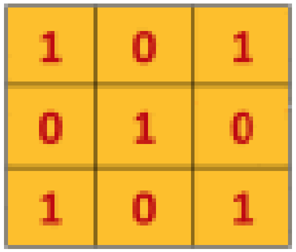
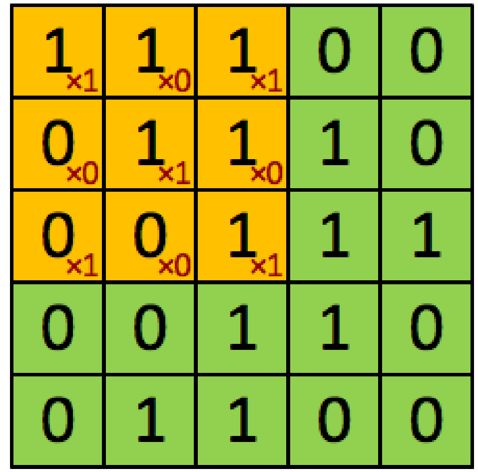
Convolutional Neural Networks

Convolutional Neural Networks

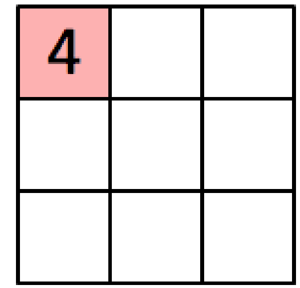
- Convolutional Layer [3, 4]



Filter / Kernel /
Set of Weights /
Feature Detector



Image



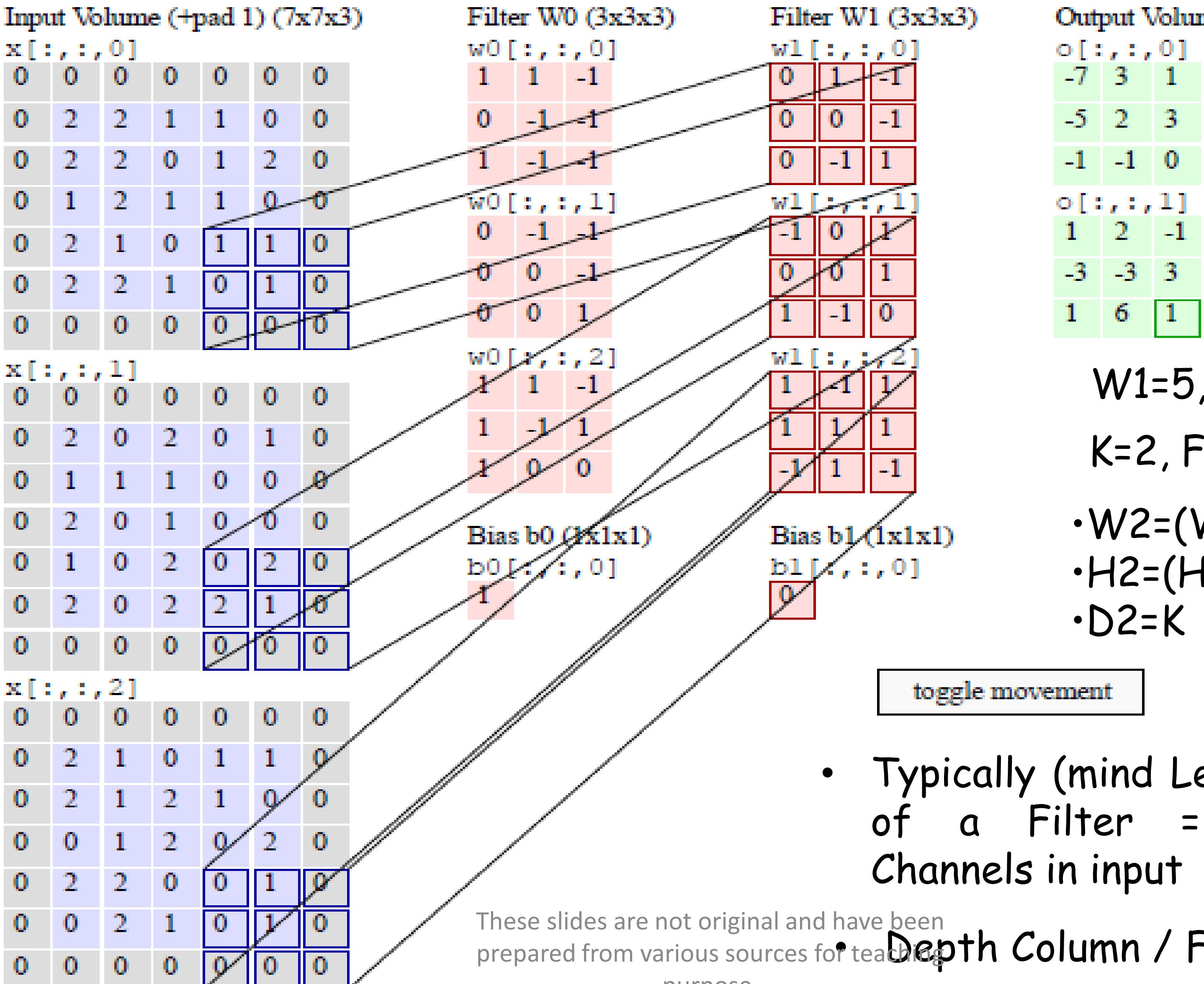
Convolved
Feature

Activation Map
/ Feature Map

- Size of output volume = $\dfrac{W - F + 2P}{S} + 1$
  - where W is the size of our input volume, F is the size of our filter, P is the amount of padding, and S is the stride

Convolutional Neural Networks

➢ Convolution Layer [3]



W1=5, H1=5, D1=3

K=2, F=3, S=2, P=1

- W2=(W1−F+2P)/S +1
- H2=(H1−F+2P)/S +1
- D2=K

- Typically (mind LeNet 5), Depth of a Filter = Number of Channels in input
- Depth Column / Fiber

Convolutional Neural Networks

- Convolution Layer [3]
  - Depth Column / Fiber: a set of neurons that are all looking at the same region of the input

Convolutional Neural Networks

- Convolution Layer [4]

Convolutional Neural Networks

- Convolution Layer [3]
  - Parameter Sharing

Convolutional Neural Networks

- Convolution Layer [3]
  - Local Connectivity & Receptive Field

Convolutional Neural Networks

- Convolution Layer [3]
  - Use of zero padding
    - Setting zero padding to be $P = (F-1)/2$ when the stride is $S=1$ ensures that the input volume and output volume will have the same size spatially.

