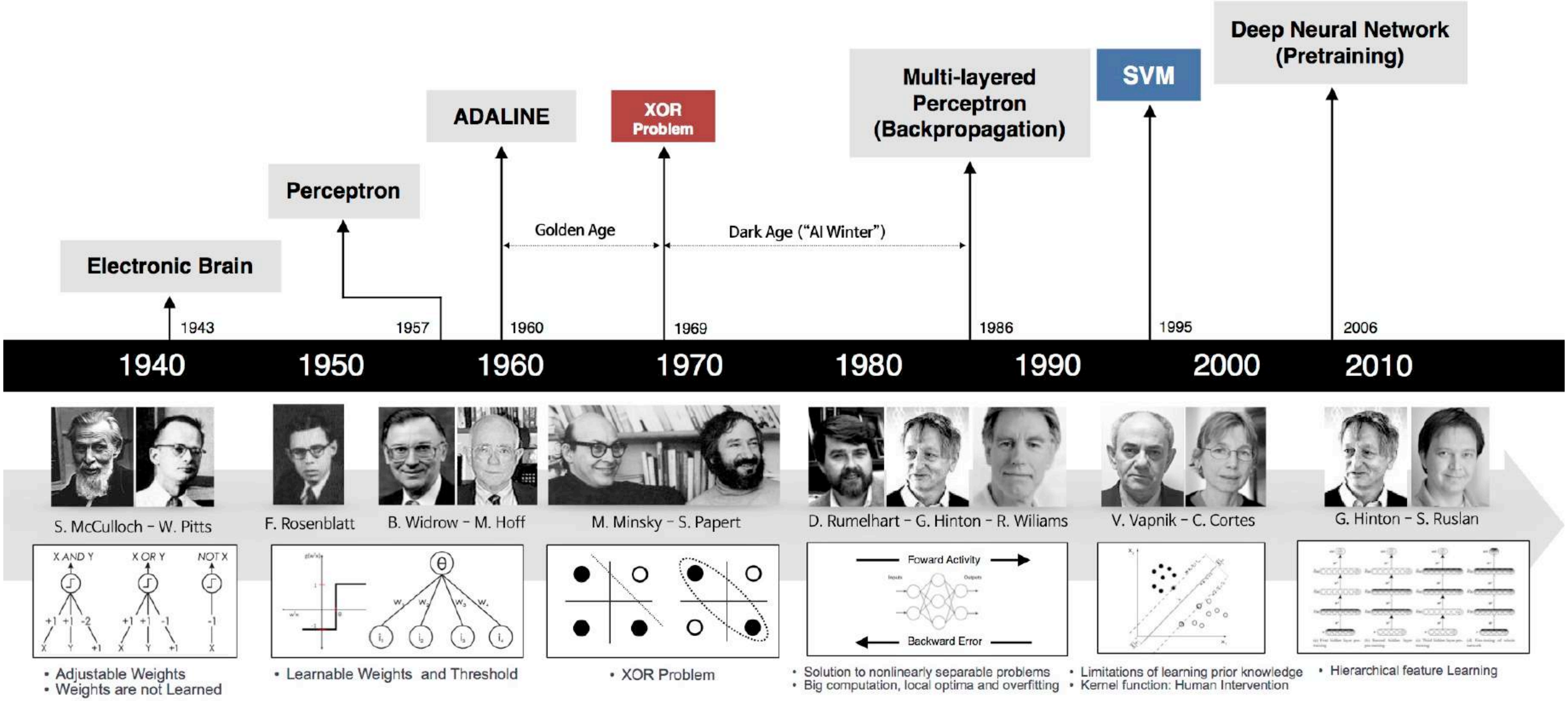
Deep Learning Architecture - I

Peep into CNNs and RNNs

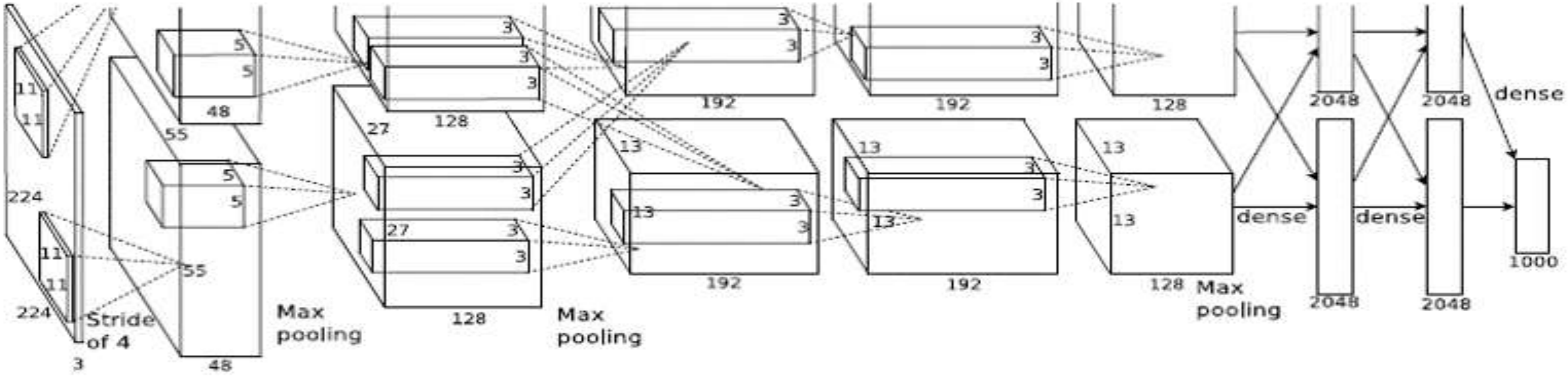
Deep Learning: Agenda

- Introduce Deep Learning and Two Popular Architectures
  - Convolutional Neural Networks (CNNs)
  - Recurrent Neural Networks (RNNs)
- Introduce PyTorch and Programming for DL

Summary of History



AlexNet (NIPS 2012)



ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

ImageNet Classification Task:

Previous Best: ~25% (CVPR-2011)
AlexNet : ~15 % (NIPS-2012)

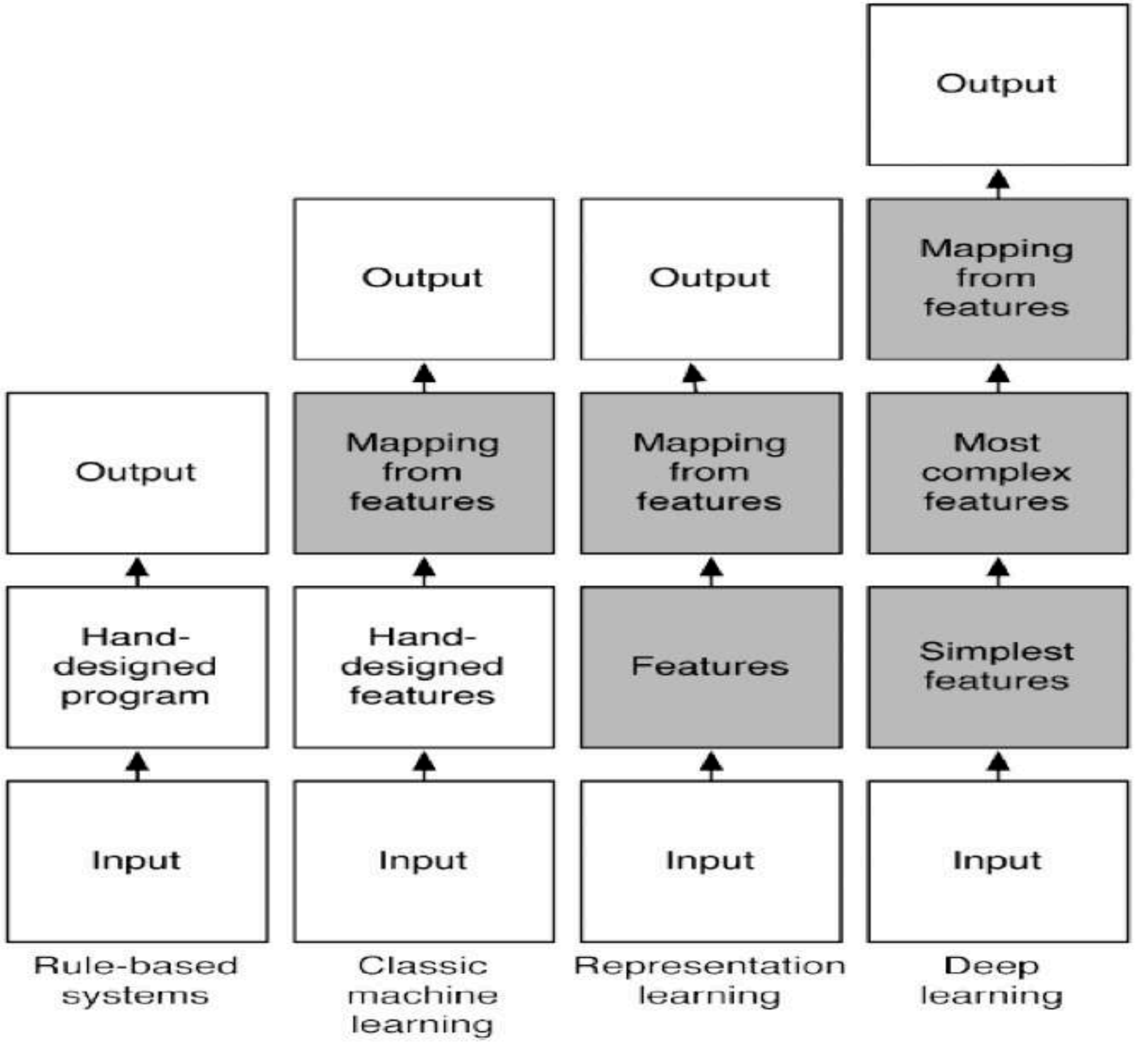
Success of “Deep Learning”: ImageNet Challenge

