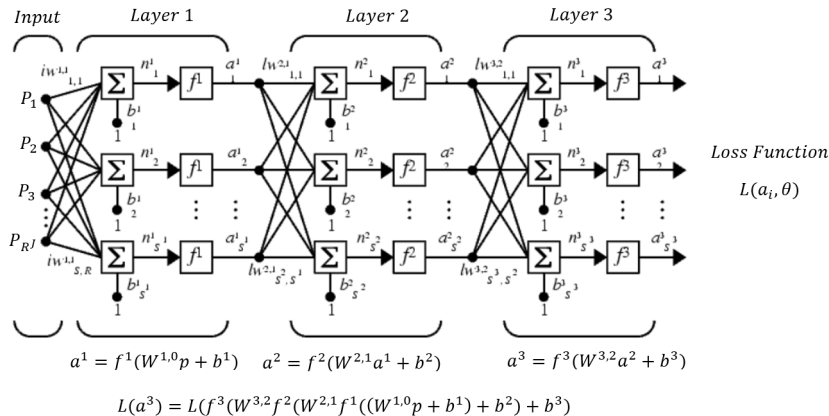
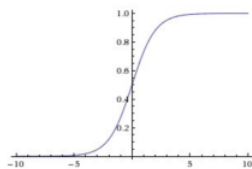


Figure: Deep Neural Networks. Taken from [what is deep learning](#)

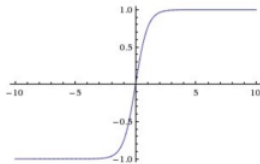
- Design of the network:
 - How many nodes should we use in first or last layer?
 - How many nodes should we use in hidden layers?
- Forward-propagation & get loss
- Backward-propagation
 - Differentiate using chain rule.
- Update the parameters using the gradient

Different types of activation

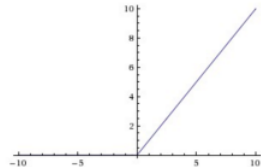
- Activation functions. Why should we use activation?



Sigmoid



tanh



ReLU

Figure: Activation functions

Computer vision tasks handled via CNN

- Localization
- Object detection
- Human pose estimation
- Other recognition tasks
- Object tracking tasks

Convolution definition

- Convolution is a mathematical **operation** on two functions f and g expressing how the **shape** of one is modified by the other.

$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(t - \tau)g(\tau) d\tau$$

- For functions f, g supported on only $[0, \infty)$ (zero for negative arguments)

$$(f * g)(t) = \int_0^t f(\tau)g(t - \tau) d\tau \quad \text{for } f, g : [0, \infty) \rightarrow \mathbb{R}$$

Convolution definition ctd..

- For complex-valued functions f , g defined on the set Z of integers, the discrete convolution of f and g is given by:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n - m]$$

- When g has finite support in the set $\{-M, -M + 1, \dots, M - 1, M\}$

$$(f * g)[n] = \sum_{m=-M}^M f[n - m]g[m]$$

- At any point n convolution is the **summation of element-wise product** of the function values.

Convolution definition ctd..

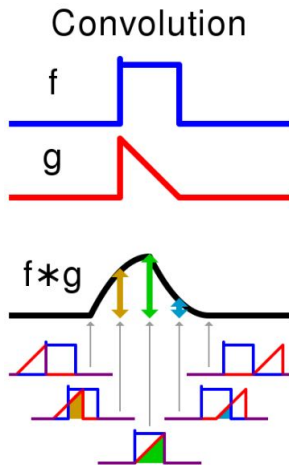


Figure: Convolution of f and g functions

Convolutional filter (kernel)

- In image processing, a kernel, convolution matrix, or mask is a small matrix.
- It is used for blurring, sharpening, embossing, edge detection, and more.
- This is accomplished by doing a convolution between a kernel and an image. $g(x, y) = \omega * f(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b \omega(s, t) f(x - s, y - t)$, where $g(x, y)$ is the filtered image, $f(x, y)$ is the original image, ω is the filter kernel.
- Every element of the filter kernel is considered by $-a \leq s \leq a$ and $-b \leq t \leq b$.