Convolutional Neural Networks

- Convolution Layer [3]
  - Constraints on Stride
    - Note again that the spatial arrangement hyper-parameters have mutual constraints.

Convolutional Neural Networks

- Convolution Layer [3]
  - Constraints on Stride
    - Note again that the spatial arrangement hyper-parameters have mutual constraints.

    - For example, when the input has size W=10, no zero-padding is used P=0, and the filter size is F=3, then it would be impossible to use stride S=2, since $(W-F+2P)/S+1=(10-3+0)/2+1=4.5$, i.e. not an integer, indicating that the neurons don't "fit" neatly and symmetrically across the input.

Convolutional Neural Networks

- Convolution Layer [3]
  - Constraints on Stride
    - Note again that the spatial arrangement hyper-parameters have mutual constraints.
    - For example, when the input has size W=10, no zero-padding is used P=0, and the filter size is F=3, then it would be impossible to use stride S=2, since $(W-F+2P)/S+1=(10-3+0)/2+1=4.5$, i.e. not an integer, indicating that the neurons don't "fit" neatly and symmetrically across the input.
    - Therefore, this setting of the hyper-parameters is considered to be invalid, and a ConvNet library could throw an exception or zero pad the rest to make it fit, or crop the input to make it fit, or something.

Convolutional Neural Networks

- Convolution Layer [3]
  - Constraints on Stride
    - As we will see in the ConvNet architectures section, sizing the ConvNets appropriately so that all the dimensions “work out” can be a real headache, which the use of zero-padding and some design guidelines will significantly alleviate.

Convolutional Neural Networks

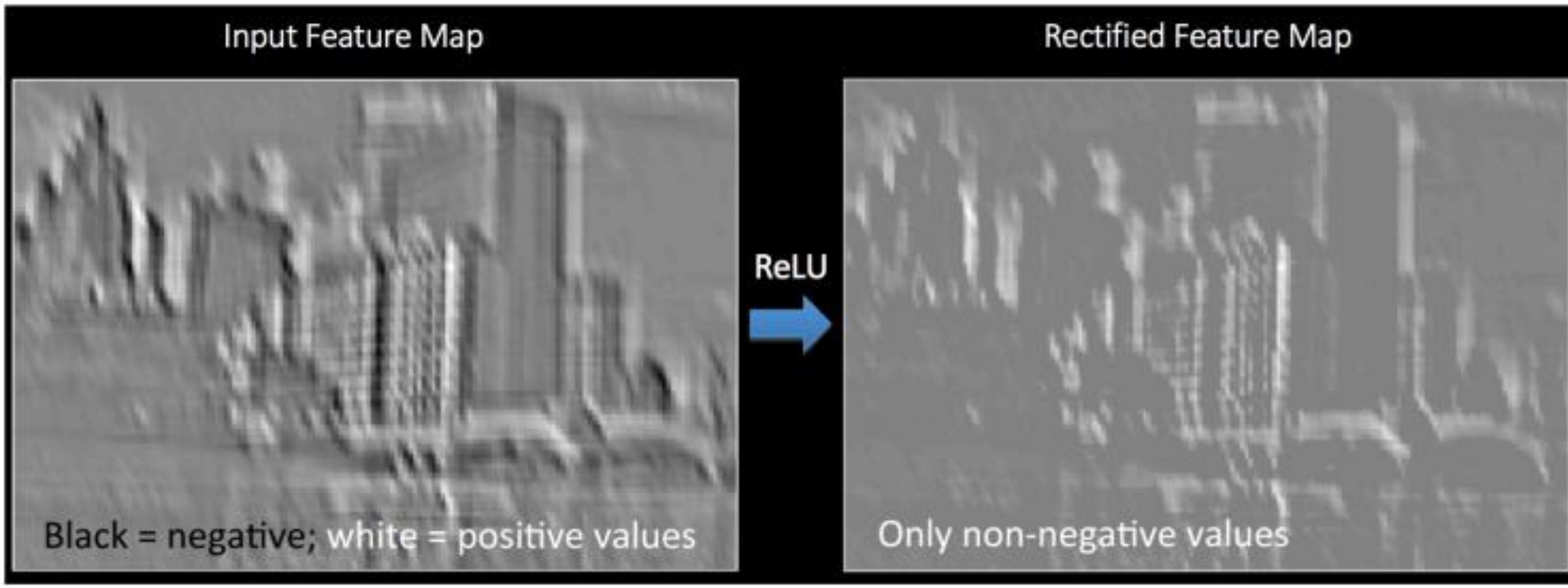
- Convolutional Layer [3]
  - Summary
    - Accepts a volume of size $W1 \times H1 \times D1$
    - Requires four hyper-parameters:
      - Number of filters K,
      - their spatial extent F,
      - the stride S,
      - the amount of zero padding P.
    - Produces a volume of size $W2 \times H2 \times D2$ where:
      - $W2=(W1-F+2P)/S+1$
      - $H2=(H1-F+2P)/S+1$
      - $D2=K$

Convolutional Neural Networks

- Convolutional Layer [3]
  - Summary
    - With parameter sharing, it introduces F·F·D1 weights per filter, for a total of (F·F·D1)·K weights and K biases.
    - In the output volume, the d^{th} depth slice (of size W2×H2) is the result of performing a valid convolution of the d^{th} filter over the input volume with a stride of S, and then offset by d^{th} bias.