Top-5 Error on Imagenet Classification Challenge (1000 classes)

Method	Top-Error Rate
SIFT+FV [CVPR 2011]	~25.7%
AlexNet [NIPS 2012]	~15%
OverFeat [ICLR 2014]	~ 13%
ZeilerNet [ImageNet 2013]	~11%
Oxford-VGG [ICLR 2015]	~7%
GoogLeNet [CVPR 2015]	~6%, ~4.5%
ResNet [CVPR16]	~3.5%
Human Performance	3 to 5 %

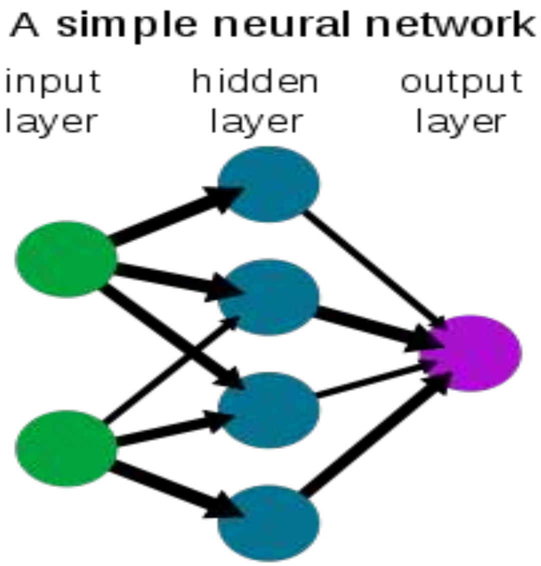
Mostly Deeper Networks
Smaller Convolutions
Many Specific Enhancements

What is deep learning?



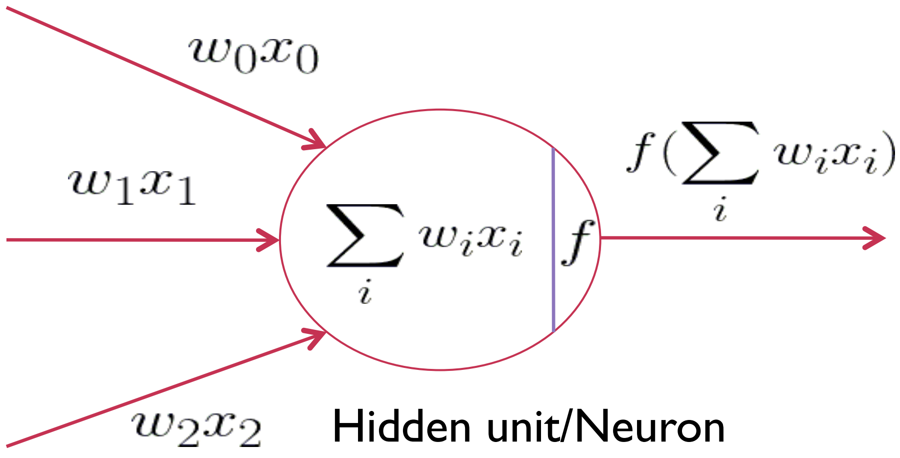
Y. Bengio et al, ``Deep Learning", MIT Press, 2015

Neural Networks

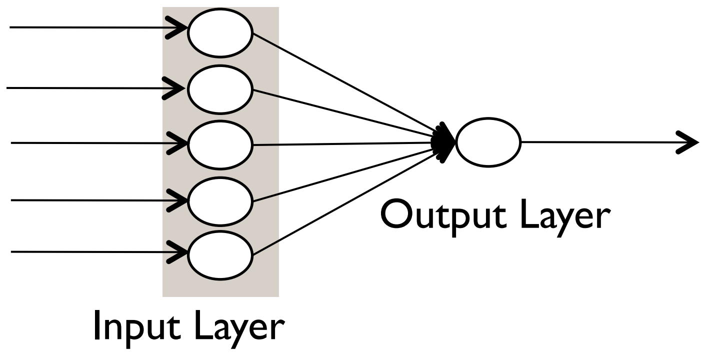


- Biologically inspired networks.
- Complex function approximation through composition of functions.
- Can learn arbitrary Nonlinear decision boundary

