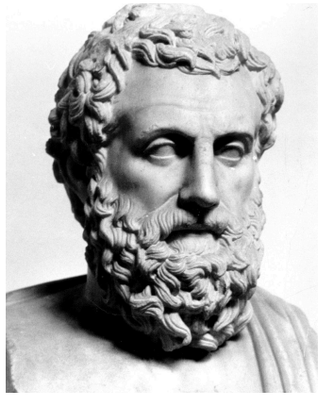
Artificial Intelligence For NLP Lesson- 12

人工智能与自然语言处理课程组

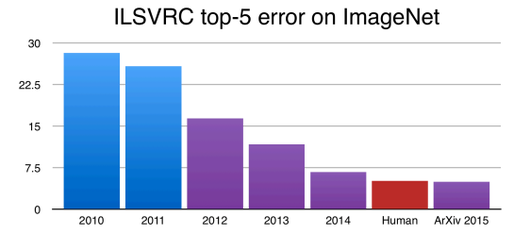
2019.Sept. 21

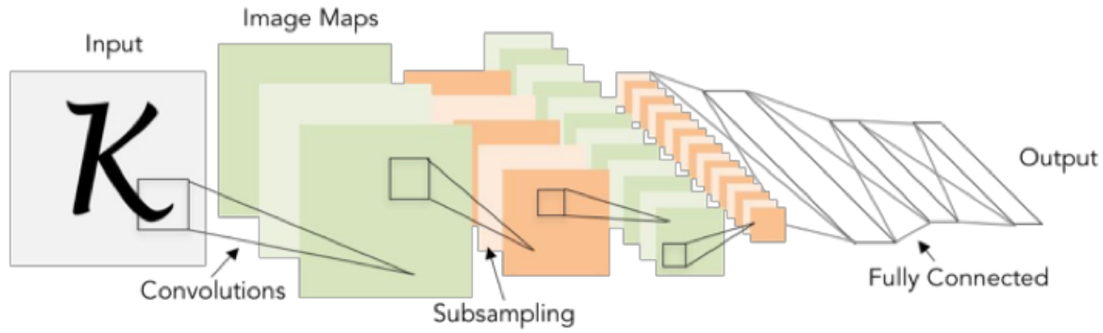


Outline

- 1. Project II
- 2. Convolutional Neural Networks
- 3. Pooling, Batch Normalization
- 4. Transfer Learning

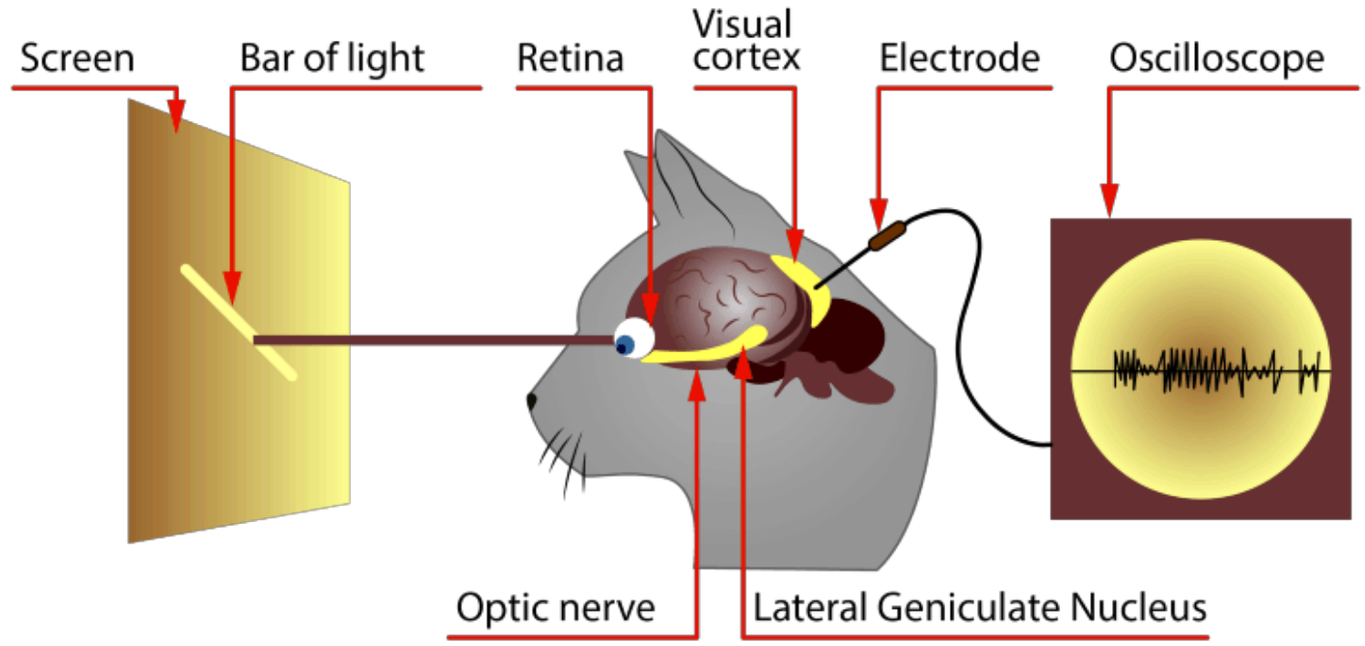
Any differences with the previous Neural Networks?