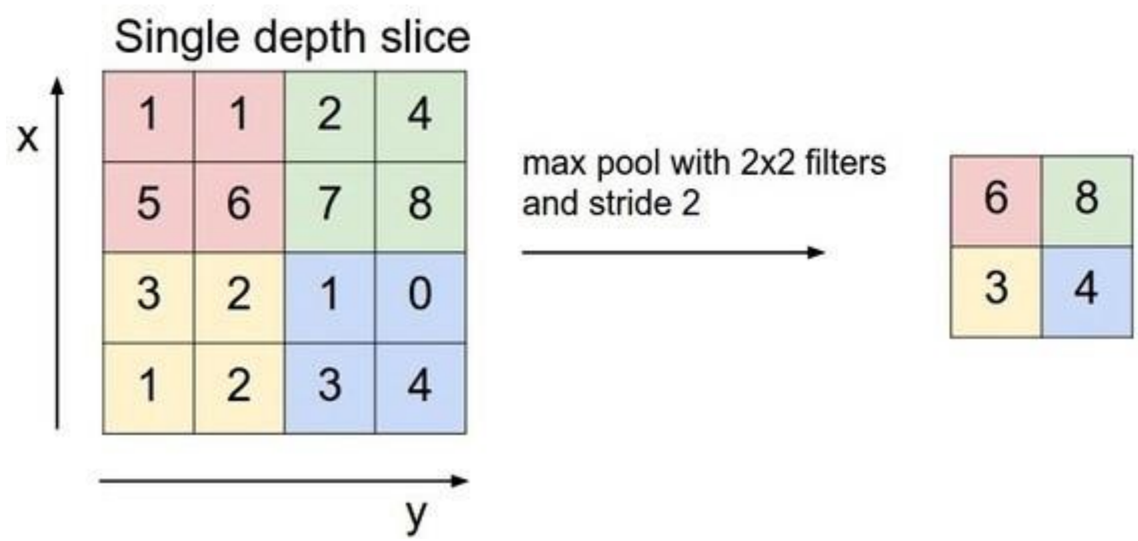
Convolutional Neural Networks

- The ReLU Operation [4]



Convolutional Neural Networks

- Max Pooling Layer [3]



- Size of output volume = $\dfrac{W - F}{S} + 1$

Convolutional Neural Networks

- Pooling Layer [3]
  - Summary
    - Accepts a volume of size W1×H1×D1
    - Requires two hyper-parameters:
      - their spatial extent F,
      - the stride S,
    - Produces a volume of size W2×H2×D2 where:
      - $W2=(W1-F)/S+1$
      - $H2=(H1-F)/S+1$
      - $D2=D1$
    - Introduces zero parameters since it computes a fixed function of the input
    - Note that it is not common to use zero-padding for Pooling layers

Convolutional Neural Networks

- Pooling Layer [3]
  - Summary
    - It is worth noting that there are only two commonly seen variations of the max pooling layer found in practice: **A pooling layer with F=3, S=2** (also called overlapping pooling), and **more commonly F=2, S=2**. Pooling sizes with larger receptive fields are too destructive.

Convolutional Neural Networks

- Pooling Layer [3]
  - Summary
    - It is worth noting that there are only two commonly seen variations of the max pooling layer found in practice: **A pooling layer with F=3, S=2** (also called overlapping pooling), and **more commonly F=2, S=2**. Pooling sizes with larger receptive fields are too destructive.

    - In addition to max pooling, the pooling units can also perform other functions, such as **average pooling** or even **L2-norm pooling**. Average pooling was often used historically but has recently fallen out of favour compared to the max pooling operation, which has been shown to work better in practice.

Convolutional Neural Networks

- Pooling Layer [3]
  - Getting rid of Pooling
    - Many people dislike the pooling operation and think that we can get away without it.

Convolutional Neural Networks

- Pooling Layer [3]
  - Getting rid of Pooling
    - Many people dislike the pooling operation and think that we can get away without it.
    - For example, the paper "Striving for Simplicity: The All Convolutional Net" proposes to discard the pooling layer in favour of architecture that only consists of repeated CONV layers.

Convolutional Neural Networks

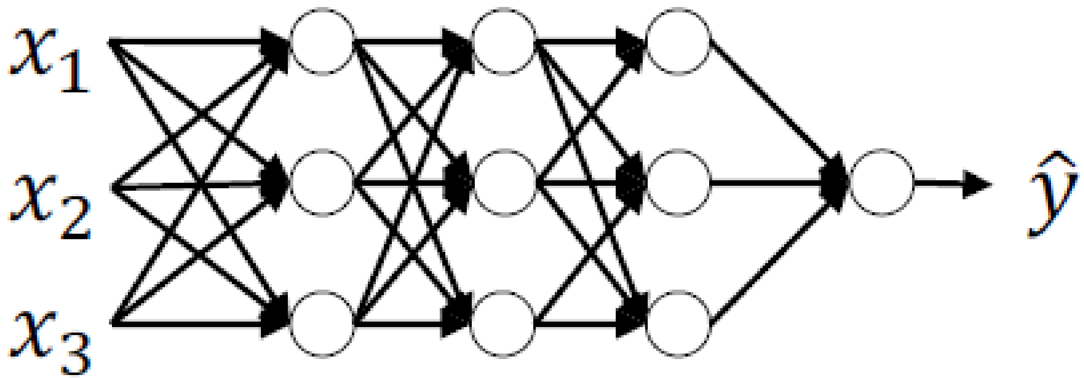
- Pooling Layer [3]
  - Getting rid of Pooling
    - Many people dislike the pooling operation and think that we can get away without it.
    - For example, the paper “Striving for Simplicity: The All Convolutional Net” proposes to discard the pooling layer in favour of architecture that only consists of repeated CONV layers.
    - To reduce the size of the representation they suggest using larger stride in CONV layer once in a while.