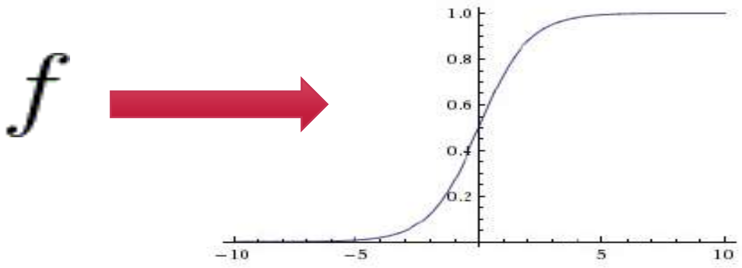
Neuron, Perceptron and MLP



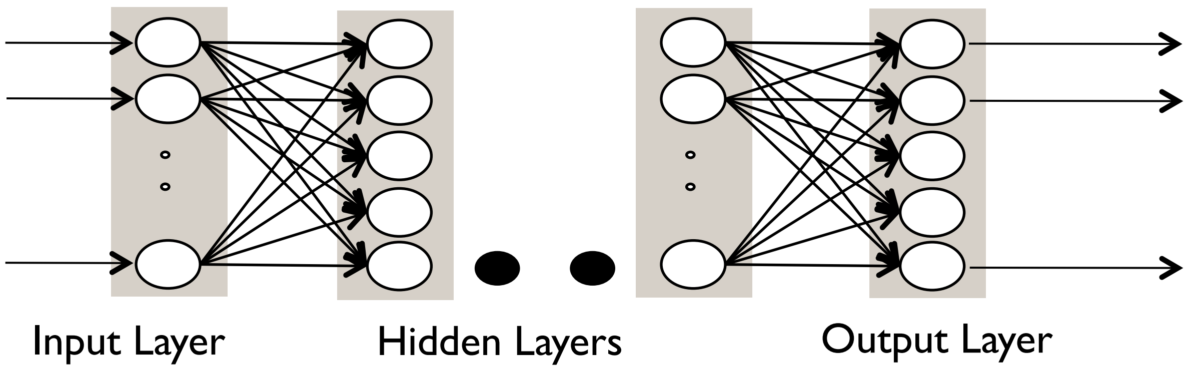
Hidden unit/Neuron



Perceptron

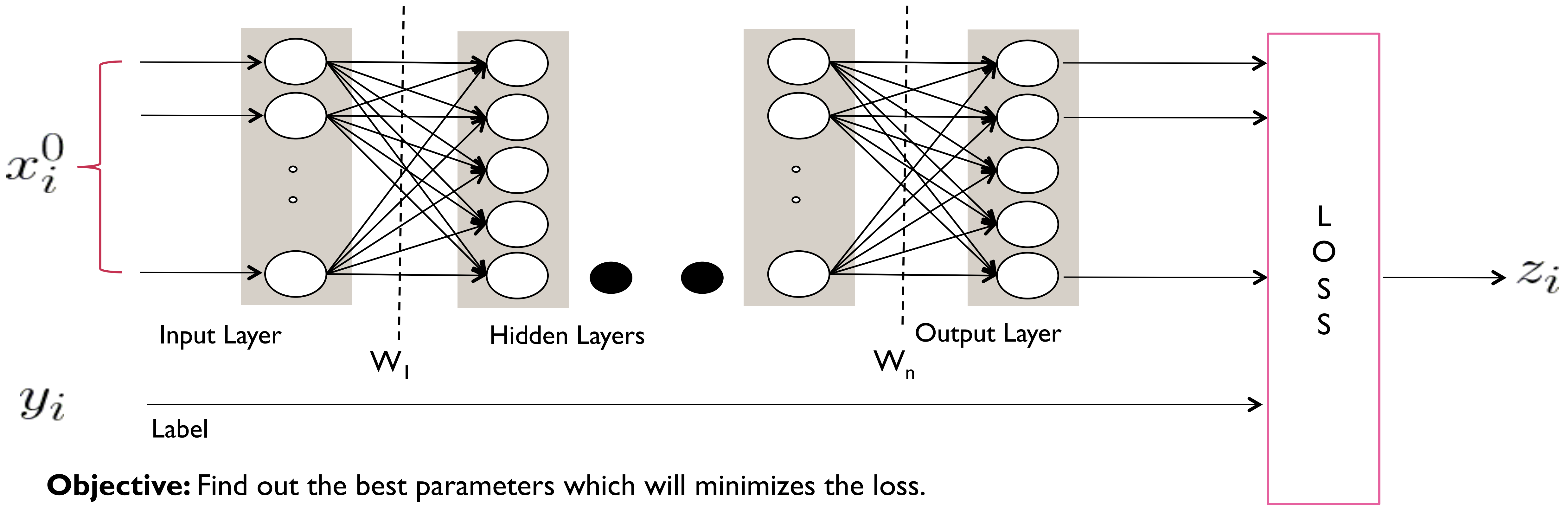


E.g. Sigmoid Activation Function



Multi Layer Perceptron

Loss or Objective

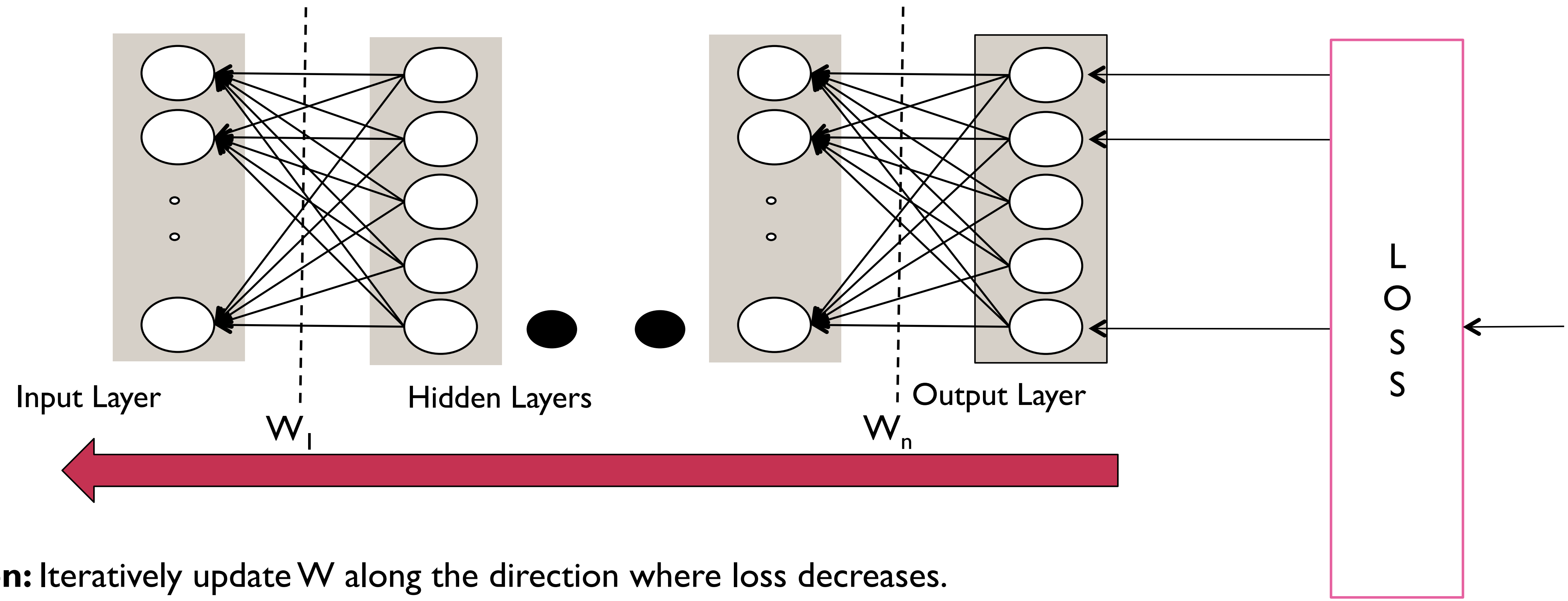


Objective: Find out the best parameters which will minimizes the loss.

$$W^* = arg\min_{W} \sum_{i=1}^{N} L(x_i^n, y_i; W) \longrightarrow \boxed{\text{Weight Vector}}$$

$$z_i = \frac{1}{2} \| x_i^n - y_i \|_2^2 \quad \text{E.g. Squared Loss}$$

Back propagation



Solution: Iteratively update W along the direction where loss decreases.

Each layer weights are updated based on the derivative of its output w.r.t. input and weights

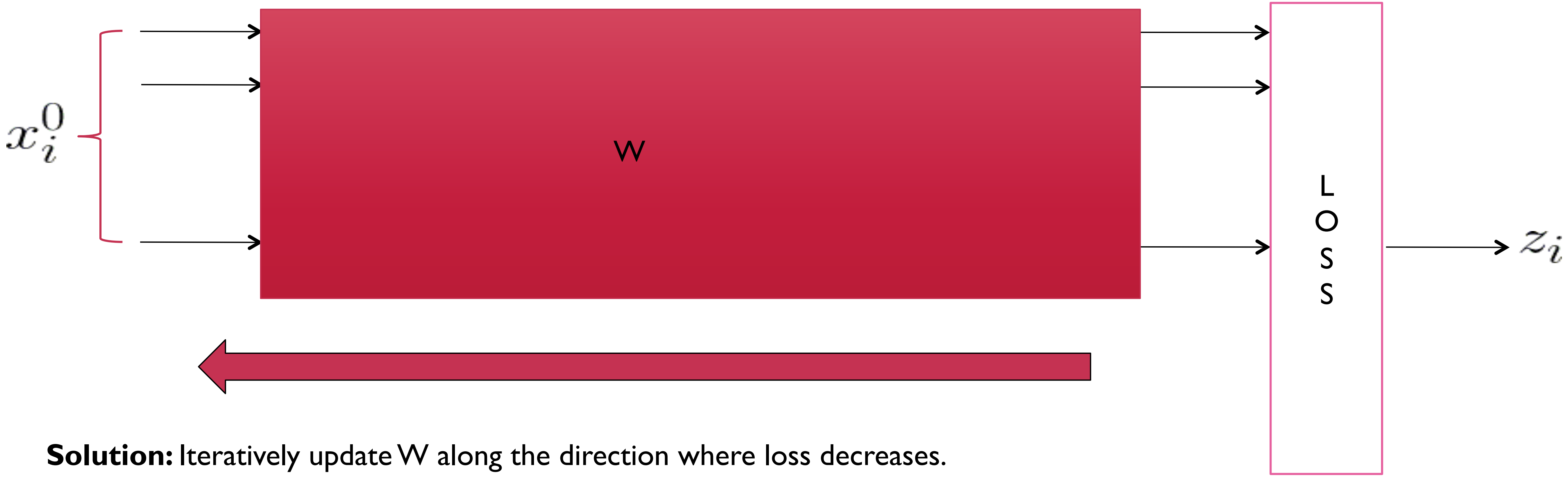
Loss or Objective

Objective: Find out the best parameters which will minimizes the loss.

$$W^* = arg\min_W \sum_{i=1}^{N} L(x_i^n, y_i; W)$$ Weight Vector

$$z_i = \frac{1}{2} \parallel x_i^n - y_i \parallel_2^2$$ E.g. Squared Loss

Back Propagation



Solution: Iteratively update W along the direction where loss decreases.

Each layer weights are updated based on the derivative of its output w.r.t. input and weights

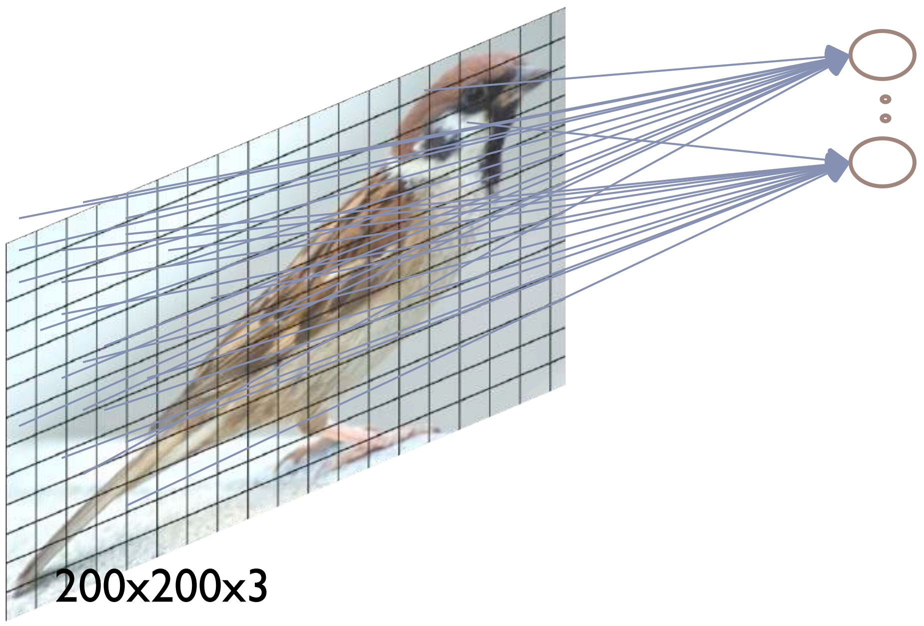
DL Module (M3)