Images in computer

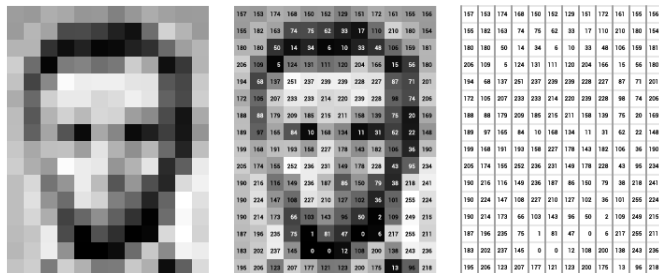


Figure: matrix (or vectors) of numbers

Images in computer ctd..

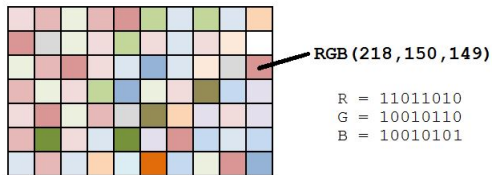


Figure: Each number represents the intensity of the colors in that location (pixel)

Convolution filters (kernels) operations

Convolution is the **summation** of **element-wise product** of the function values.

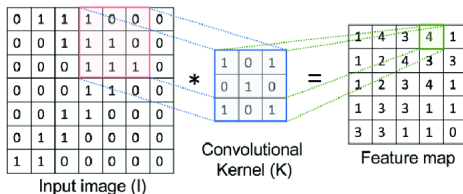


Figure: Convolution filter

Some convolution filters (kernels)





Operation	Kernel ω	Image result $g(x,y)$
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Box blur	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Gaussian blur 3×3	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5×5	$\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

Figure: Different kernels for different purposes

- If the summation of the filter elements adds up to one means the intensity of the image in general stays the same.
- If we infinitely many use the filter blur we make the intensity of pixel getting closer to each other.

Some more filters



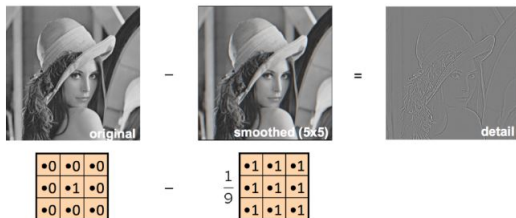
<u>Edge detection</u>	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

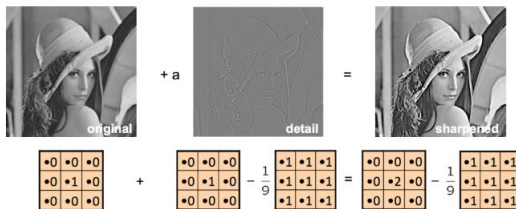
Figure: Edge detection

Creating new filters

Step 1: Original - Smoothed = "Details"



Step 2: Original + "Details" = Sharpened



What is convolutional layer? (example with AlexNet [1])

- A layer in which we use multiple kernels (filters, masks) is a convolutional layer.
- Among the previous mentioned filters, **which one** should we choose for **neural networks**?

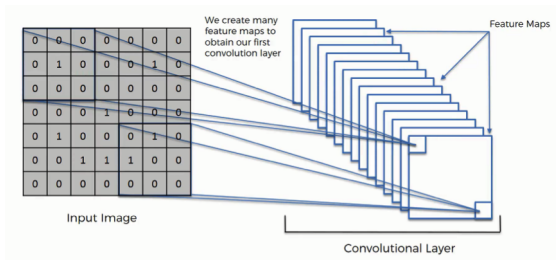


Figure: Convolution layer

Network parameters

- Networks weights are the filter numbers.
- We initialize them with normal (Gaussian) values. (This helps each filter ends up learning different aspect of the image)
- Number of the filters stack up with each-other to build one layer.