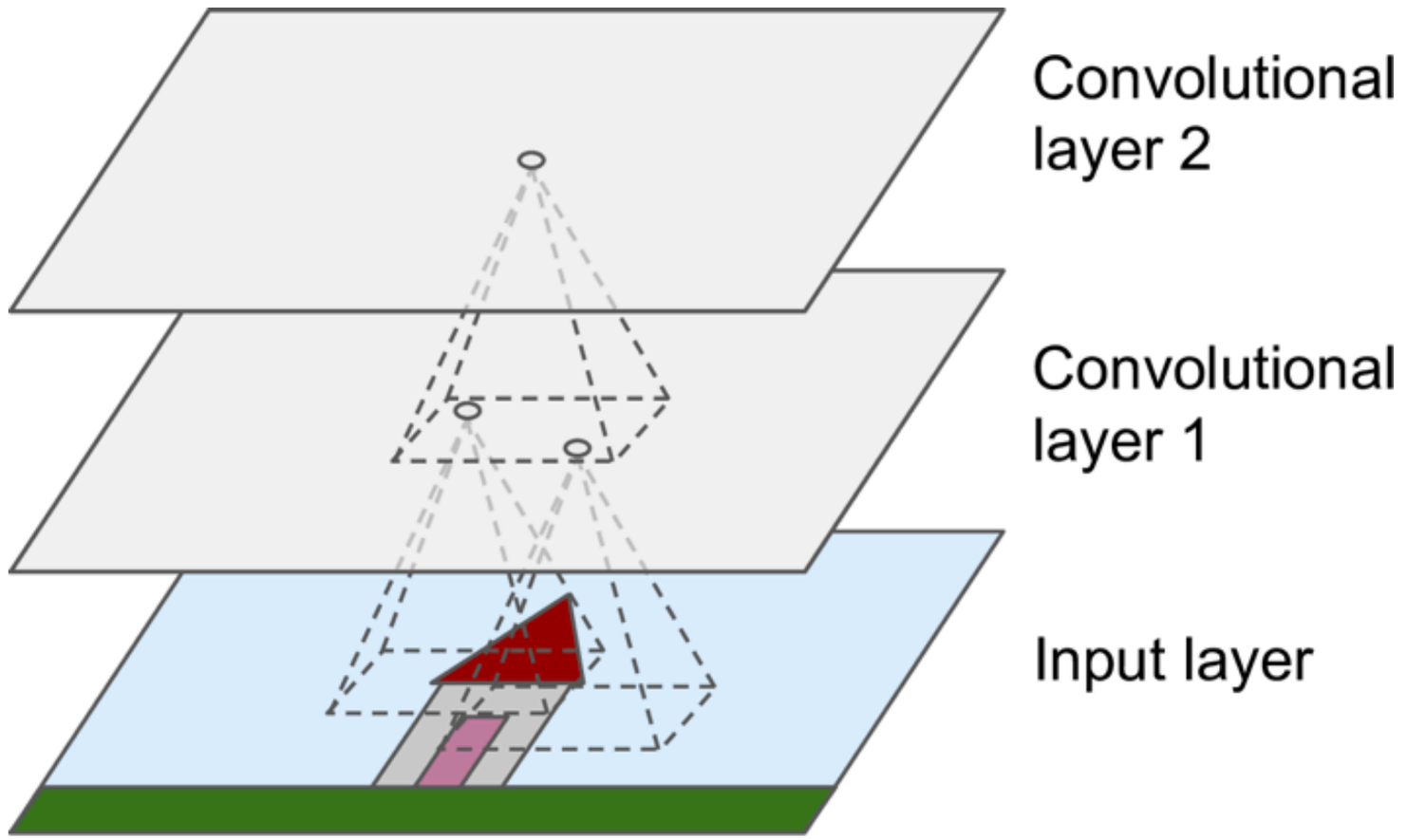


- 1960s, Harvard

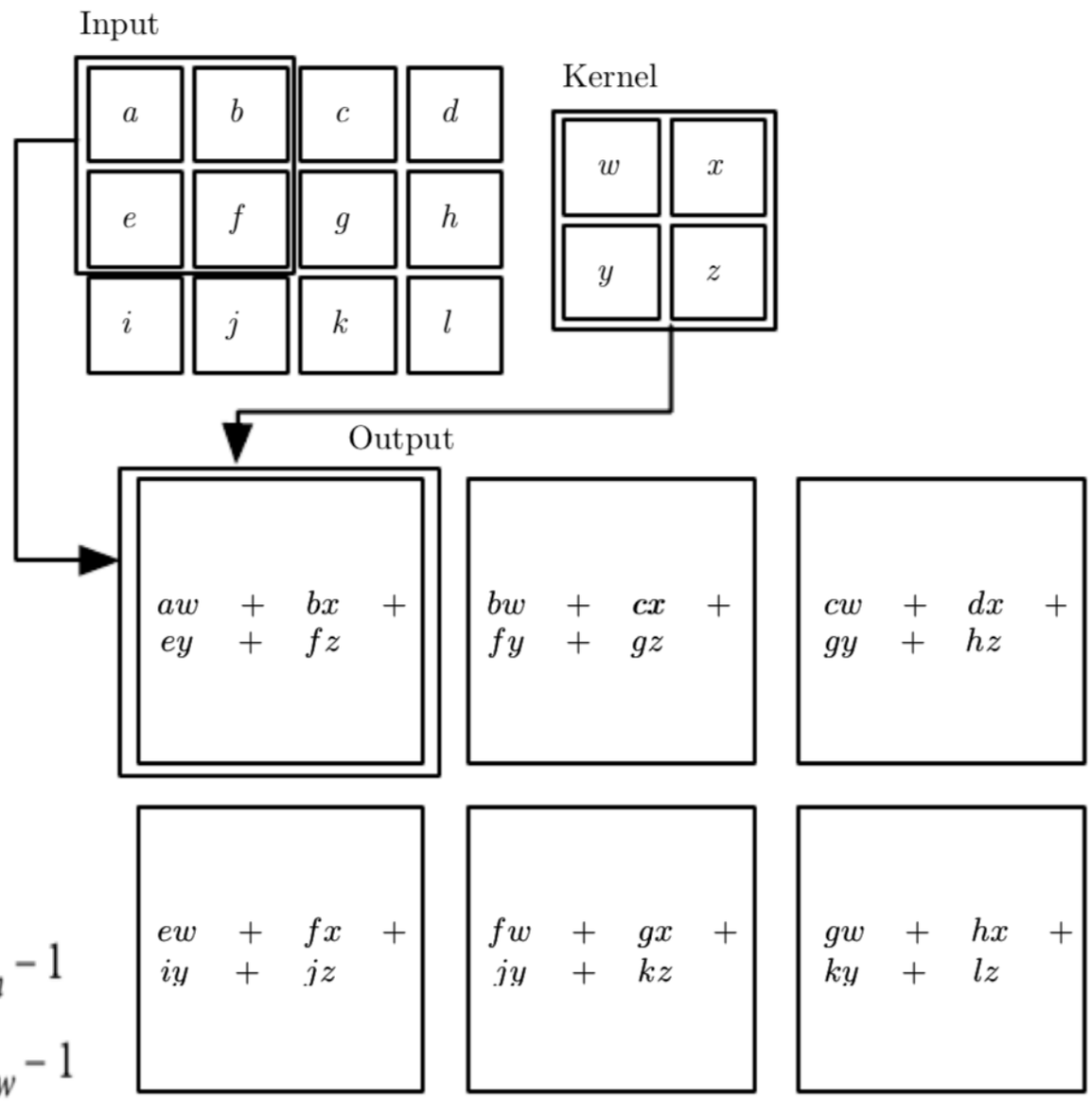
- Layers (hierarchy)
- Abstraction
- Local (spatial)