- M3L1: DL Architectures
  - Peep into CNNs and RNNs
- M3L2: DL Architectures
  - More on RNNs, CNNs and others
- M3L3: Training
  - Nuances of Back Propagation
- M3L4: DL Applications -1
- M3L5: DL Applications -2

Convolutional Neural Networks (CNNs)

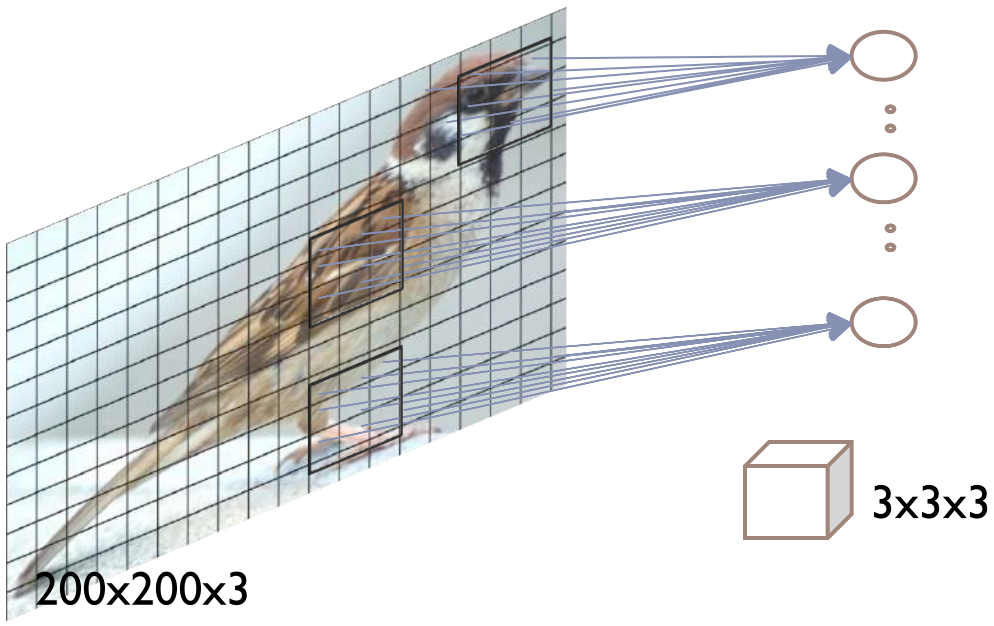
Convolutional Neural Network (CNN)

Fully connected layer



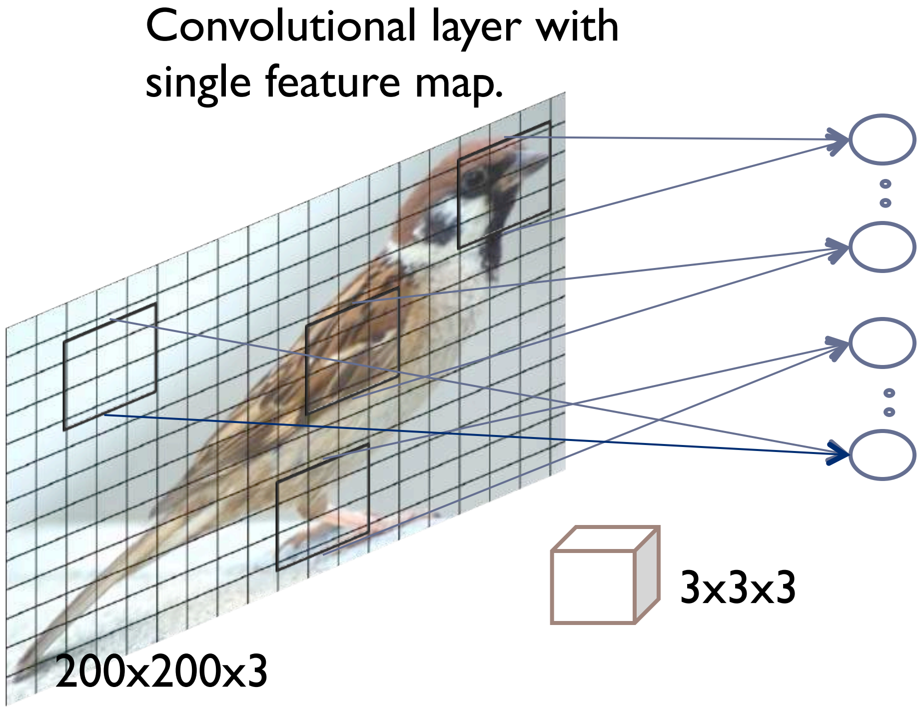
- Image of size 200 X 200 and 3 colours (RGB)
- #Hidden Units: 120,000 (=200X200X3)
- #Params: 14.4 billion (=120K X 120K)
- Need huge training data to prevent over-fitting!

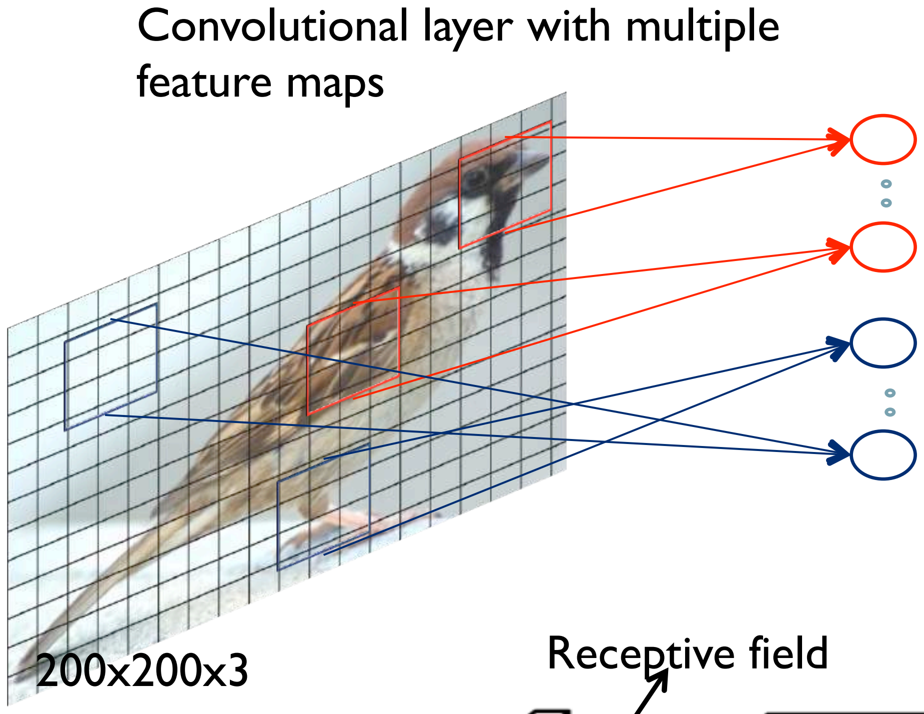
Locally connected layer



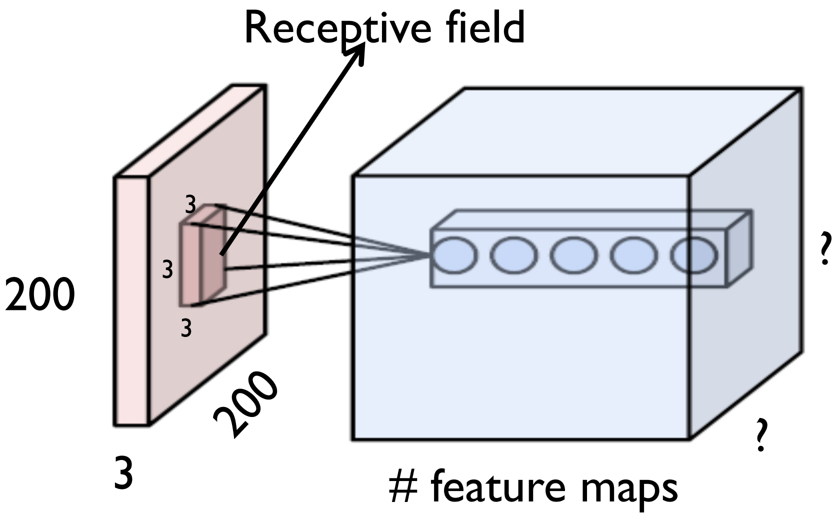
- #Hidden Units: 120,000
- #Params: 3.2 Million (=120K X 27)
- Useful when the image is highly registered