Stride & Padding

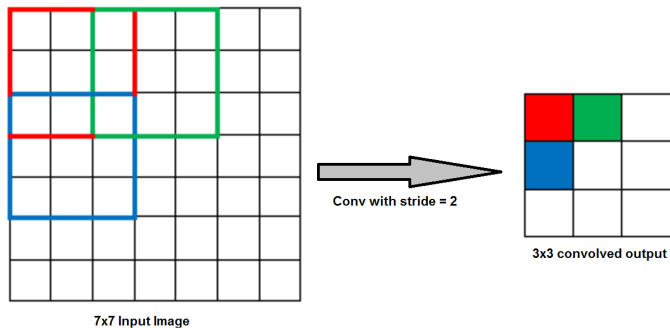


Figure: Stride

Stride & Padding

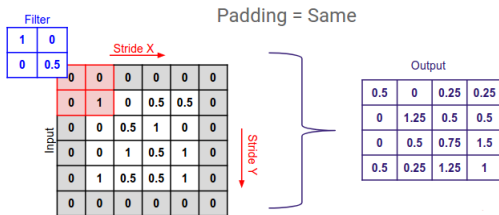
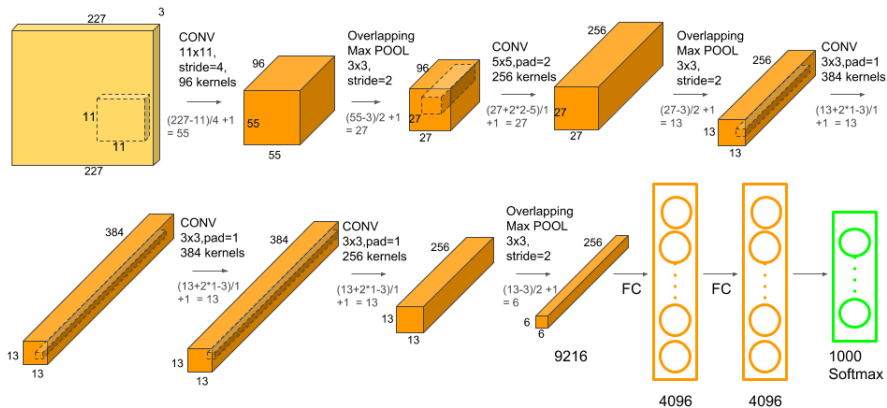


Figure: Padding

AlexNet [1]



- Each Channel has their own filter. (Average them up)
- 96,256,384,384, are the number of features in each filter
- So, far all we said was linear summation. Element-wise Activation (First time ReLU). (6 times faster than sigmoid).
- Counting the number of parameters
First Conv layer: $11 \times 11 \times 3 \times 96 = 34.848$
First Fully-connected Layer: $9216 \times 4096 \cong 37m$.
Total: 62.3 million = ? **conv parameters**
- Computation time comparison.
- Softmax is a function to give the probability pf selection. (1000 is the number of classes in ImageNet dataset)
- Max-pooling:

Max-pooling

- A way to summarize the information of adjacent pixels into one pixel.
- For size reduction.
- It is a cheap layer (Network do not tune any parameter)