- Local Invariant

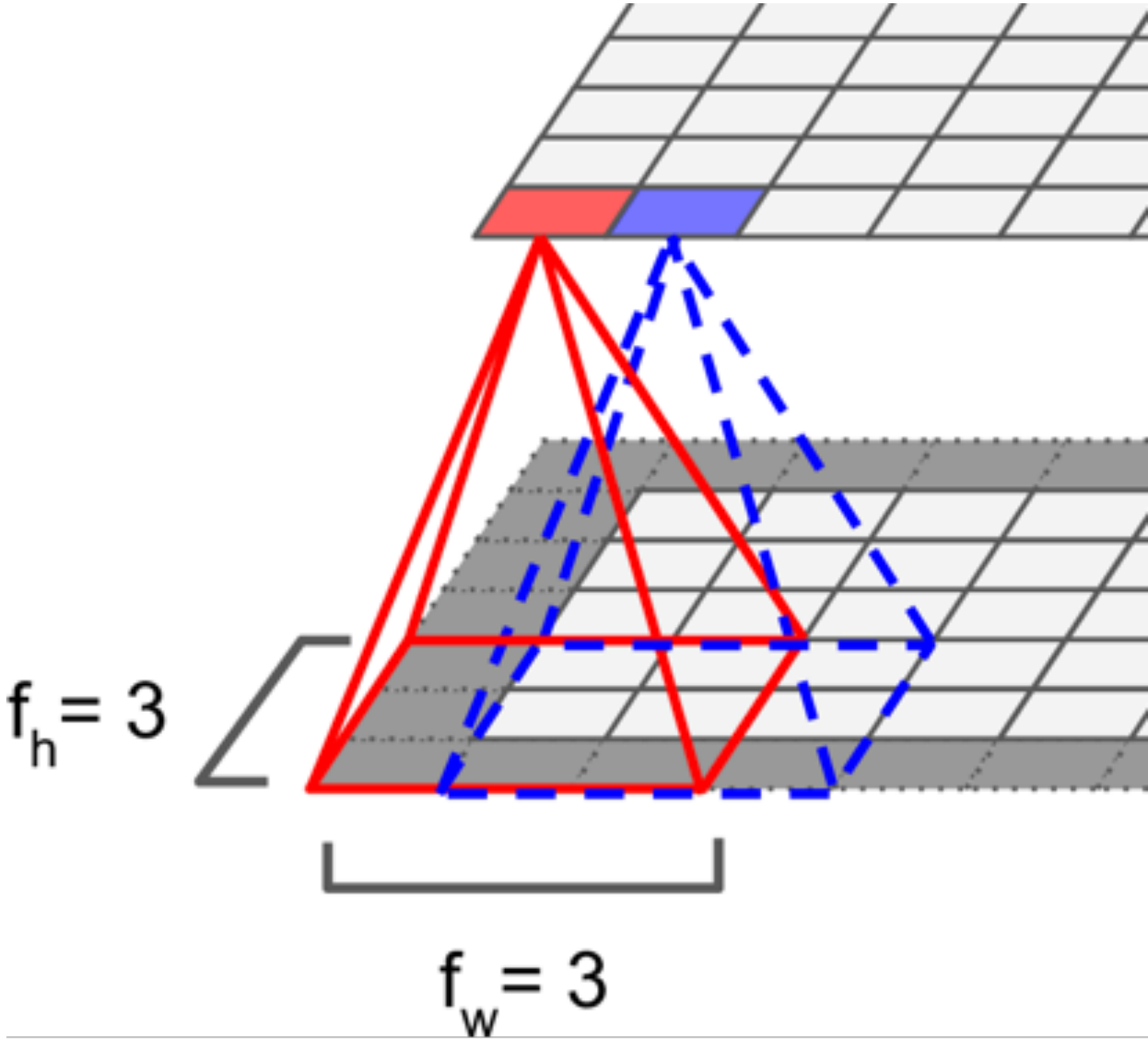


Parameters sharing

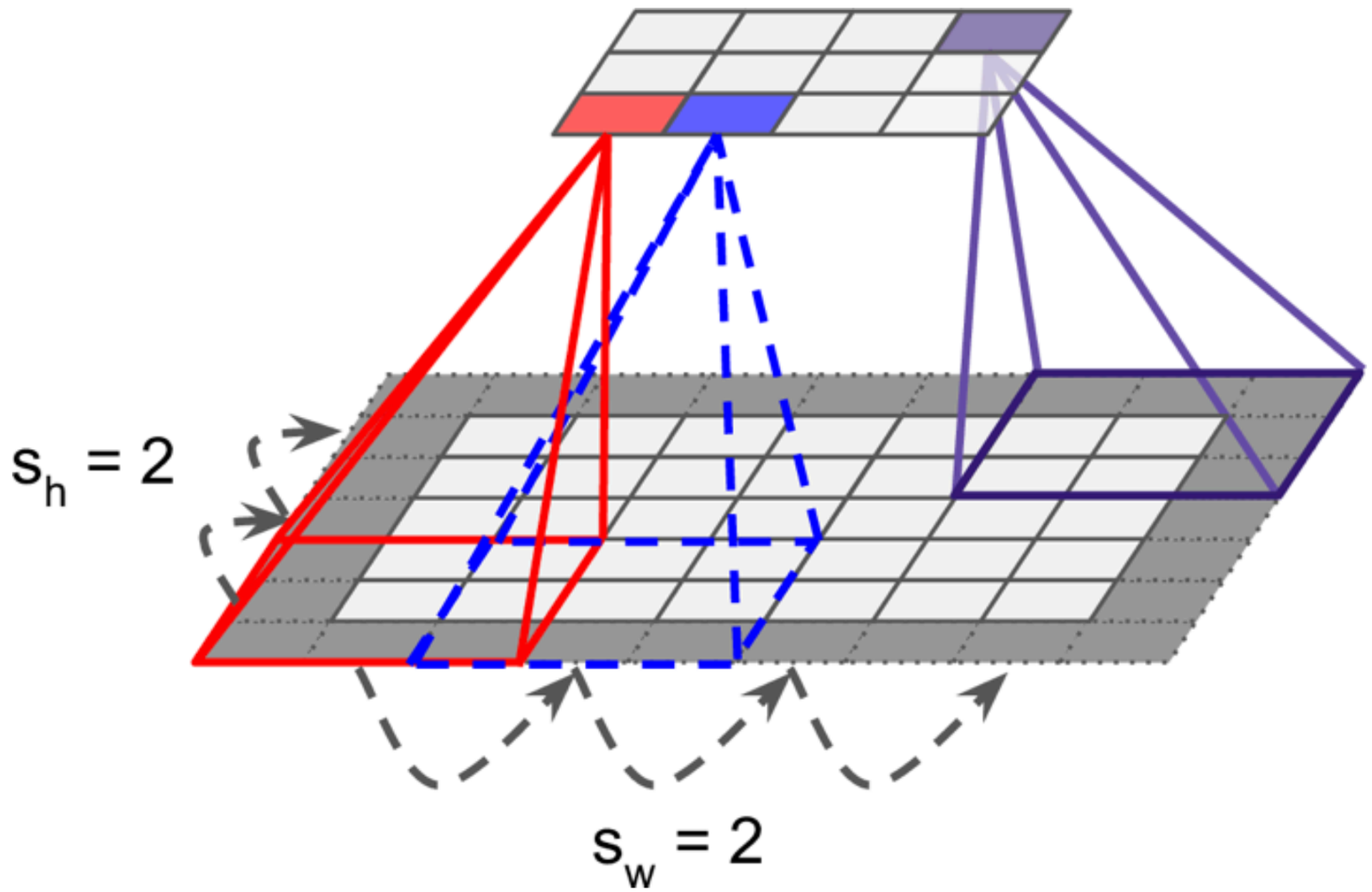


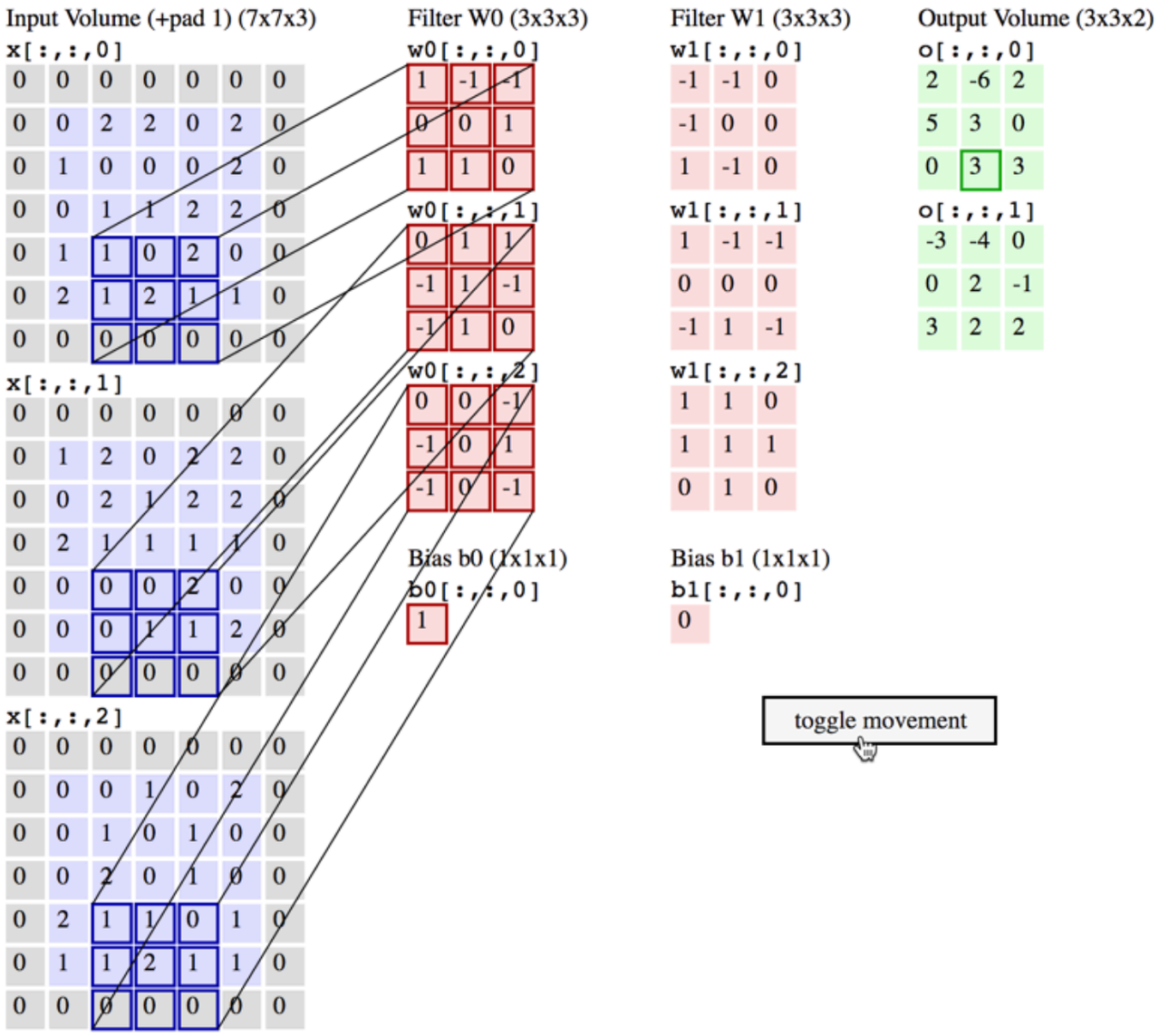
$$z_{i,j,k} = b_k + \sum_{u=1}^{f_h} \sum_{v=1}^{f_w} \sum_{k'=1}^{f_{n'}} x_{i',j',k'} \cdot w_{u,v,k',k} \quad \text{with} \begin{cases} i' = u \cdot s_h + f_h - 1 \\ j' = v \cdot s_w + f_w - 1 \end{cases}$$

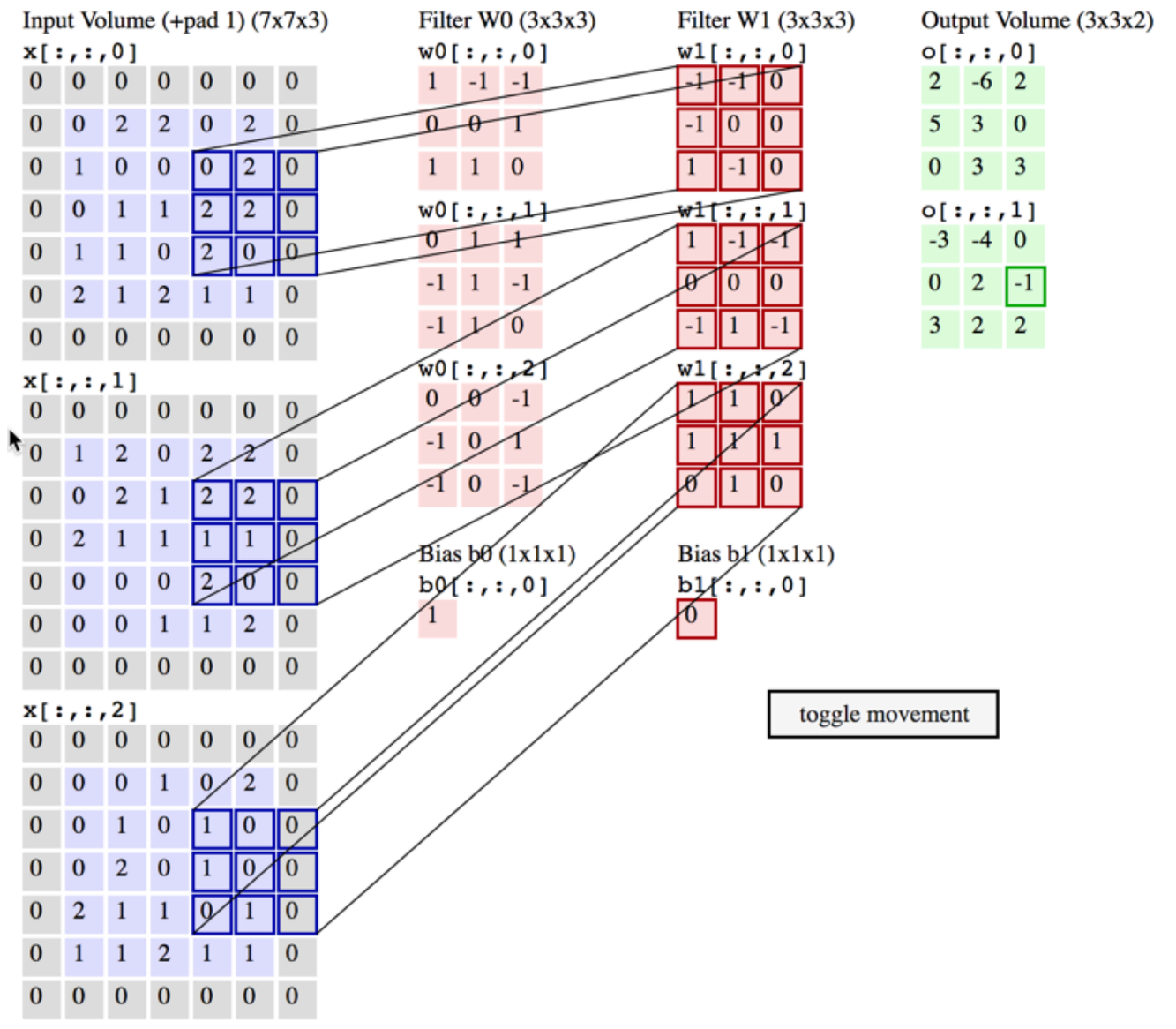
- window
- filter
- strides
- padding
  - In order for a layer to have the same height and width as the previous layer, it is common to add zeros around the inputs. so-called *Zero Padding*
  - *Valid Padding do not add extra padding in the input layers.*