Convolutional Neural Network

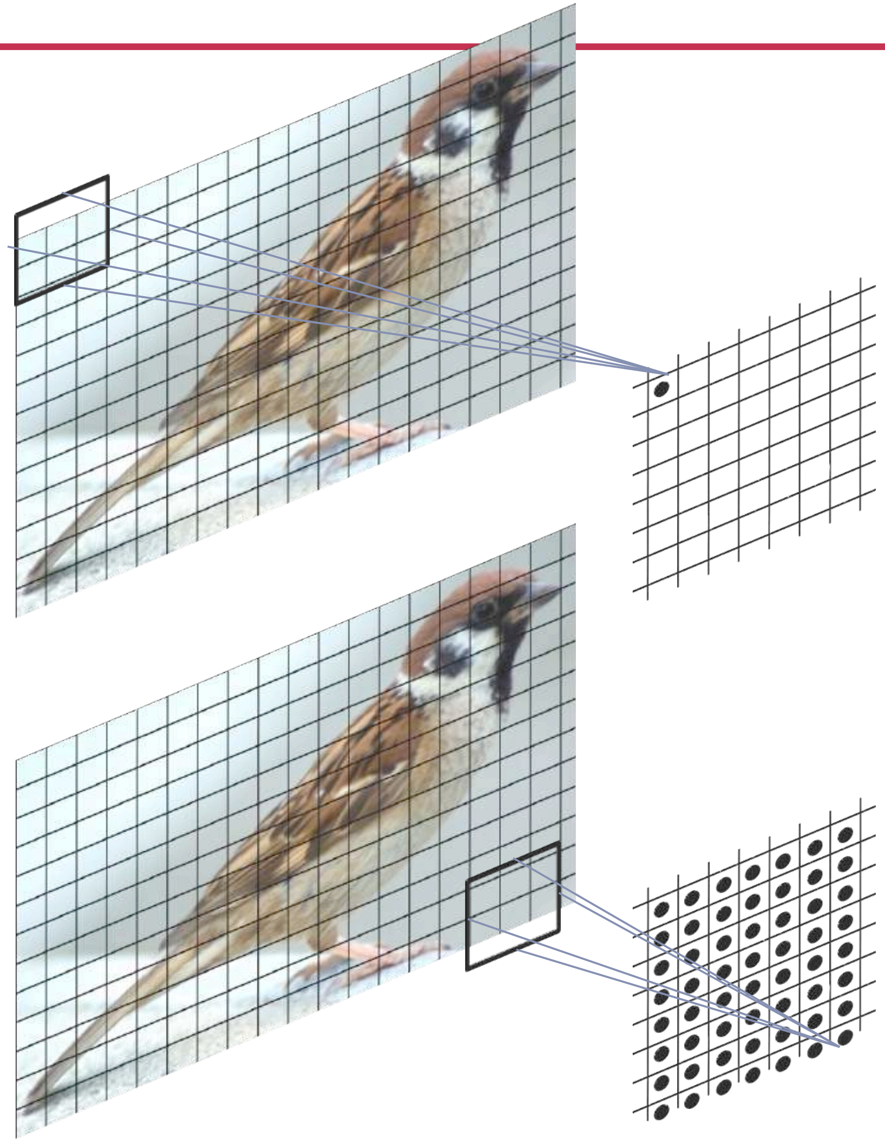




- #Hidden Units: 120,000
- #Params: 27 x #Feature Maps
- Sharing parameters
- Exploiting the stationarity property and preserves locality of pixel dependencies



Convolutional Neural Network



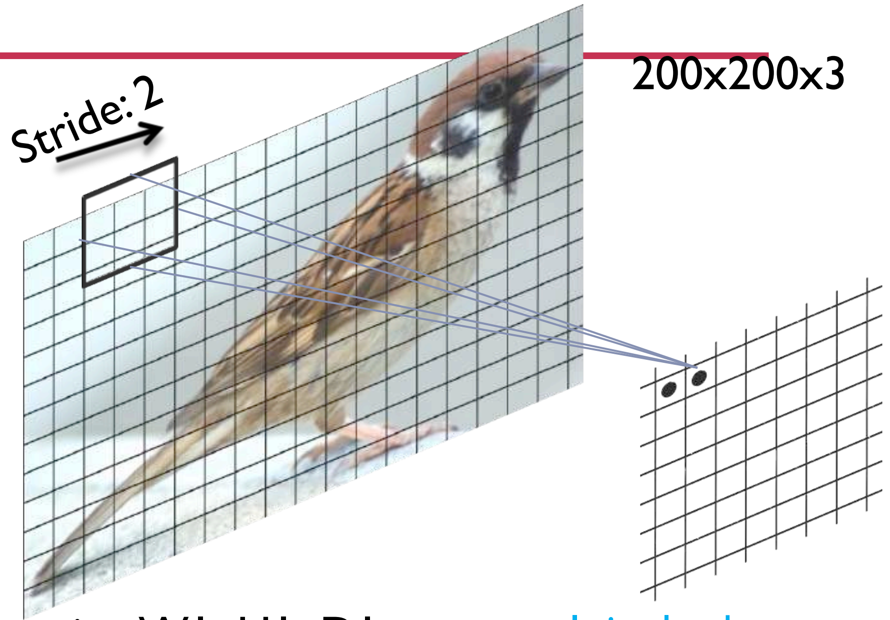
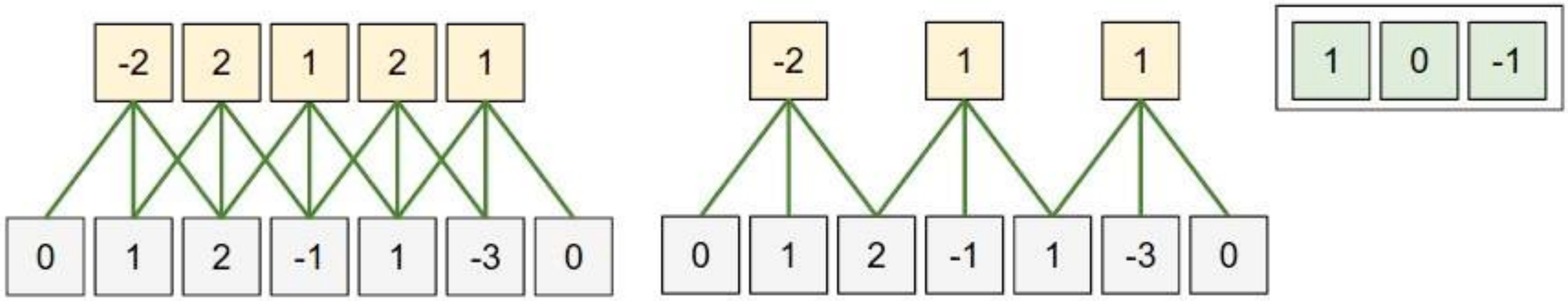


Image size: W1xH1xD1
Receptive field size: FxF
#Feature maps: K

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

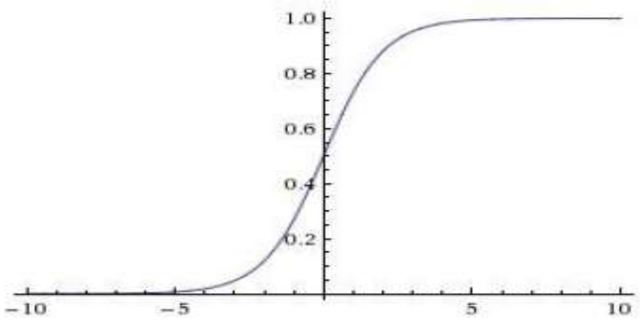
It is also better to do zero padding t
preserve input size spatially.