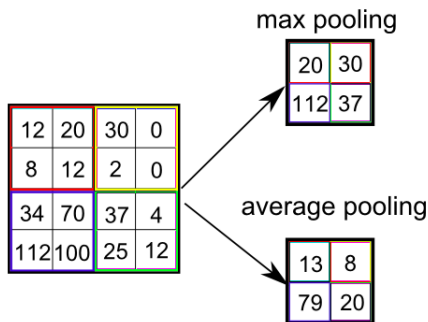


Figure: max & average pooling

Bottleneck layer

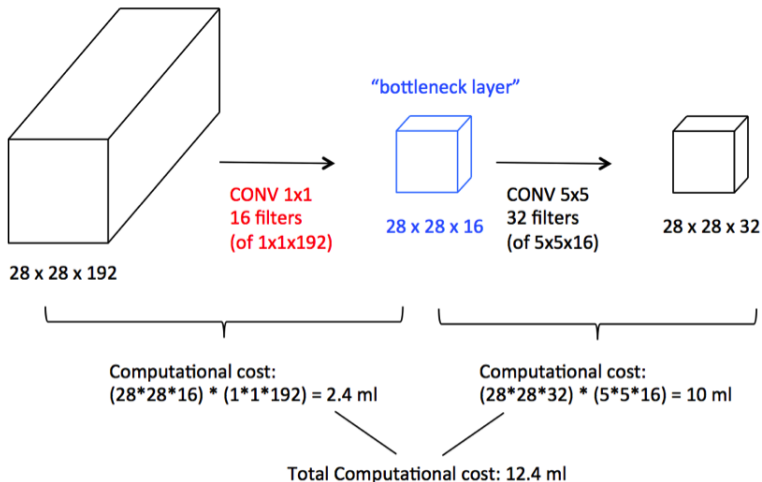


Figure: Bottleneck layer example

Filters Learned

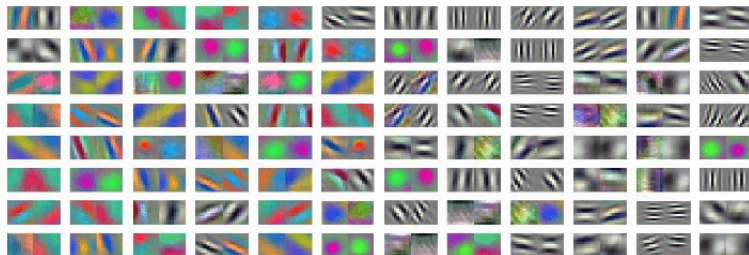


Figure: Learned filters for Conv1 layer of Alex-Net

Being able to **learn** the values of **different filters**, CNNs can find more meaning from images that humans and **human designed filters** might not be able to find.

Filters learned

- Filters in a convolutional layer learn to detect abstract concepts, like the boundary of a face or the shoulders of a person.
- By stacking layers of convolutions on top of each other, we can get more abstract and in-depth information from a CNN.

Human Vision vs Computer Vision

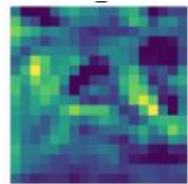


Figure: What is it? A cloud, a zombie, a raccoon or a horse?

Human Vision vs Computer Vision ctd..

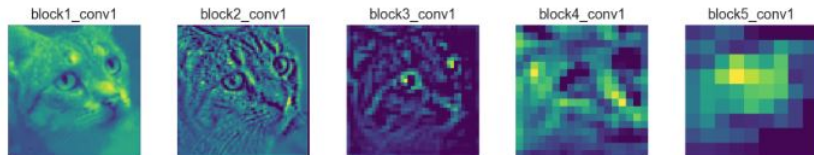


Figure: output results of conv layers

- As we go deeper into the network, the feature maps look **less** like the original image and more like an abstract representation of it.

Two main ideas after Alex-Net success!

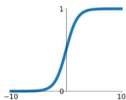
- Stacking up the layers (go deeper)! To find more abstract and deep features!
- Reduce the number of parameters used (in general reduce the training time)

Major concerns with going deeper

- First obstacle was the notorious problem of **vanishing/exploding gradients**
- Activations need to keep the gradient information.

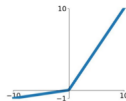
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



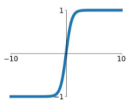
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$

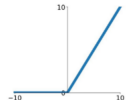


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ReLU

$$\max(0, x)$$



ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

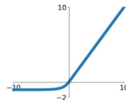


Figure: Which one can transfer the gradient better?