Convolutional Neural Networks

- Pooling Layer [3]
  - Getting rid of Pooling
    - Many people dislike the pooling operation and think that we can get away without it.
    - For example, the paper "Striving for Simplicity: The All Convolutional Net" proposes to discard the pooling layer in favour of architecture that only consists of repeated CONV layers.
    - To reduce the size of the representation they suggest using larger stride in CONV layer once in a while.
    - Discarding pooling layers has also been found to be important in training good generative models, such as variational autoencoders (VAEs) or generative adversarial networks (GANs). It seems likely that future architectures will feature very few to no pooling layers.

Convolutional Neural Networks

- Normalization Layer [3, Andrew Ng's Lecture on BN]



Convolutional Neural Networks

➢ Normalization Layer [Andrew Ng's Lecture on BN]

$$\mu = \frac{1}{m}\sum_i z^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_i (z_i - \mu)^2$$

$$z^{(i)}_{norm} = \frac{z^{(i)} - \boxed{\mu}}{\boxed{\sqrt{\sigma^2 + \varepsilon}}}$$

Mean = 0
Variance = 1

$$\tilde{z}^{(i)} = \gamma z^{(i)}_{norm} + \beta$$

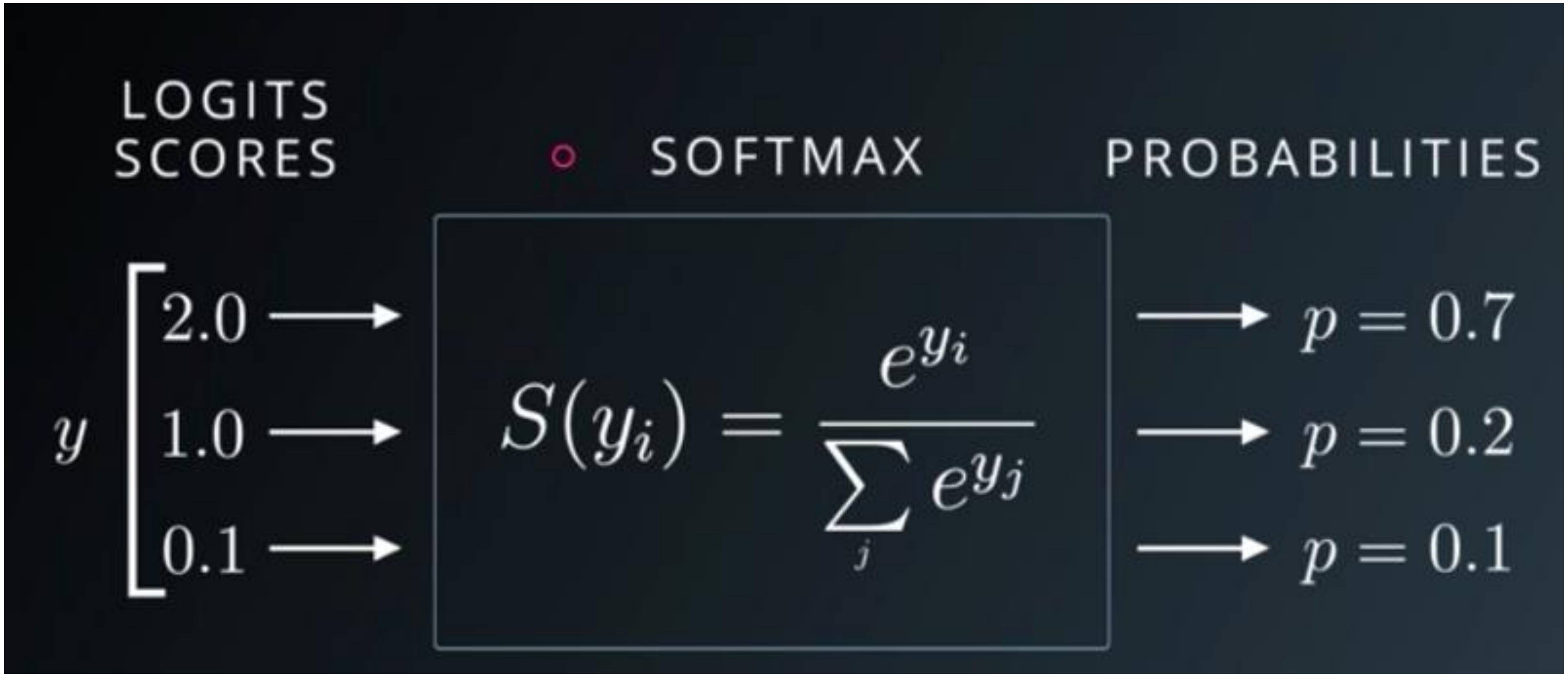
Mean = Beta
Variance = Gamma

Convolutional Neural Networks

- Fully Connected Layer [3]
  - Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks.
  - Their activations can hence be computed with a matrix multiplication followed by a bias offset.