Move quicker

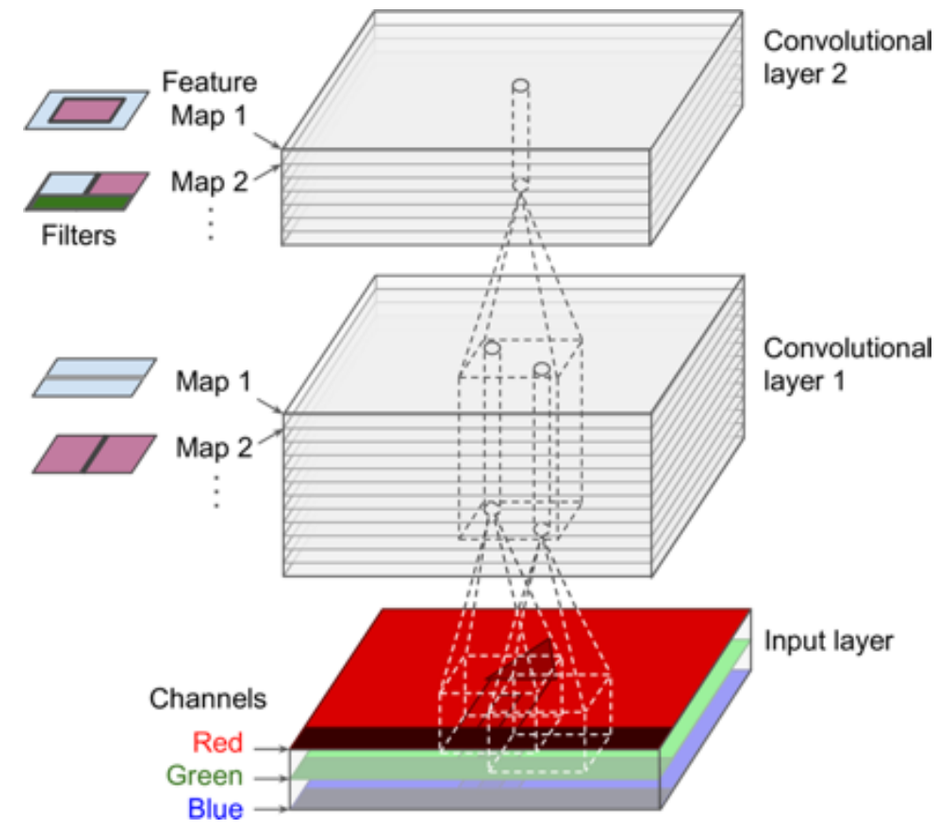


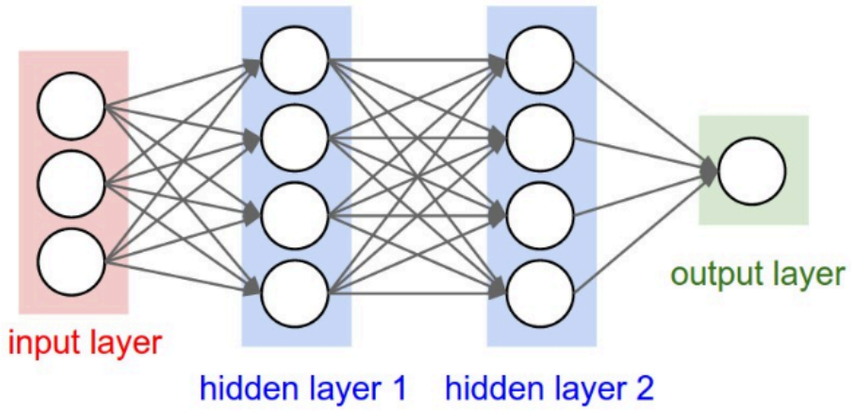

Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 0 2 2 0 2 0
0 1 0 0 0 2 0
0 0 1 1 2 2 0
0 1 1 0 2 0 0
0 2 1 2 1 1 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 2 0 2 2 0
0 0 2 1 2 2 0
0 2 1 1 1 1 0
0 0 0 0 2 0 0
0 0 0 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 1 0 2 0
0 0 1 0 1 0 0
0 0 2 0 1 0 0
0 2 1 1 0 1 0
0 1 1 2 1 1 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
1 -1 -1
0 0 1
1 1 0
w0[:,:,1]
0 1 1
-1 1 -1
-1 1 0
w0[:,:,2]
0 0 -1
-1 0 1
-1 0 -1
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 -1 0
-1 0 0
1 -1 0
w1[:,:,1]
1 -1 -1
0 0 0
-1 1 -1
w1[:,:,2]
1 1 0
1 1 1
0 1 0
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
2 -6 2
5 3 0
0 3 3
o[:,:,1]
-3 -4 0
0 2 -1
3 2 2
toggle movement

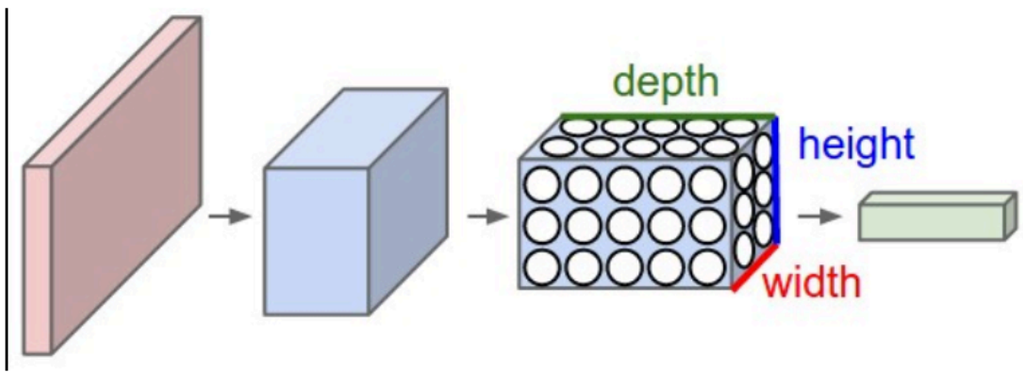

Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 0 2 2 0 2 0
0 1 0 0 0 2 0
0 0 1 1 2 2 0
0 1 1 0 2 0 0
0 2 1 2 1 1 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 2 0 2 2 0
0 0 2 1 2 2 0
0 2 1 1 1 1 0
0 0 0 0 2 0 0
0 0 0 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 1 0 2 0
0 0 1 0 1 0 0
0 0 2 0 1 0 0
0 2 1 1 0 1 0
0 1 1 2 1 1 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
1 -1 -1
0 0 1
1 1 0
w0[:,:,1]
0 1 1
-1 1 -1
-1 1 0
w0[:,:,2]
0 0 -1
-1 0 1
-1 0 -1
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 -1 0
-1 0 0
1 -1 0
w1[:,:,1]
1 -1 -1
0 0 0
-1 1 -1
w1[:,:,2]
1 1 0
1 1 1
0 1 0
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
2 -6 2
5 3 0
0 3 3
o[:,:,1]
-3 -4 0
0 2 -1
3 2 2
toggle movement

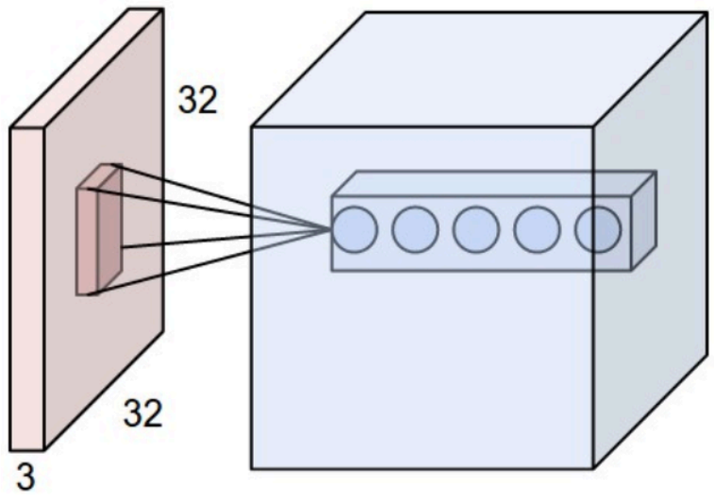
- 1. There is 3 dimensions stride, width, height, deep, but few.
- 2. Convolutional Operation could be over and over again.

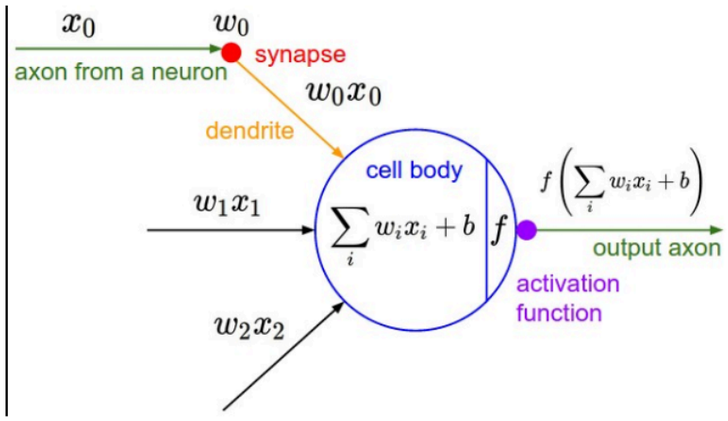
$$z_{i,j,k} = b_k + \sum_{u=1}^{f_h} \sum_{v=1}^{f_w} \sum_{k'=1}^{f_{n'}} x_{i',j',k'} \cdot w_{u,v,k',k} \quad \text{with} \begin{cases} i' = u \cdot s_h + f_h - 1 \\ j' = v \cdot s_w + f_w - 1 \end{cases}$$