Vanishing gradient

- We need the gradient values to update the parameters!
- Vanishing gradient!

$$\frac{\partial L}{\partial W^{1,0}} = \frac{\partial L}{\partial a^3} \frac{\partial a^3}{\partial f^3} \frac{\partial f^3}{\partial f^2} \frac{\partial f^2}{\partial f^1} \frac{\partial f^1}{\partial W^{1,0}} W^{3,2} W^{2,1} W^{1,0}.$$

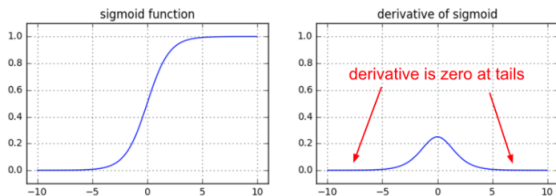


Figure: Vanishing gradient.

Batch-normalization

- We do preprocessing to the input data!
- In the intermediate layers: The distribution of the activations is constantly changing during training, the training process slows down
- Each layer must learn to adapt themselves to a new distribution in every training step.
- This problem is known as internal covariate shift.
- Batch normalization is a method to normalize the inputs of each layer, to overcome the internal covariate shift problem.

Design of the structure

- “Batch normalization” and reducing “vanishing gradient” enable networks with tens of layers to start converging for stochastic gradient descent (SGD) with back-propagation!
- When deeper networks are able to start converging, a degradation problem has been exposed!

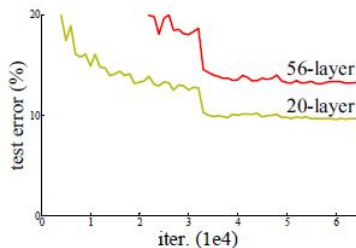


Figure: Deeper network perform inferior! Is it overfitting?

Design of the structure ctd..

- It's not due to overfitting

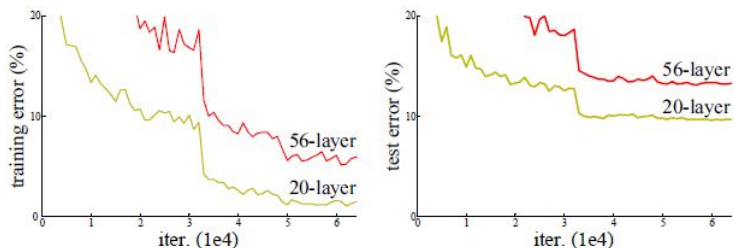


Figure: The deeper network has higher training error, and thus test error. (Train and test CIFAR-10 with 20-layer and 56-layer “plain” networks)

Major problems

- Choosing the right kernel sizes is crucial for convolution and for the overall learning process.
- Larger networks increase the chance of overfitting.
- Larger networks make the training procedure (back-propagation) difficult and accuracy will decrease (degradation problem).
- Current computational techniques are finely tuned for dense connections (easily parallelizable), thus employing sparse structures hasn't been utilized well.

- They approximate a local sparse (optimal) structure in the convolutional layers by readily available dense components.

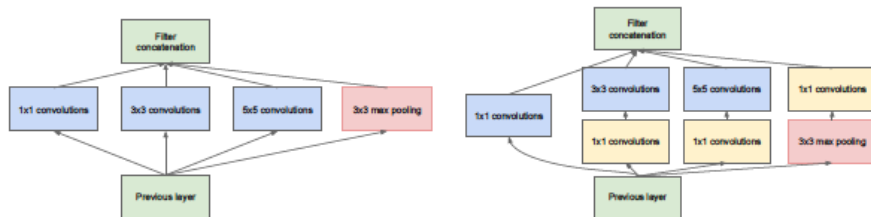


Figure: GoogleNet inception modules with dimension reduction (right)

- By considering the inception modules, they incorporate multiple kernels in a single layer in order extract more information. (Choosing right kernel size is a problem).
- They scale up the network without additional computational cost.
- They include auxiliary classifiers inside the architecture to increase the gradient signal that gets propagated back (this means another intermediary discounted loss is added to the objective).
- Trained via SGD with momentum.

- They consider small convolution filters with more convolution layers.
- This will increase the nonlinearity in the function, while still capturing similar information obtained by larger convolution filters.
- Assuming each convolution layer has C channels, and via a 3×3 filter, a layer requires $3(3^2 C^2) = 27C^2$ parameters.
- Trained via SGD (mini-batch) with momentum.