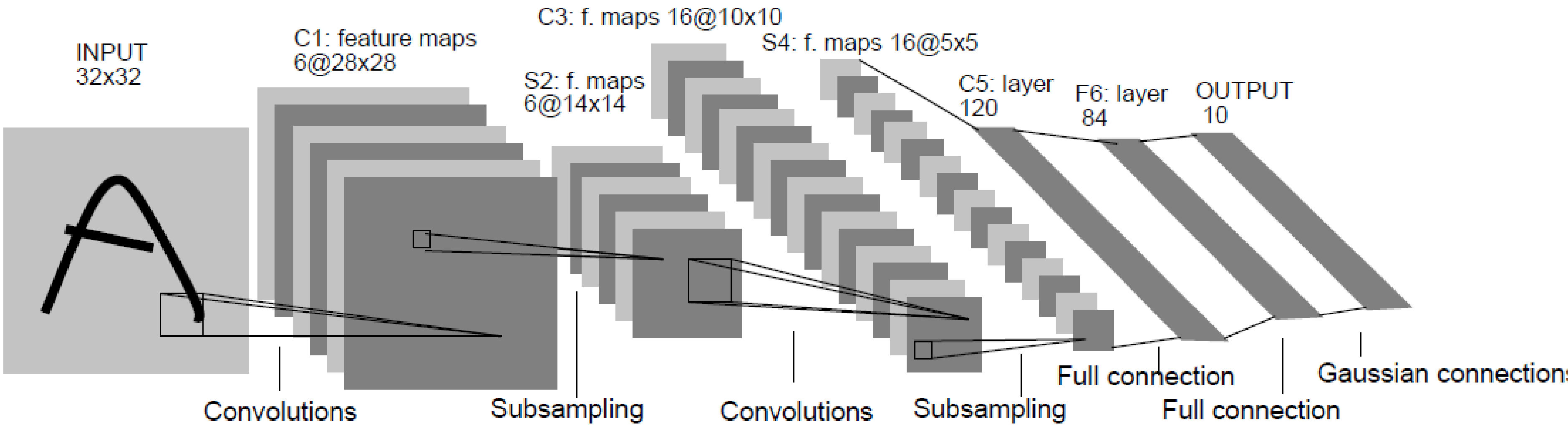
Convolutional Neural Networks

- Softmax Activation Function [5]



- Original Source: Udacity Deep Learning Slides on Softmax

Convolutional Neural Networks [1]



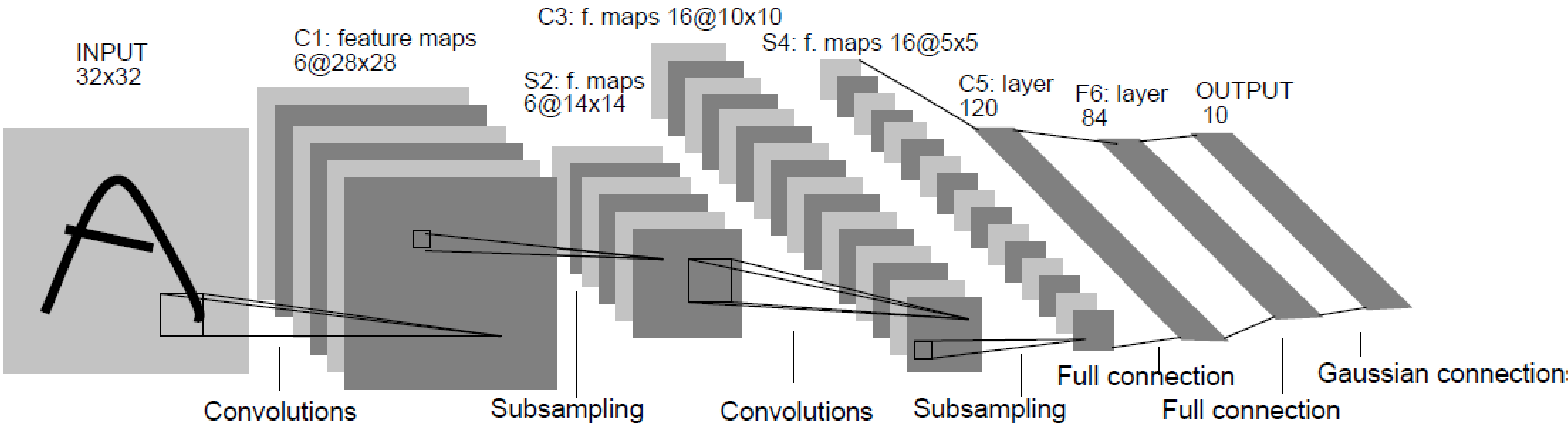
- Layer C1:

  (5*5+1)*6=156 parameters to learn

  Connections: 28*28*(5*5+1)*6=122304

  If it was fully connected we had (32*32+1)*(28*28)*6 parameters

Convolutional Neural Networks [1]

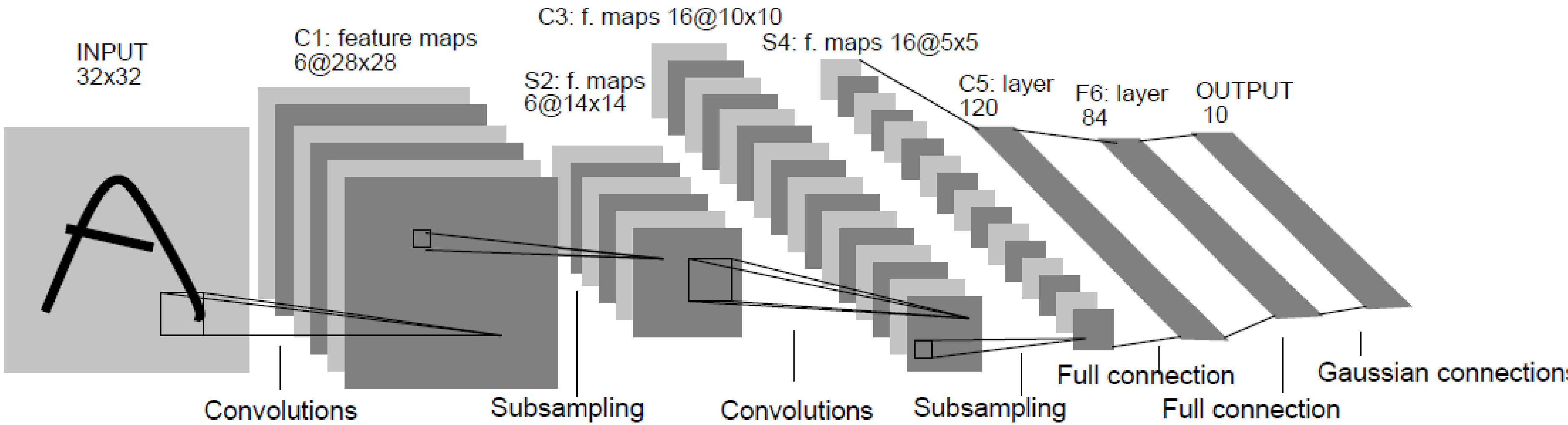


- Layer S2:

  Layer S2: 6*2=12 trainable parameters.