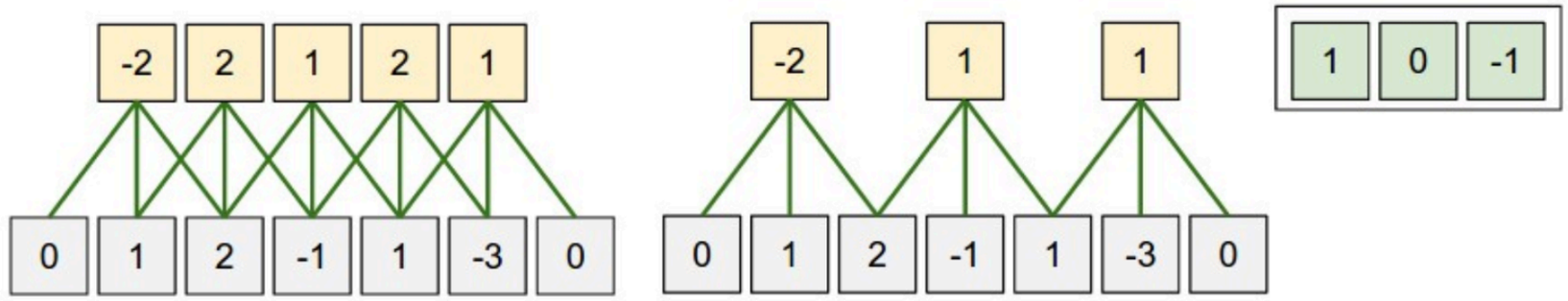



input layer
hidden layer 1
hidden layer 2
output layer


depth
height
width

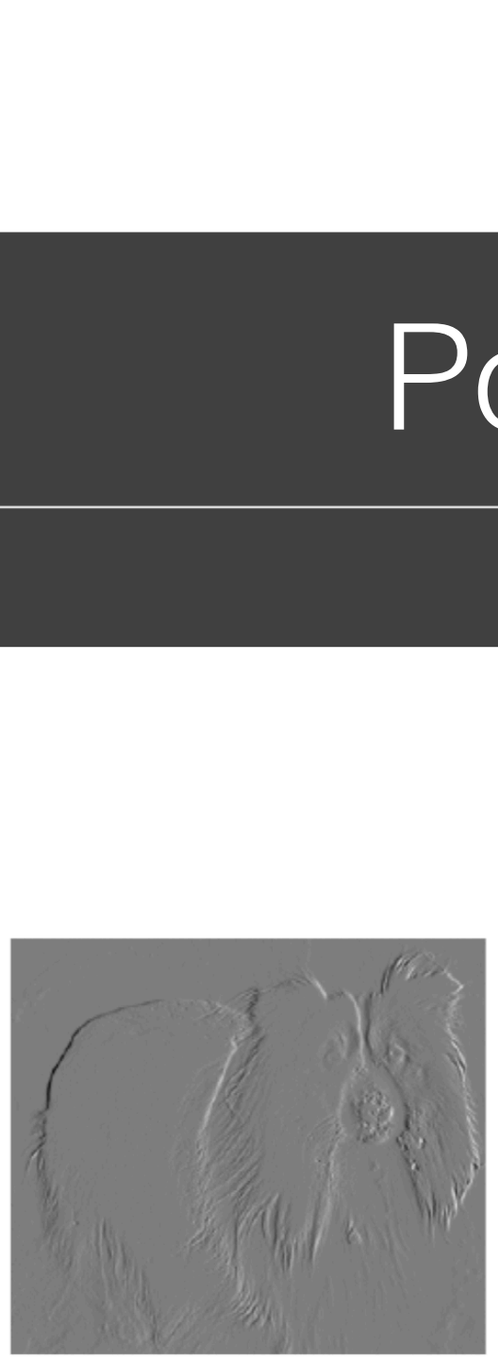

32
32
3

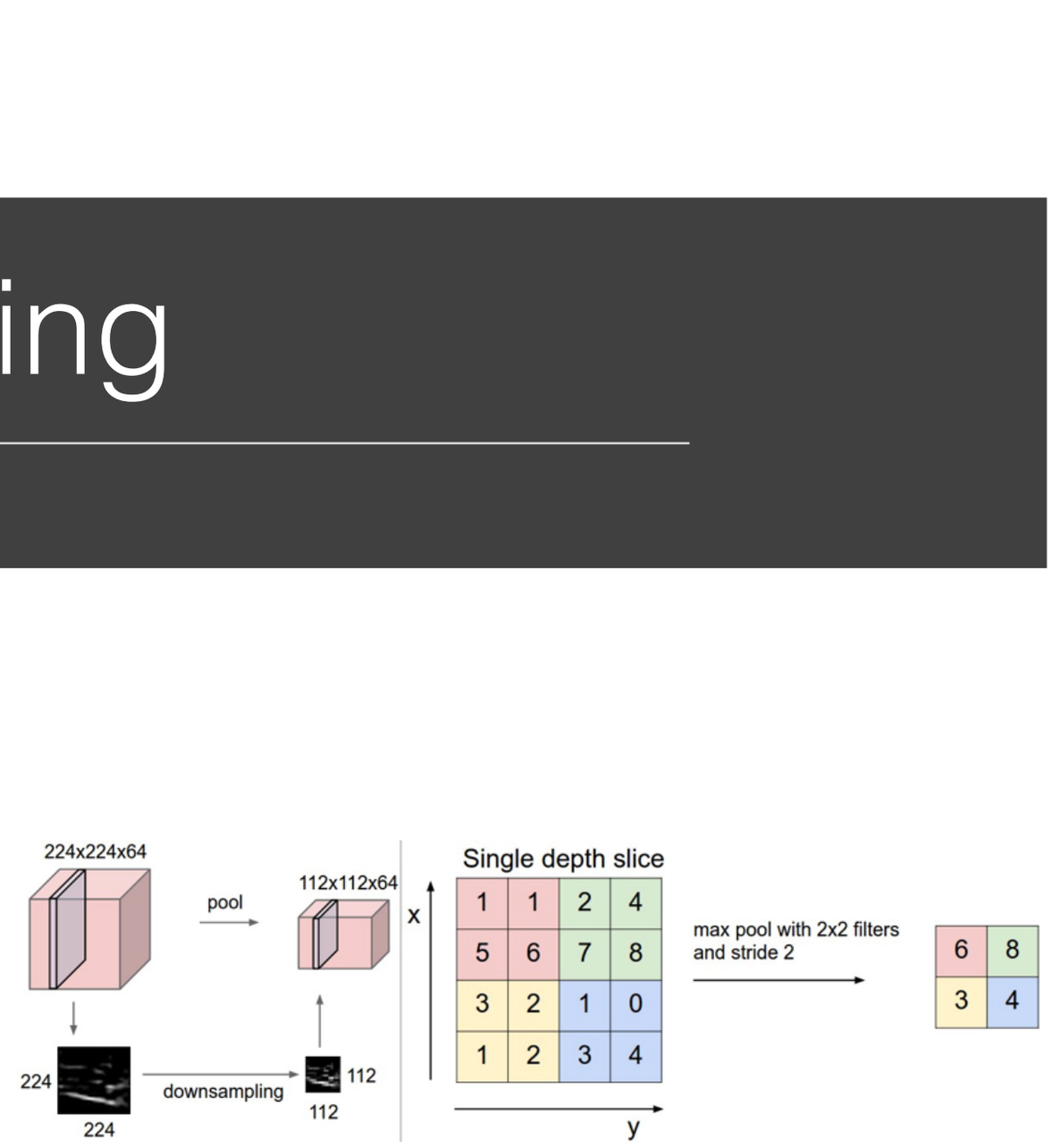

x_0
w_0
synapse
axon from a neuron
$w_0 x_0$
dendrite
cell body
$w_1 x_1$
$\sum_i w_i x_i + b$
f
$f\left(\sum_i w_i x_i + b\right)$
output axon
activation function
$w_2 x_2$



- Accepts a volume of size W1×H1×D1
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size W2×H2×D2 where:
  - W2=(W1−F+2P)/S+1
  - H2=(H1−F+2P)/S+1
  - (i.e. width and height are computed equally by symmetry)
  - D2=K

Pooling

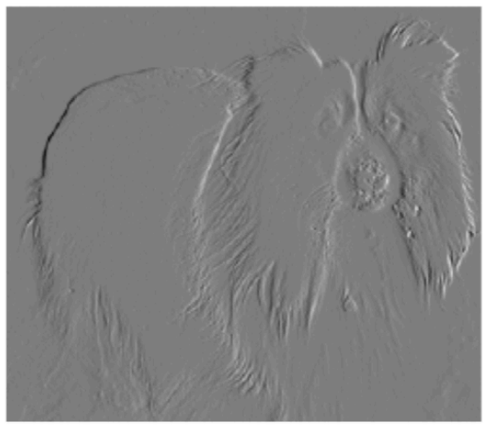




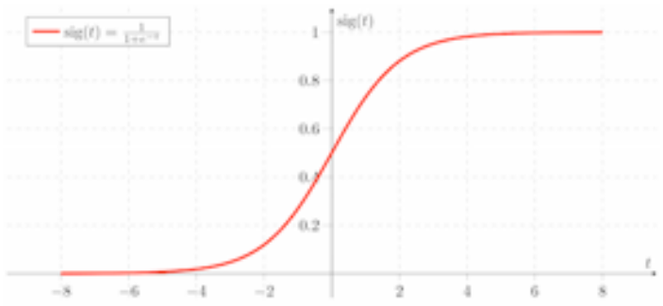
reduce the spatial size of parameters,
also control overfitting!