Highway networks [4]

- “Gating mechanism that allows for computation paths along which information can flow across many layers without attenuation.”
- They denote those paths as information highways.

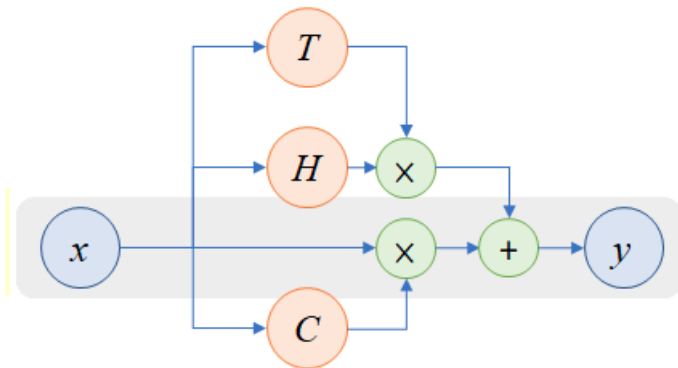


Figure: Highway circuit

Highway networks ctd..

- $y_i = H_i(x)T_i(x) + x_i(1 - T_i(x))$ where $H_i(\cdot)$ is the nonlinear transformation and $0 < T_i(\cdot) < 1$ is the transform gate in layer i .
- They use plain layers to change the dimension of $H_i(\cdot)$ to be same as the dimension of x .
- Trained via SGD with momentum.

- They consider learning the residual mapping ($F(x) = H(x) - x$) rather than learning unreferenced mapping $H(x)$.
- They utilize “shortcut connections” to pass x in every block.

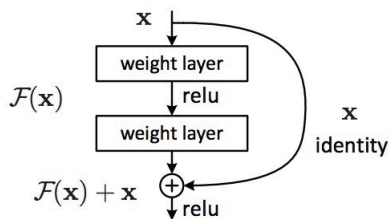


Figure: ResNet block

- Two options to match the dimensions of $F(x)$ and x :
 - Extra zero entries padded for increasing dimensions.
 - The projection shortcut is used to match dimensions.
- It is much easier to learn to push $F(x)$ to 0 and leave the output as x than to learn an identity transformation from scratch.
- Easier to optimize, and increased accuracy with deeper networks.
- Trained via SGD (mini-batch) with momentum.

- Connects each layer to every other layer in a feed-forward fashion.
 - L layers will have $\frac{L(L+1)}{2}$ direct connections.
- Main advantages can be listed as:
 - Overcoming vanishing gradient problem
 - Encourage feature reuse
 - Reduce the number of parameters (**How??**)
- Utilizes the idea of creating short paths from early layers to later layers (as done in ResNet, Highway networks).
 - Instead of summation, it combines different features maps via concatenation.

DenseNet structure

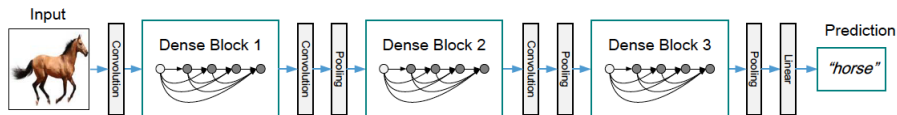


Figure: DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

- They use bottleneck layers and compression (referred as BC) in the transition layers.
 - Transition layers generate $\lfloor \theta m \rfloor$ output feature-maps, where m is the number of feature maps inside a block, and $0 < \theta \leq 1$ is the compression factor.

Some observations:

- Partly inspired by the idea of stochastic depth.
- There is no need to learn redundant feature-maps.
- So each layer has few filters (growth rate k).
- Implicit deep supervision is observed (via shorter connections to the classifier).
- They observe that dense connections have a regularizing effect.
- Trained via SGD (mini-batch) with momentum.