  Connections: 14*14*(2*2+1)*6=5880

Convolutional Neural Networks [1]



➢ Layer C3:

- C3: Convolutional layer with 16 feature maps of size 10x10
- Each unit in C3 is connected to several! 5x5 receptive fields at identical locations in S2

Layer C3:

1516 trainable parameters.

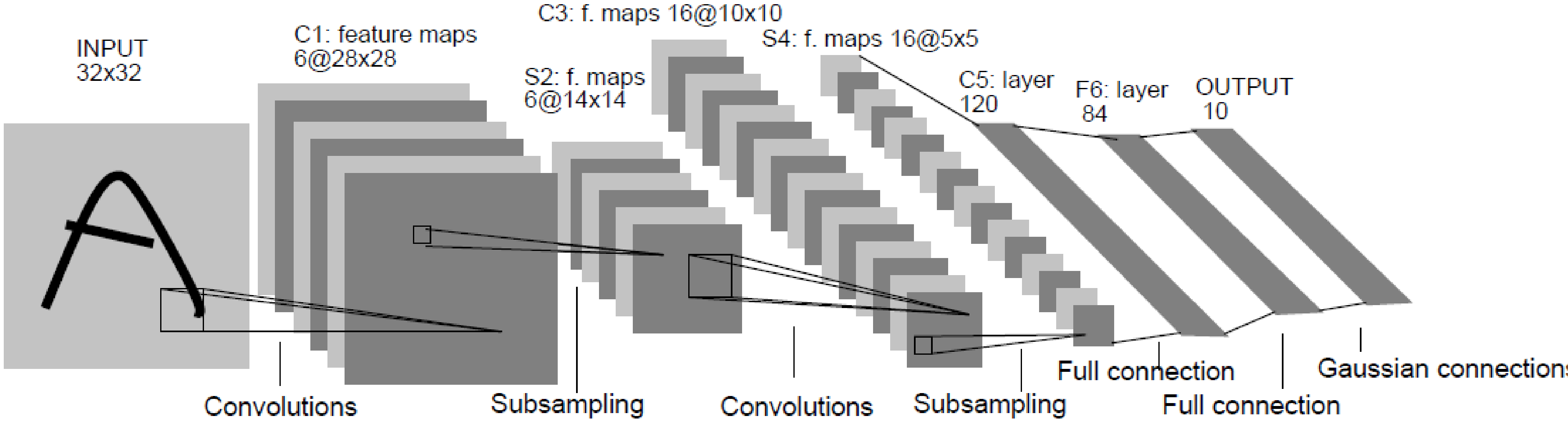
Connections: 151600

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I

Each column indicates which feature map in S2 are combined by the units in a particular feature map of C3.

Convolutional Neural Networks [1]



- Layer C3:

- C3: Convolutional layer with 16 feature maps of size 10x10
- Each unit in C3 is connected to several! 5x5 receptive fields at identical locations in S2

Layer C3: 6*(5*5*3) + 9*(5*5*4) + 1*(5*5*6) + 16

1516 trainable parameters.

Connections: 151600

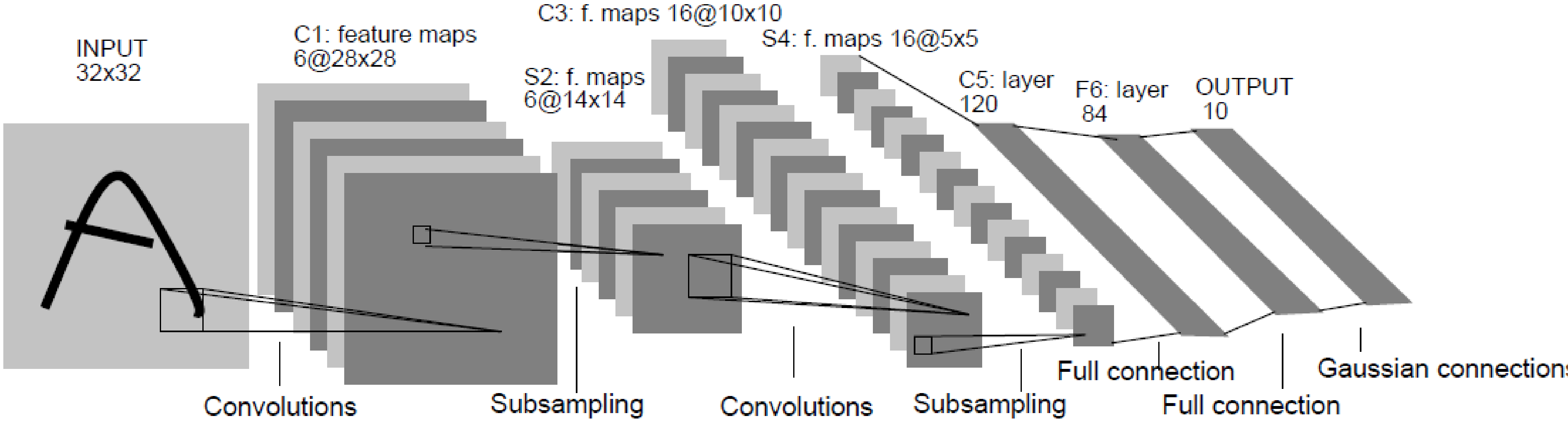
[(75 + 1) * (10 * 10) * 6] +
[(100 + 1) * (10 * 10) * 9] +
[(150 + 1) * (10 * 10) * 1]

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I

Each column indicates which feature map in S2 are combined by the units in a particular feature map of C3.