Max Pooling, Average Pooling
Input: W1 * H1 * D1
Output: ((W1 - F) / S + 1, (H1 - F)/S + 1, D1)

- Can we do not use Pooling Layers?
  - Answer: could use just repeated CONV Net.
  - reduce size of parameters just use larger stride in



Xavier Initialization



- Start too small, then the signal shrinks as it passes through each layer until too tiny to use;
- Start too large, then the signal grows as it passes through each layer until too massive to use.
- https://keras.io/initializers/

Batch Normalization

- Batch Normalization allows us to use much higher learning rates and be less careful about i

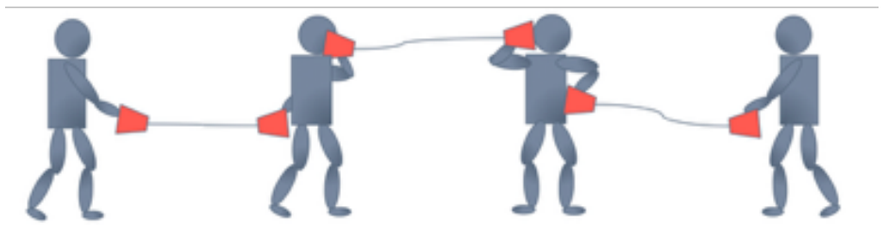
Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

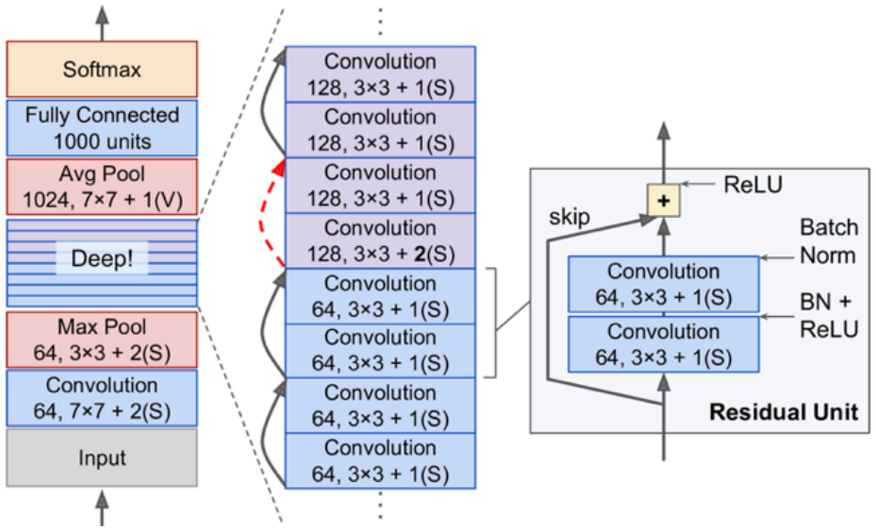
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$



- cite: https://arxiv.org/pdf/1502.03167.pdf

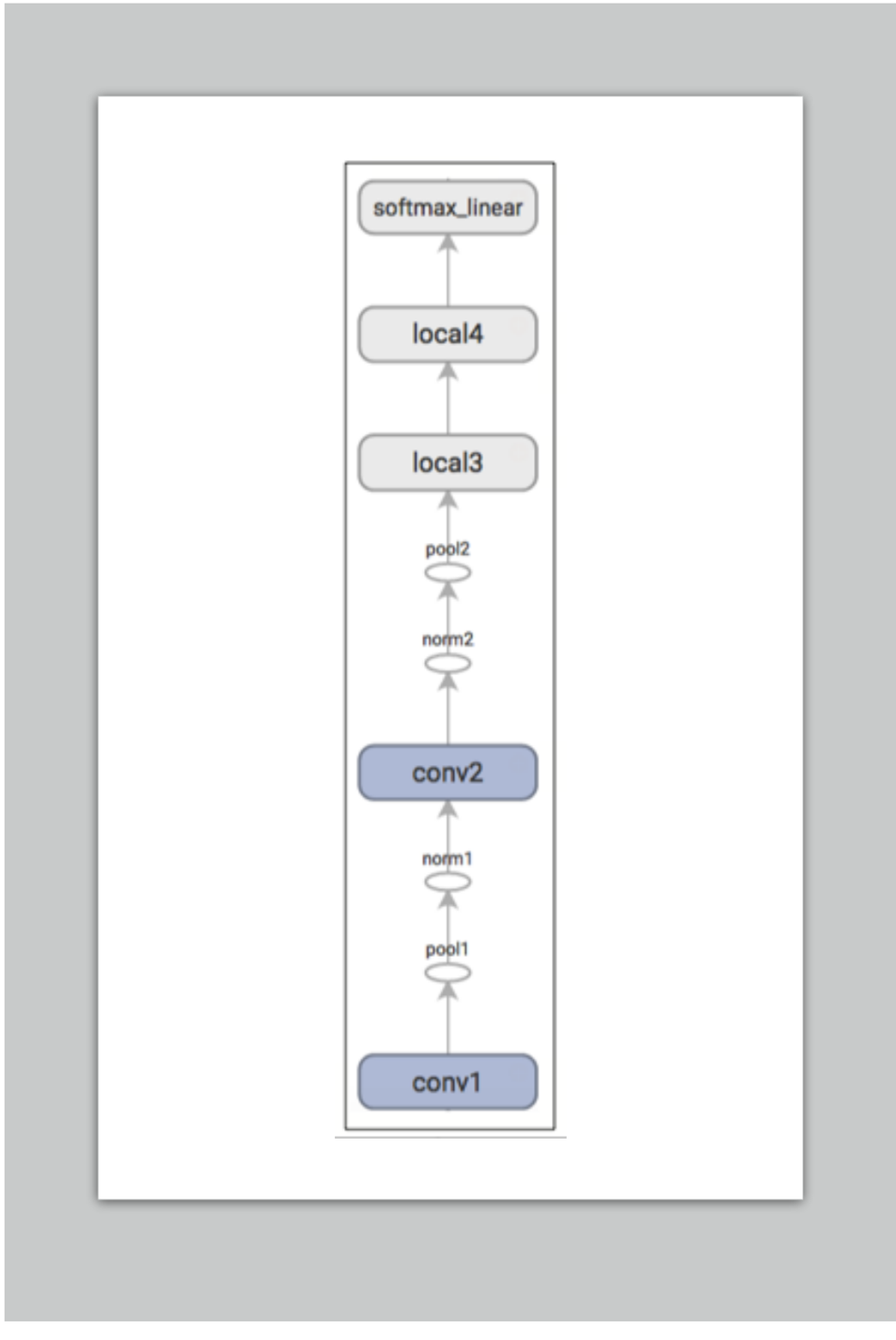
Fully Connected Layer

- full connections to all activations in the previous layer.



General CNN Model

- CONV: Representation & Local Invariant
- POOLING: Reduce Parameters & Reduce Overfitting
- Batch Normalization: Recovery the data;
- Fully Connected: Use all the information to classify;
- Softmax: Get Probabilities



LeNet-5

Yan LeCun in 1998 used for hand written digit recognition.

Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1×1	5×5	1	tanh
S4	Avg Pooling	16	5×5	2×2	2	tanh
C3	Convolution	16	10×10	5×5	1	tanh
S2	Avg Pooling	6	14×14	2×2	2	tanh
C1	Convolution	6	28×28	5×5	1	tanh
In	Input	1	32×32	–	–	–