Some results

Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network [22]	-	-	10.41	8.81	35.68	-	2.35
All-CNN [31]	-	-	9.08	7.25	-	33.71	-
Deeply Supervised Net [20]	-	-	9.69	7.97	-	34.57	1.92
Highway Network [33]	-	-	-	7.72	-	32.39	-
FractalNet [17]	21	38.6M	10.18	5.22	35.34	23.30	2.01
with Dropout/Drop-path	21	38.6M	7.33	4.60	28.20	23.73	1.87
ResNet [11]	110	1.7M	-	6.61	-	-	-
ResNet (reported by [13])	110	1.7M	13.63	6.41	44.74	27.22	2.01
ResNet with Stochastic Depth [13]	110	1.7M	11.66	5.23	37.80	24.58	1.75
	1202	10.2M	-	4.91	-	-	-
Wide ResNet [41]	16	11.0M	-	4.81	-	22.07	-
	28	36.5M	-	4.17	-	20.50	-
with Dropout	16	2.7M	-	-	-	-	1.64
ResNet (pre-activation) [12]	164	1.7M	11.26*	5.46	35.58*	24.33	-
	1001	10.2M	10.56*	4.62	33.47*	22.71	-
DenseNet ($k = 12$)	40	1.0M	7.00	5.24	27.55	24.42	1.79
DenseNet ($k = 12$)	100	7.0M	5.77	4.10	23.79	20.20	1.67
DenseNet ($k = 24$)	100	27.2M	5.83	3.74	23.42	19.25	1.59
DenseNet-BC ($k = 12$)	100	0.8M	5.92	4.51	24.15	22.27	1.76
DenseNet-BC ($k = 24$)	250	15.3M	5.19	3.62	19.64	17.60	1.74
DenseNet-BC ($k = 40$)	190	25.6M	-	3.46	-	17.18	-

Figure: Error rates on CIFAR and SVHN (Street View House Numbers) datasets. “+” indicates standard data augmentation (translation and/or mirroring). “*” indicates results run by the authors.

Feature reuse for DenseNet on C10+ dataset with $L = 40$ and $k = 12$

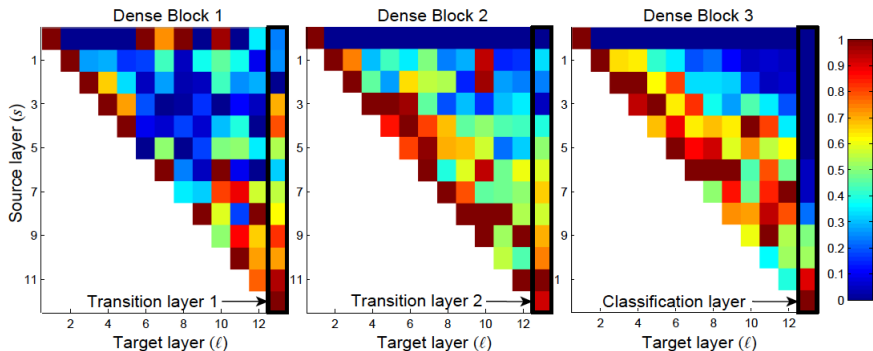




Figure: The color of pixel (s, l) encodes the average L1 norm (normalized by number of input feature-maps) of the weights connecting convolutional layer s to l within a dense block. Three columns highlighted by black rectangles correspond to two transition layers and the classification layer. The first row encodes weights connected to the input layer of the dense block.

Open questions

- Different architectures
 - Kernel sizes
 - Connection types
- Different activation functions (need to classify the required properties)
- Different operators

References I

-  Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton.
Imagenet classification with deep convolutional neural networks.
In *Advances in neural information processing systems*, pages 1097–1105, 2012.
-  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*, pages 1–9, 2015.
-  Karen Simonyan and Andrew Zisserman.
Very deep convolutional networks for large-scale image recognition.
arXiv preprint arXiv:1409.1556, 2014.



Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber.
Training very deep networks.
In *Advances in neural information processing systems*, pages 2377–2385, 2015.



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*, pages 770–778, 2016.



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*, pages 4700–4708, 2017.