Convolutional Neural Networks [1]

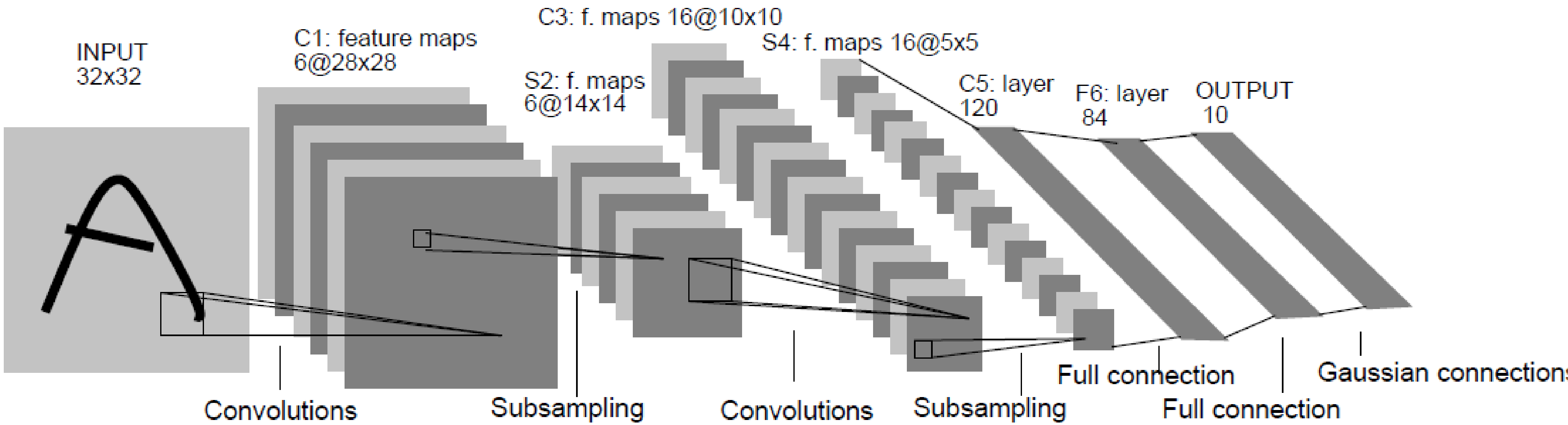


- Layer S4:
  - S4: Subsampling layer with 16 feature maps of size 5x5
  - Each unit in S4 is connected to the corresponding 2x2 receptive field at C3

Layer S4: 16*2=32 trainable parameters.

Connections: 5*5*(2*2+1)*16=2000

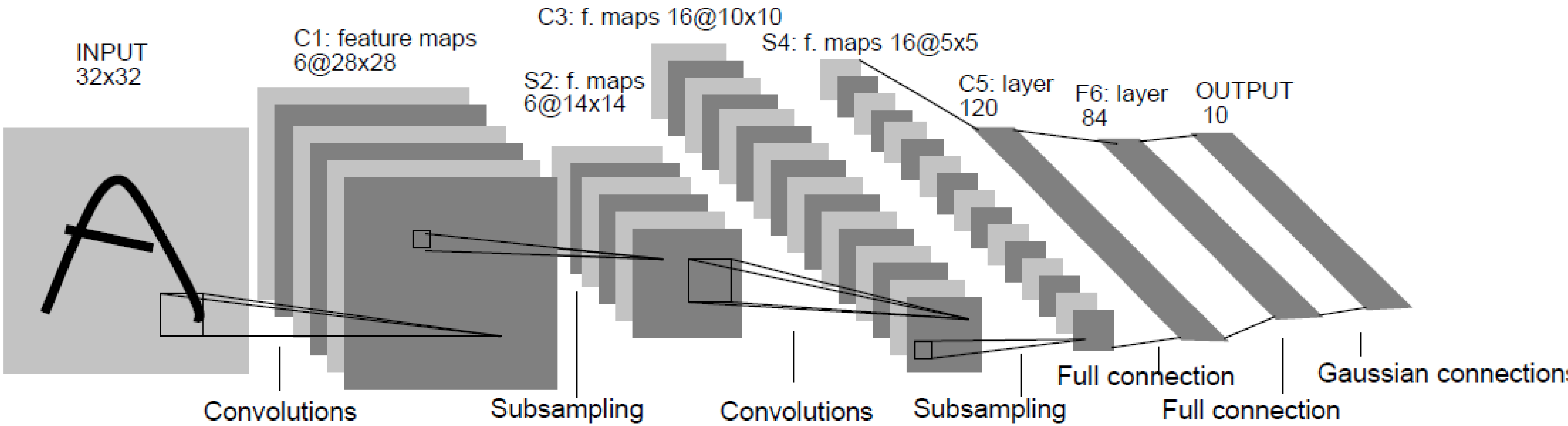
Convolutional Neural Networks [1]



- Layer C5:
  - C5: Convolutional layer with 120 feature maps of size 1x1
  - Each unit in C5 is connected to all 16 5x5 receptive fields in S4

Layer C5: 120*(16*25+1) = 48120 trainable parameters and connections
(Fully connected)

Convolutional Neural Networks [1]



- Layer F6 & Output:

  Layer F6: 84 fully connected units. 84*(120+1)=10164 trainable parameters and connections.

  Output layer: 10RBF (One for each digit)

  84=7x12, stylized image

  Weight update: Backpropagation

Convolutional Neural Networks

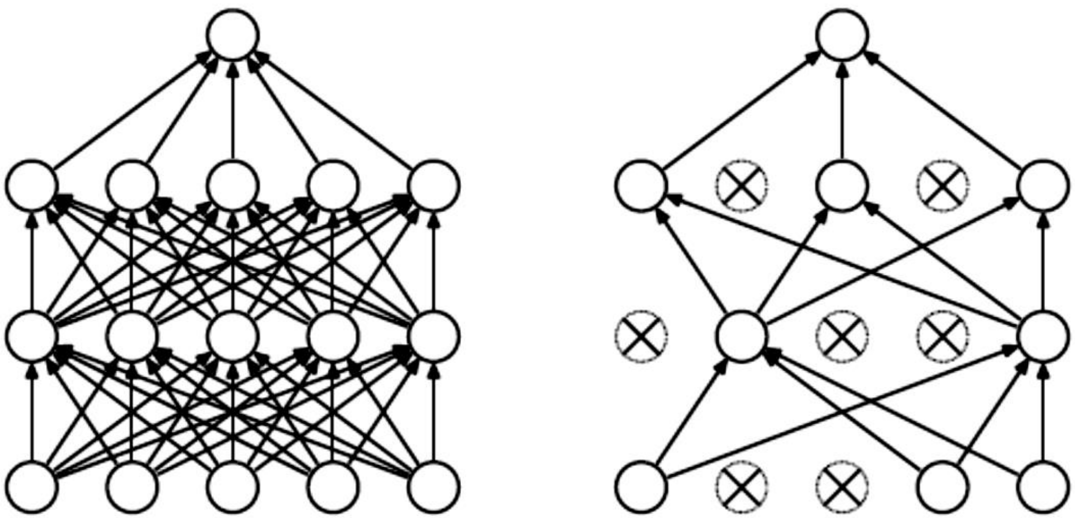
- Dropout [2]
  - It prevents overfitting

Convolutional Neural Networks

- Dropout [2]
  - It prevents overfitting
  - Provides a way of approximately combining exponentially many different neural network architectures efficiently.