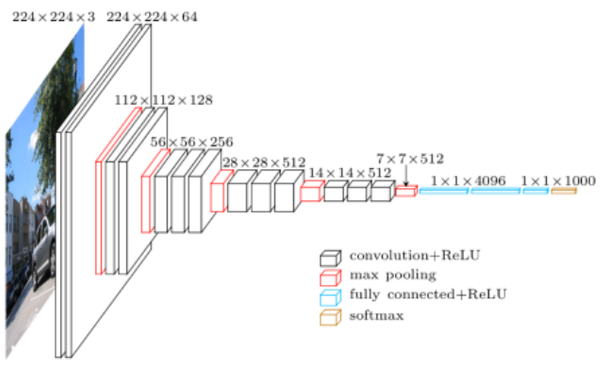
Alex Net

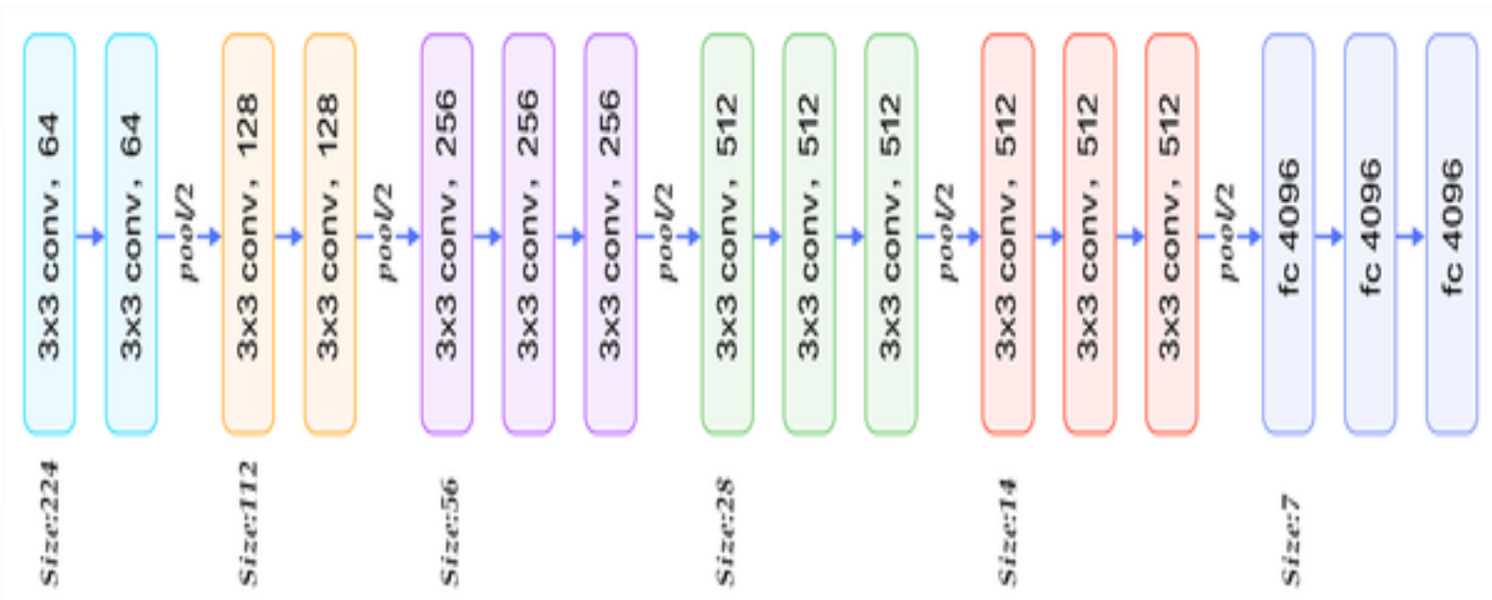
2012 Image Net

Layer	Type	Maps	Size	Kernel size	Stride	Padding	Activation
Out	Fully Connected	–	1,000	–	–	–	Softmax
F9	Fully Connected	–	4,096	–	–	–	ReLU
F8	Fully Connected	–	4,096	–	–	–	ReLU
C7	Convolution	256	13×13	3×3	1	SAME	ReLU
C6	Convolution	384	13×13	3×3	1	SAME	ReLU
C5	Convolution	384	13×13	3×3	1	SAME	ReLU
S4	Max Pooling	256	13×13	3×3	2	VALID	–
C3	Convolution	256	27×27	5×5	1	SAME	ReLU
S2	Max Pooling	96	27×27	3×3	2	VALID	–
C1	Convolution	96	55×55	11×11	4	SAME	ReLU
In	Input	3 (RGB)	224×224	–	–	–	–