1D Example



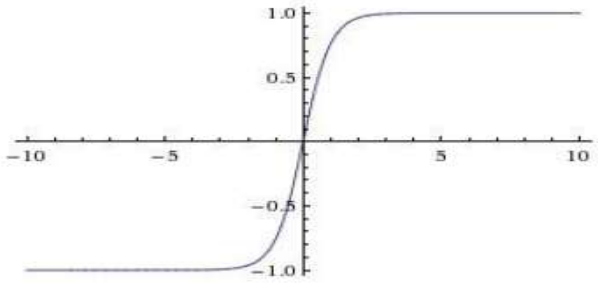
Activation Functions

Sigmoid



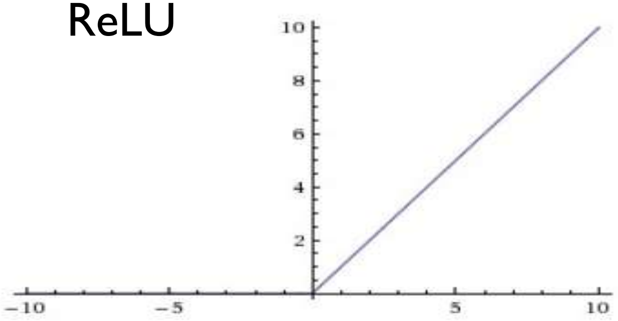
$$y = \frac{1}{1 + e^{-x}}$$

tanh



$$y = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

ReLU

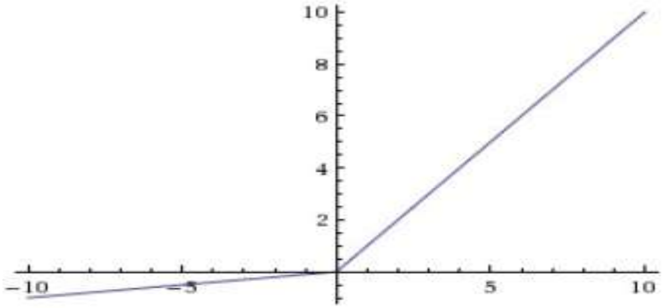


$$y = max(0, x)$$

maxout

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

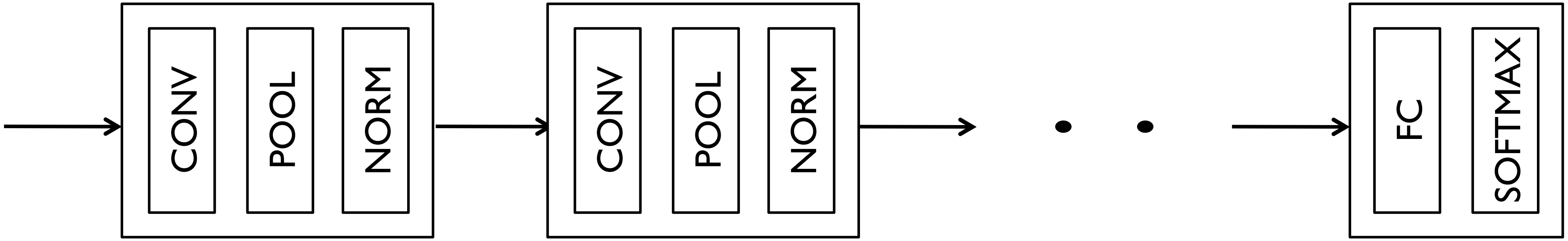
Leaky ReLU



$$y = \begin{cases} x & \text{if } x < 0 \\ 0.01x & \text{if } otherwise \end{cases}$$

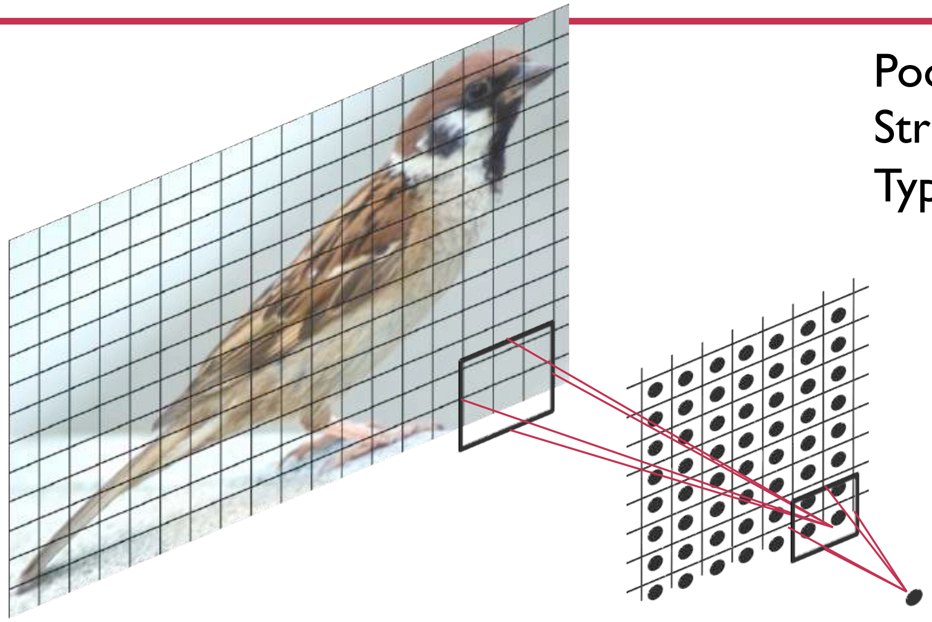
Typical Architecture

- A typical deep convolutional network



- Other layers
  - Pooling
  - Normalization
  - Fully connected
  - etc.

Pooling Layer



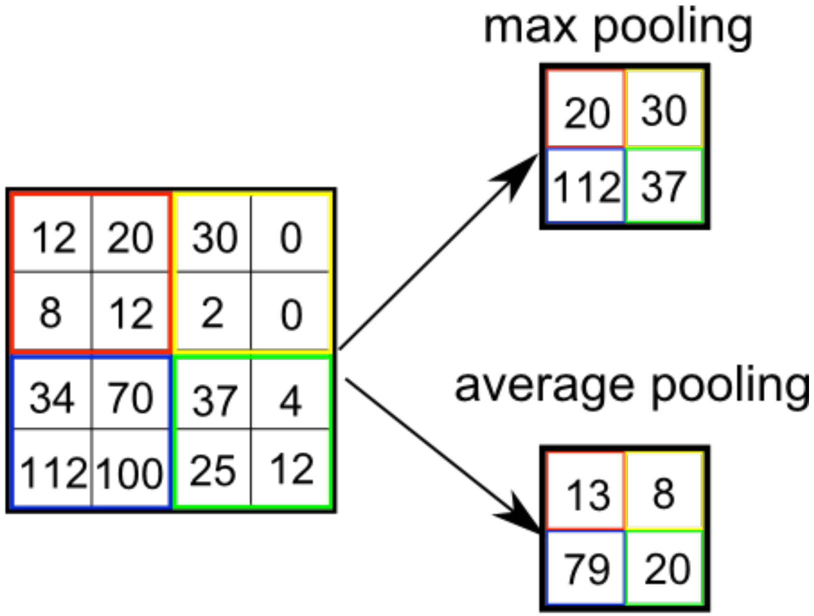
Pool Size: 2x2
Stride: 2
Type: Max

2	8	9	4
3	6	5	7
3	1	6	4
2	5	7	3

→

8	9
5	7

Max pooling



- Role of an aggregator.
- Invariance to image transformation and increases compactness to representation.
- Pooling types: Max, Average, L2 etc.

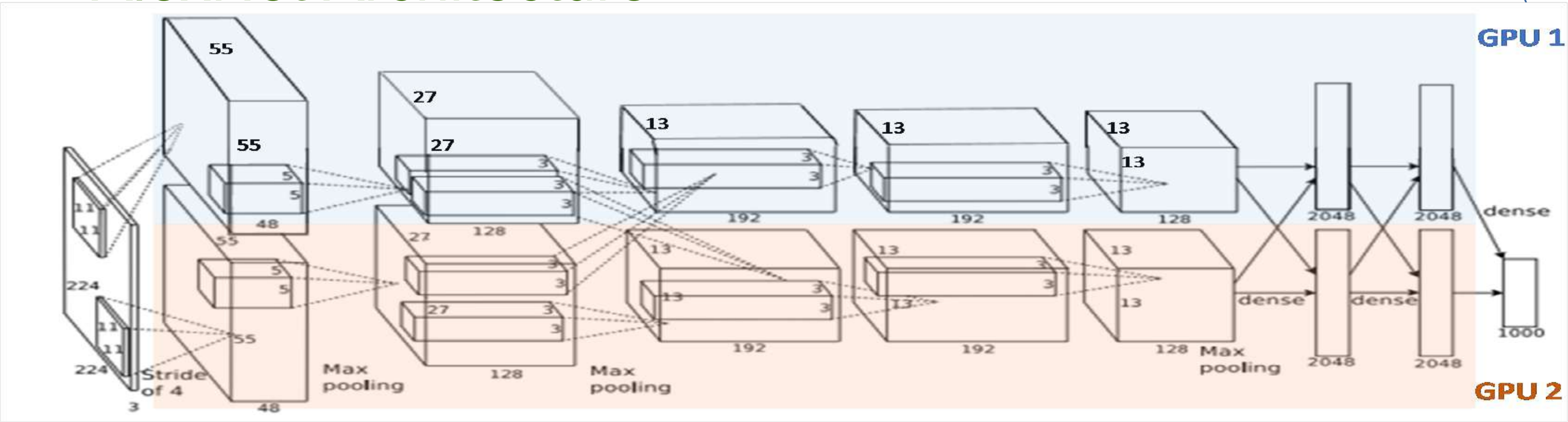
Fully connected

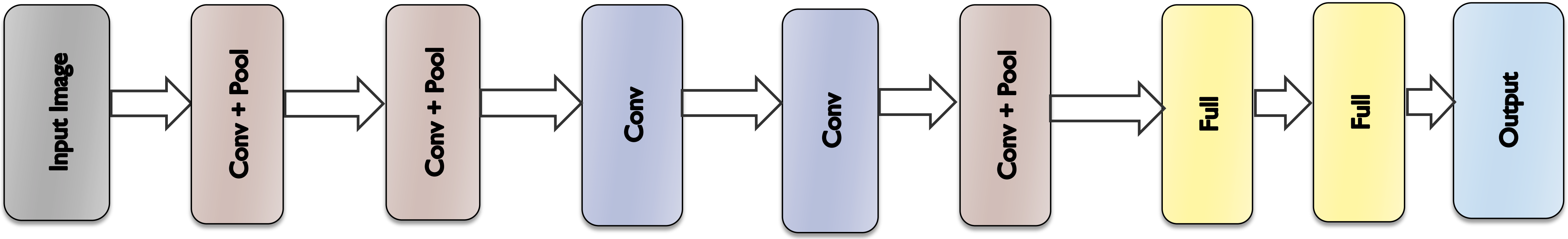
- Multi layer perceptron
- Role of a classifier
- Generally used in final layers to classify the object represented in terms of discriminative parts and higher semantic entities.