Convolutional Neural Networks

- Dropout [2]
  - The term "dropout" refers to dropping out units (hidden and visible) in a neural network. By dropping a unit out, we mean temporarily removing it from the network, along with all its incoming and outgoing connections, as shown in following Figure 1.



(a) Standard Neural Net (b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Convolutional Neural Networks

- Dropout [2]
  - The choice of which units to drop is random.

Convolutional Neural Networks

- Dropout [2]
  - The choice of which units to drop is random.
  - In the simplest case, each unit is retained with a fixed probability p independent of other units, where p can be chosen using a validation set or can simply be set at 0.5, which seems to be close to optimal for a wide range of networks and tasks.

Convolutional Neural Networks

- Dropout [2]
  - The choice of which units to drop is random.
  - In the simplest case, each unit is retained with a fixed probability p independent of other units, where p can be chosen using a validation set or can simply be set at 0.5, which seems to be close to optimal for a wide range of networks and tasks.
  - For the input units, however, the optimal probability of retention is usually closer to 1 than to 0.5.

Convolutional Neural Networks

- Dropout [2]
  - Applying dropout to a neural network amounts to sampling a “thinned" network from it. The thinned network consists of all the units that survived dropout (Figure 1b).

Convolutional Neural Networks

- Dropout [2]
  - Applying dropout to a neural network amounts to sampling a "thinned" network from it. The thinned network consists of all the units that survived dropout (Figure 1b).

  - A neural net with n units, can be seen as a collection of 2^n possible thinned neural networks. These networks all share weights so that the total number of parameters is still $O(n^2)$, or less.

Convolutional Neural Networks

- Dropout [2]
  - Applying dropout to a neural network amounts to sampling a "thinned" network from it. The thinned network consists of all the units that survived dropout (Figure 1b).
  - A neural net with n units, can be seen as a collection of 2^n possible thinned neural networks. These networks all share weights so that the total number of parameters is still $O(n^2)$, or less.
  - **For each presentation of each training case, a new thinned network is sampled and trained.** So training a neural network with dropout can be seen as training a collection of 2^n thinned networks with extensive weight sharing, where each thinned network gets trained very rarely, if at all.

Convolutional Neural Networks

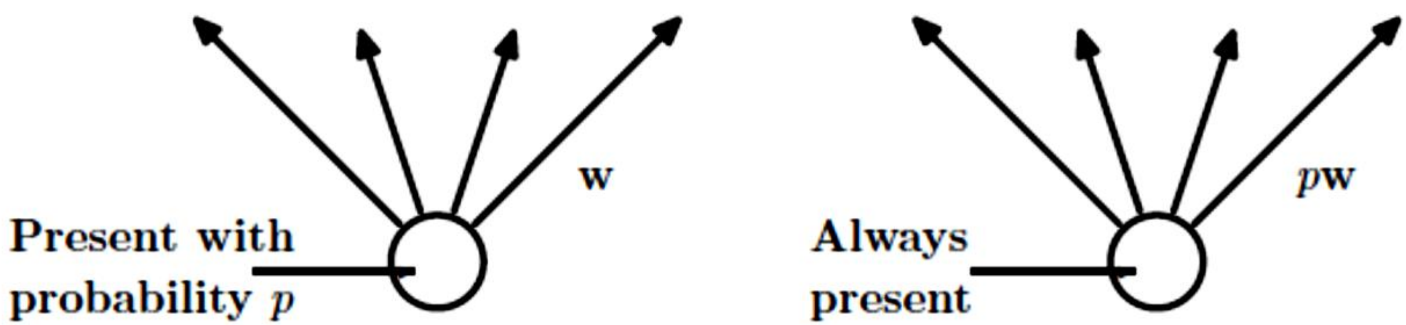
- Dropout [2]
  - At test time, it is not feasible to explicitly average the predictions from exponentially many thinned models. However, a very simple approximate averaging method works well in practice.

Convolutional Neural Networks

- Dropout [2]
  - At test time, it is not feasible to explicitly average the predictions from exponentially many thinned models. However, a very simple approximate averaging method works well in practice.
  - The idea is to use a single neural net at test time without dropout.

Convolutional Neural Networks

- Dropout [2]
  - At test time, it is not feasible to explicitly average the predictions from exponentially many thinned models. However, a very simple approximate averaging method works well in practice.
  - The idea is to use a single neural net at test time without dropout.
  - The weights of this network are scaled-down versions of the trained weights. If a unit is retained with probability p during training, the outgoing weights of that unit are multiplied by p at test time as shown in Figure 2.



(a) At training time (b) At test time

Figure 2: **Left**: A unit at training time that is present with probability p and is connected to units in the next layer with weights **w**. **Right**: At test time, the unit is always present and the weights are multiplied by p. The output at test time is same as the expected output at training time.

References

1. LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

2. Srivastava, Nitish, et al. "Dropout: A simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1 (2014): 1929-1958.

3. http://cs231n.github.io/convolutional-networks/

4. https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

5. https://medium.com/@uniqtech/understand-the-softmax-function-in-minutes-f3a59641e86d

Disclaimer

- These slides are not original and have been prepared from various sources for teaching purpose.