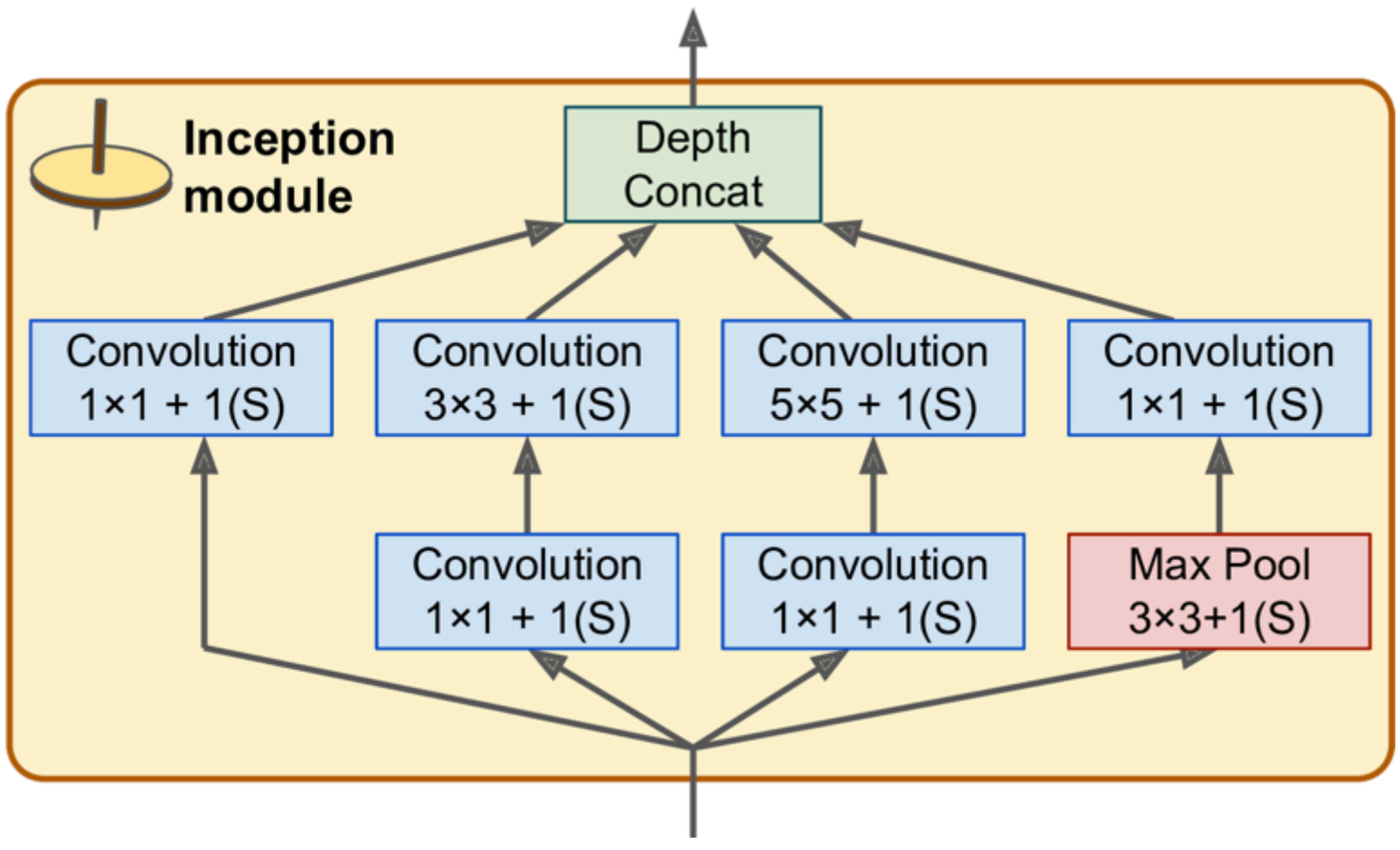
VGG

2014, The First Real Deep Neural Networks



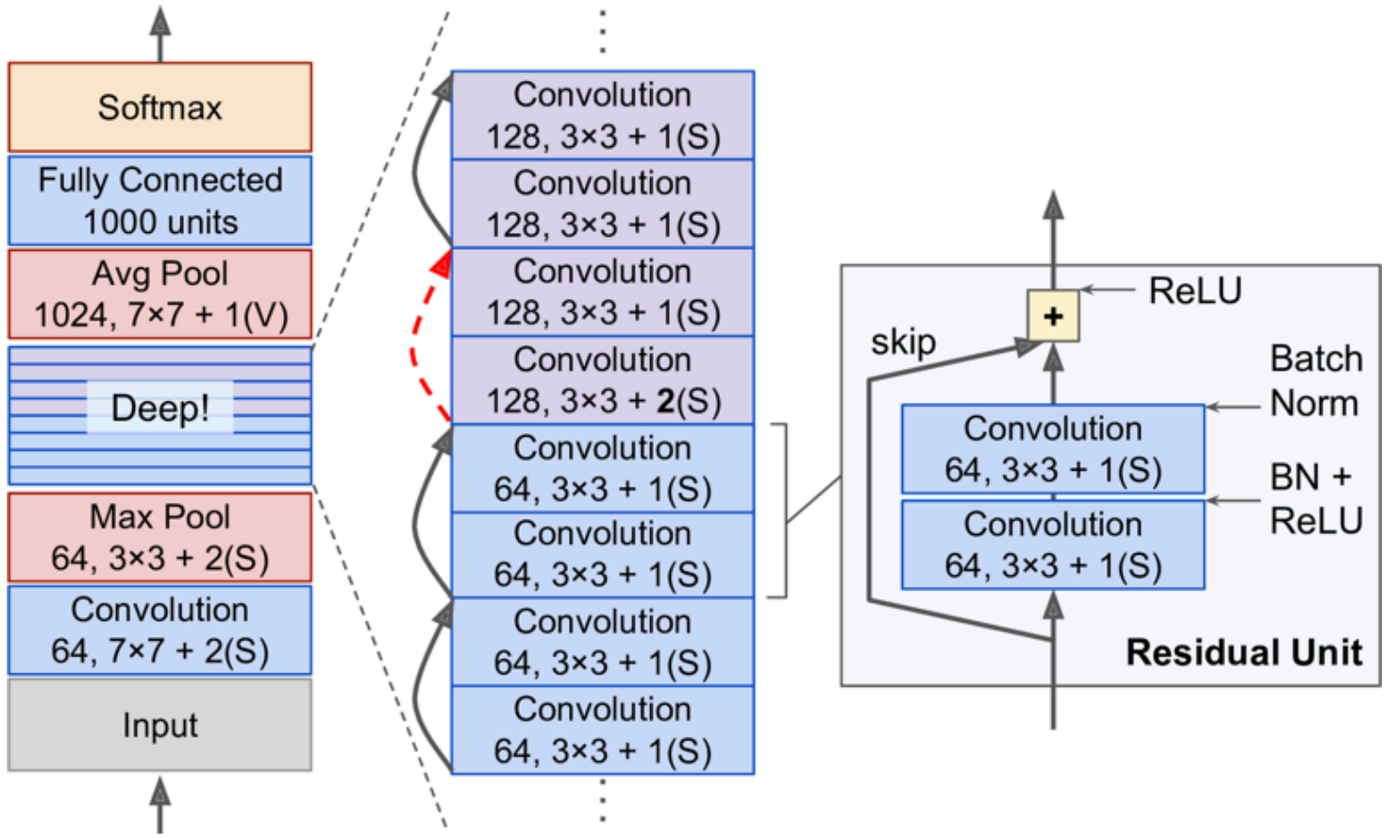


GoogLeNet
Inception Module

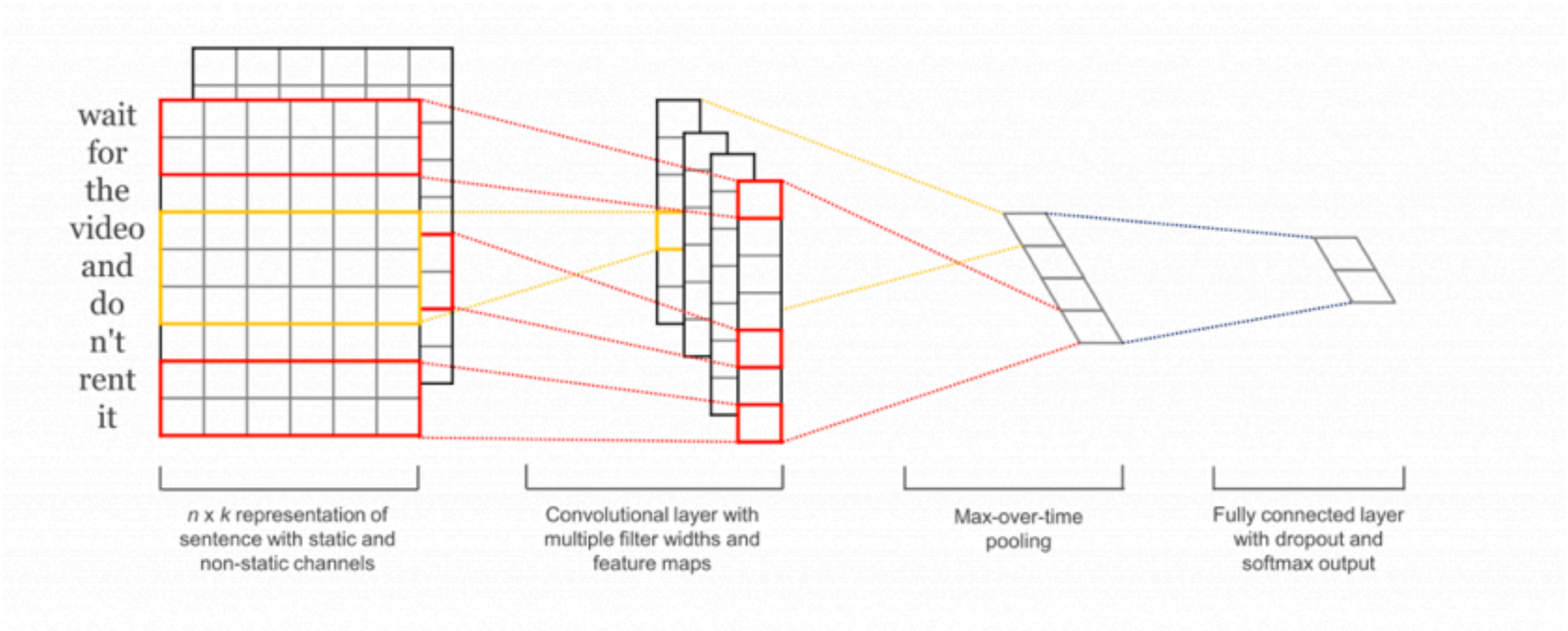


ResNet

MS, ImageNet 2015



TextCNN

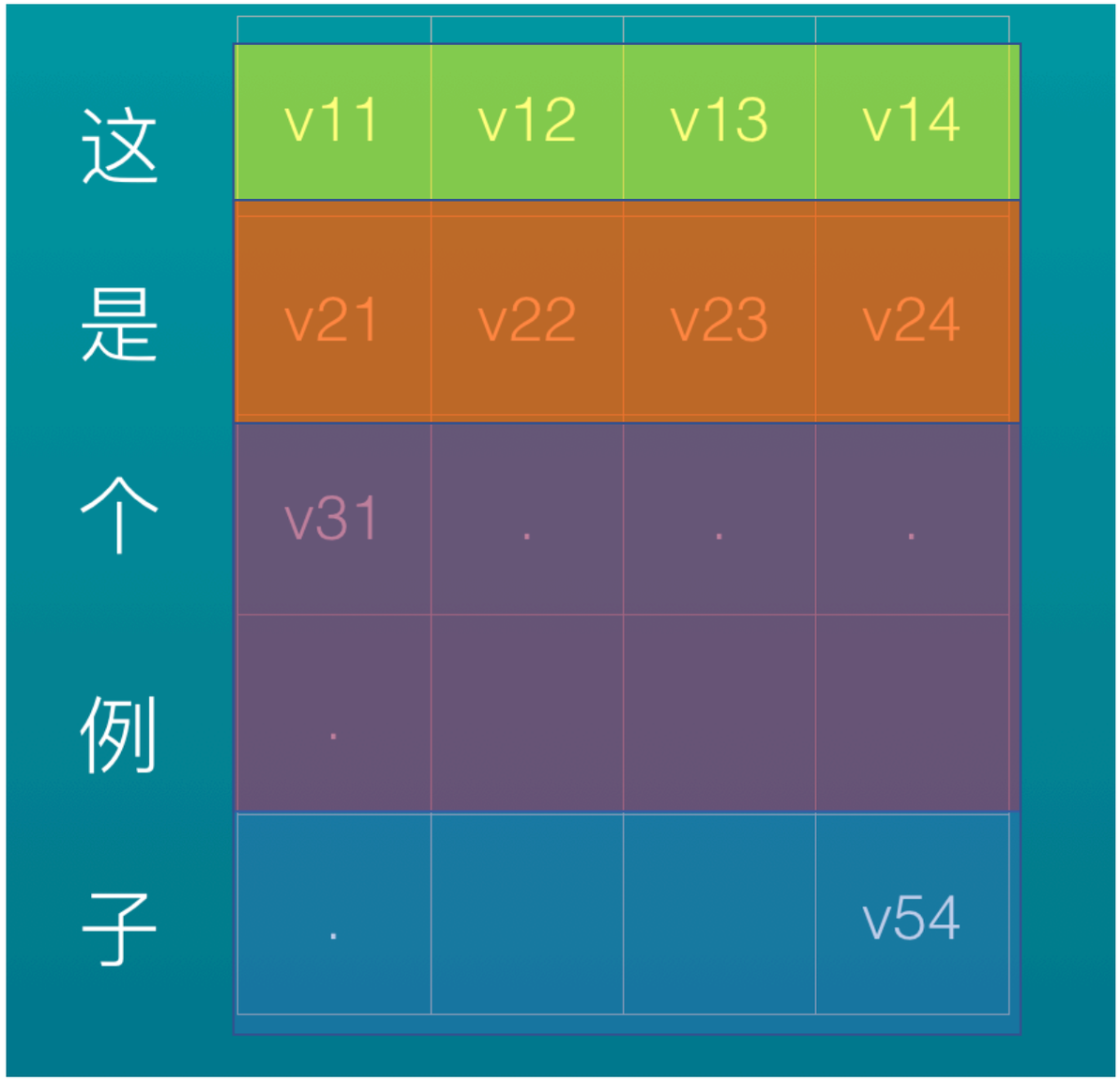

wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

- 1. word2vec, dimension is N
- 2. M words, we get M*N vectors.
- 3. similar to image processing, but the
- Width of windows must be N.

这	v11	v12	v13	v14
是	v21	v22	v23	v24
个	v31	.	.	.
例	.			
子	.			v54

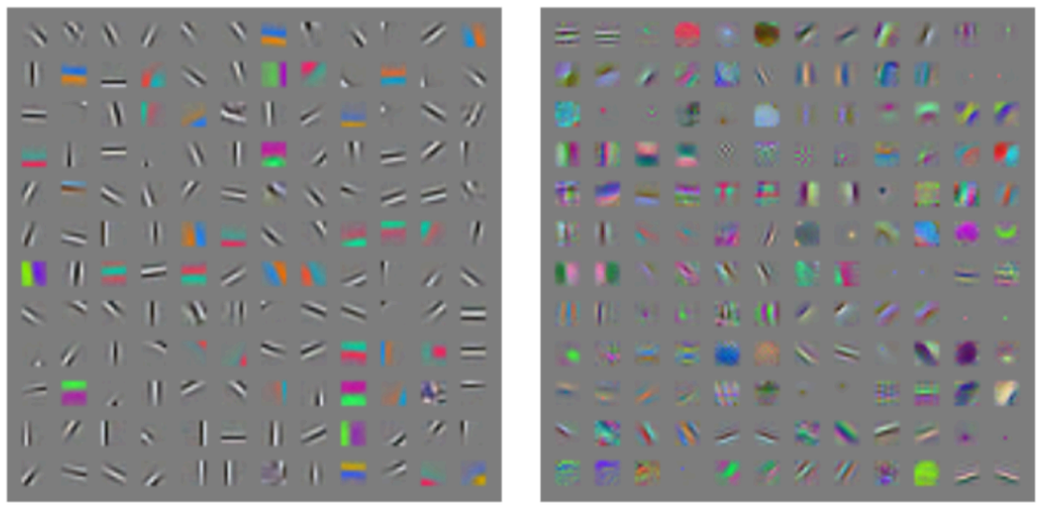
- 1. word2vec，dimension is N
- 2. M words, we get M*N vectors.
- 3. similar to image processing，but the Width of windows must be N.

- https://github.com/bhaveshoswal/CNN-text-classification-keras/blob/master/model.py
- https://github.com/dennybritz/cnn-text-classification-tf/blob/master/text_cnn.py
- https://github.com/dennybritz/cnn-text-classification-tf/blob/18762b459e21d9c70e5c242f8d43fc4e6db37a0d/text_cnn.py#L32



CNN More

- 1. CNN Visualization;
  - http://scs.ryerson.ca/~aharley/vis/conv/
- 2. Text CNN Paper Reading
  - https://arxiv.org/abs/1408.5882



Transfer Learning

An example of CNN classifier

- Jupyter Notebook

Assignment - Tensorflow CNN

1. https://github.com/Computing-Intelligence/jupyters_and_slides/blob/master/2019-summer/assignments/assignment-12-cnn.ipynb

2. 列出从LeNet到ResNet的网络的变化和区别