- SoftMax
  - Normalizes the output.

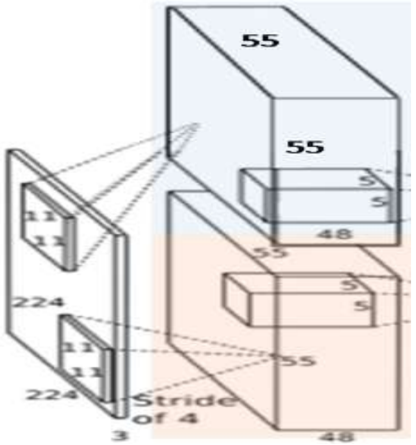
$$z_n = \frac{e^{x_n}}{\sum_{i=1}^{K} e^{x_i}}$$

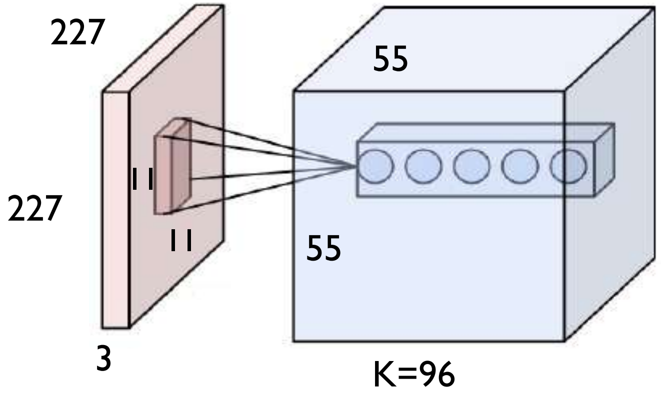
AlexNet Architecture





Parameter Calculation





- Filter Size F
- Input volume streams be D
- **#** filters be K
- **#** parameters in a layer is $\boldsymbol{(F . F . D) . K}$

Hyper parameters

Example:

For layer 1, Input images are 227 x 227 x 3

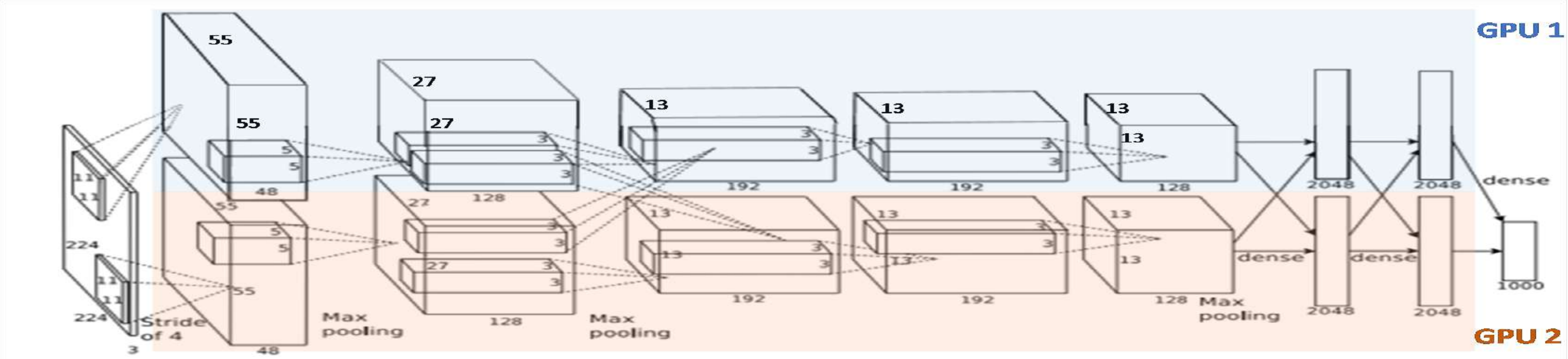
- F = 11 and K = 96
- Each filter has 11 x 11 x 3 = 363 and 1 (bias) i.e., 364 weights
- # weights = 364 x 96 = **35 K** (approx.)

- Stride S
- Zero padding P
- Input Size: $W1$ x $H1$ x $D1$
- Output Size: $\boldsymbol{W2}$ **x** $\boldsymbol{H2}$ **x** $\boldsymbol{D2}$
- $W2 = [\,(W1 - F + 2P) / S\,] + 1$ and $D2 = K$

Hyper parameters

- S = 4, $W1$ = 227, F =11, P = 0 $D2$ = 96
- W2 = (227 -11)/4 + 1 = 55
- Output Size: **55 x 55 X 96**

AlexNet Architecture

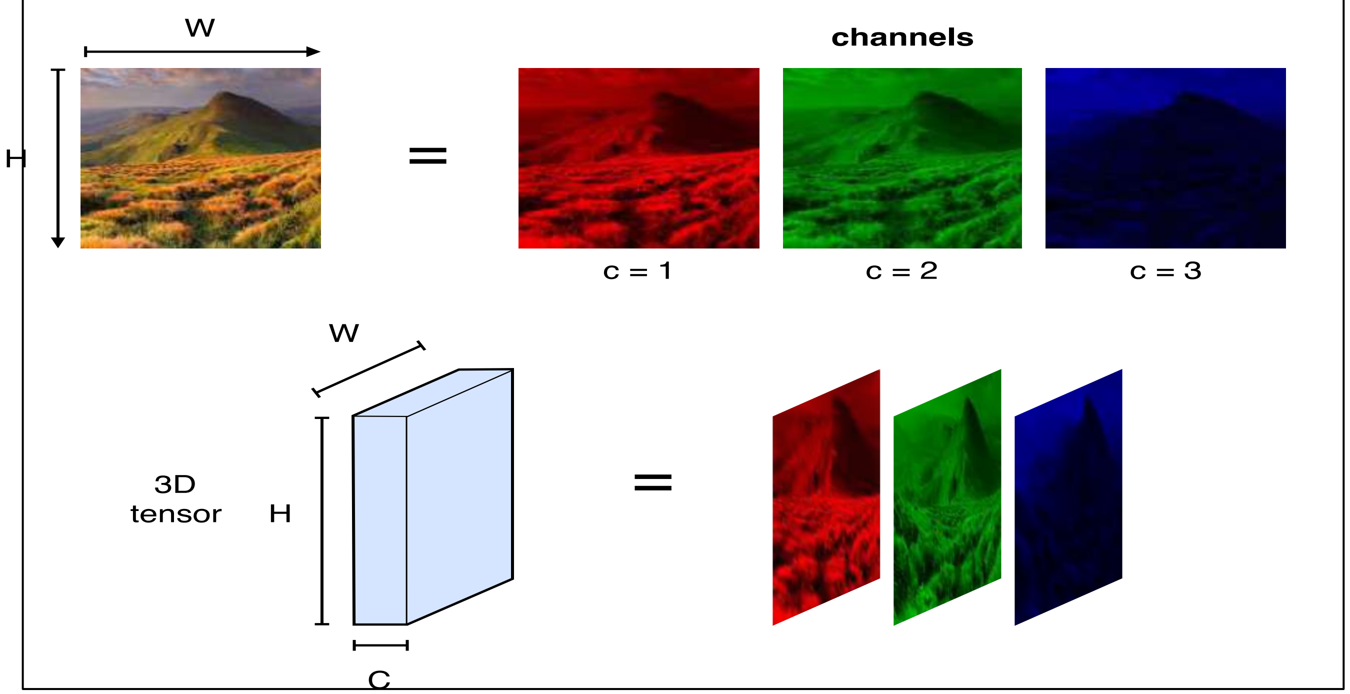


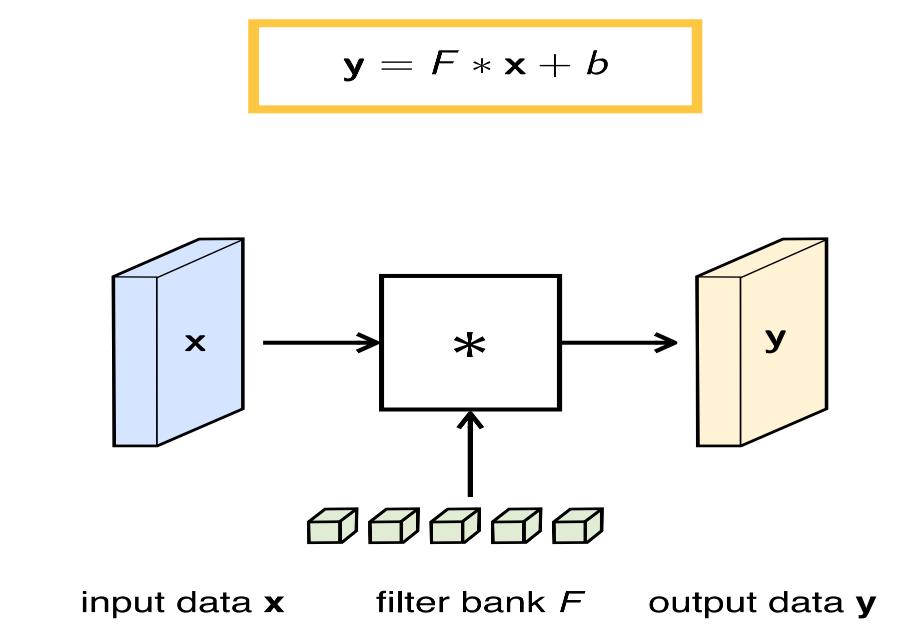
About 57 M parameters are in the fully connected layers

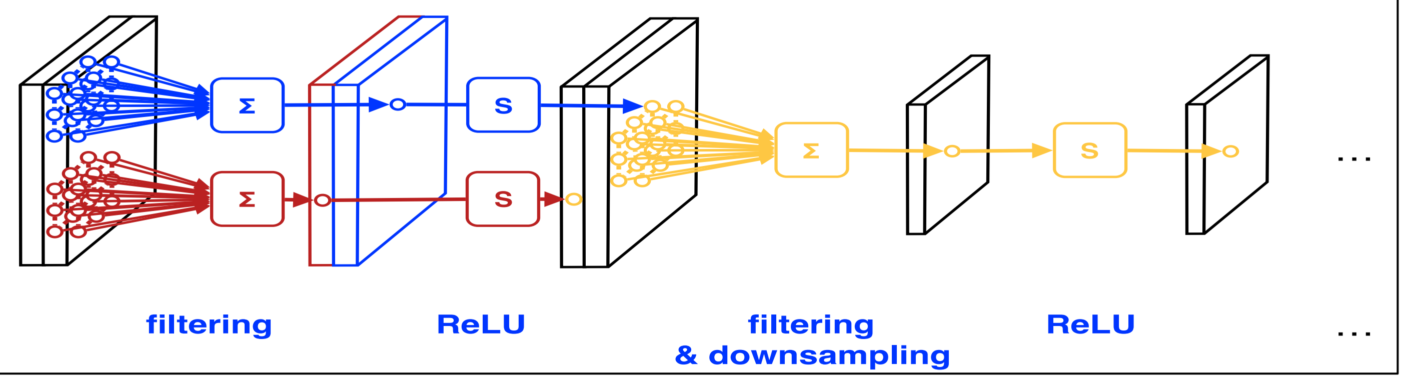
									Total
Parameters :	[(11 x 11 x 3) + 1] x 96 = 35 K	[5 x 5 x 48] x 256 = 307 K	[3 x 3 x 256] x 384 = 884 K	663 K	442 K	37 M	16 M	4 M	60 M
Neurons :	253,440	27 x 27 x 256 = 186,624	13 x 13 x 384 = 64,896	13 x 13 x 384 = 64,896	13 x 13 x 256 = 43,264	4096	4096	1000	0.63 M

- Convolutional layers cumulatively contain about 90-95% of computation, only about 5% of the parameters
- Fully-connected layers contain about 95% of parameters.

Summary of CNNs







Introduction to RNNs

More in the next lecture

Why RNNs?

- Intelligent systems (Networks) need memory.
- Many inputs are sequential in nature.
- Concepts have long term dependencies.
  - Not just one or two steps backwards.
  - Eg. What controls tomoto price of tomorrow?
- Popular networks (eg. CNNs) do not have cycles.

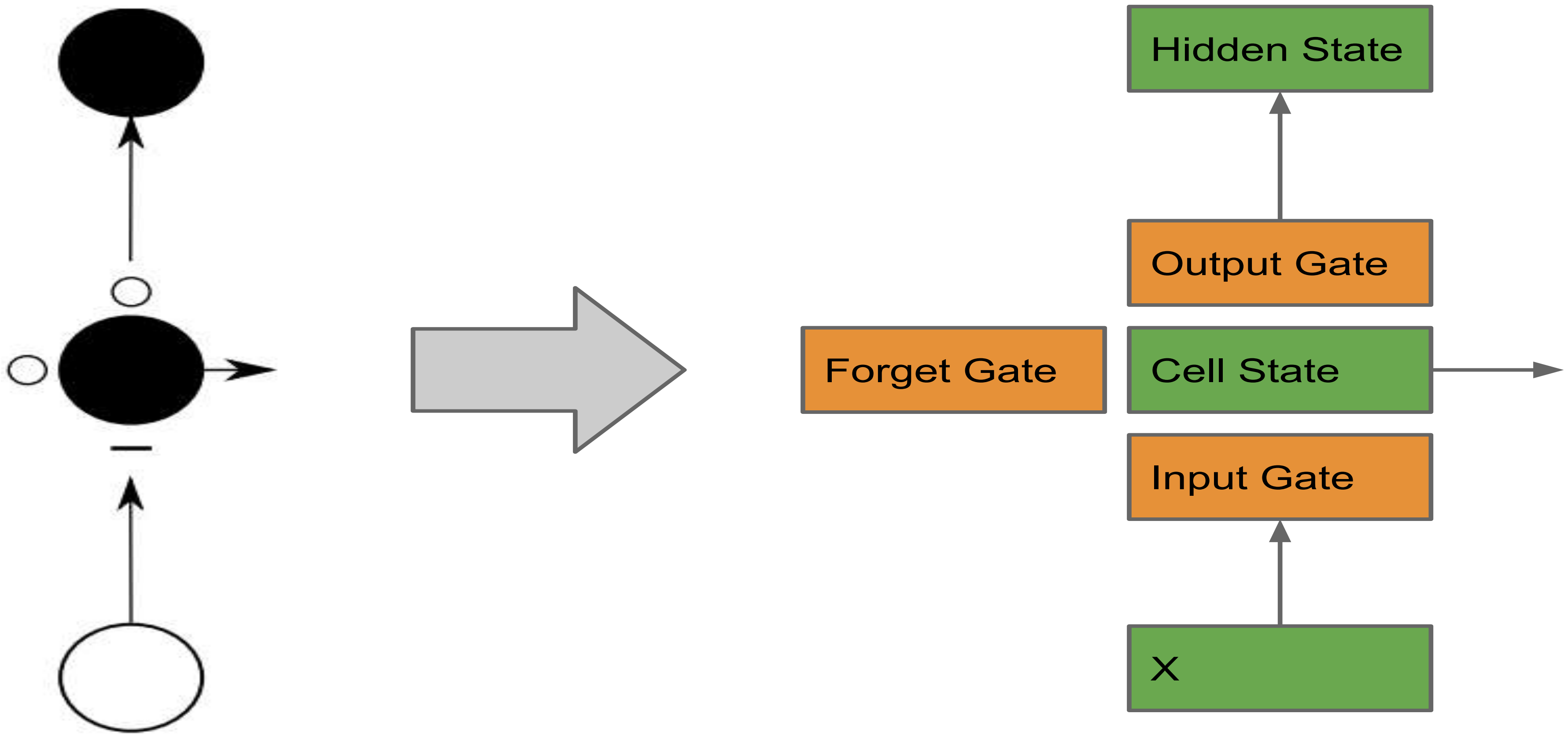
Generating poetry with RNNs

Sonnet 116 – Let me not ...

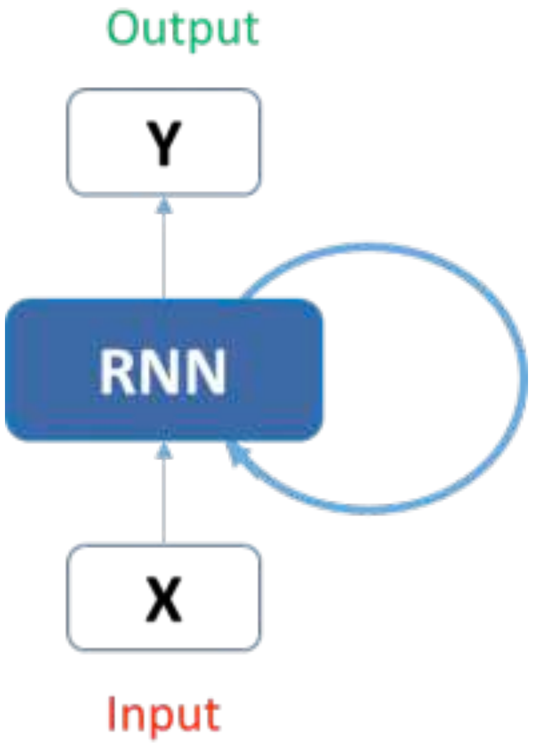
by William Shakespeare

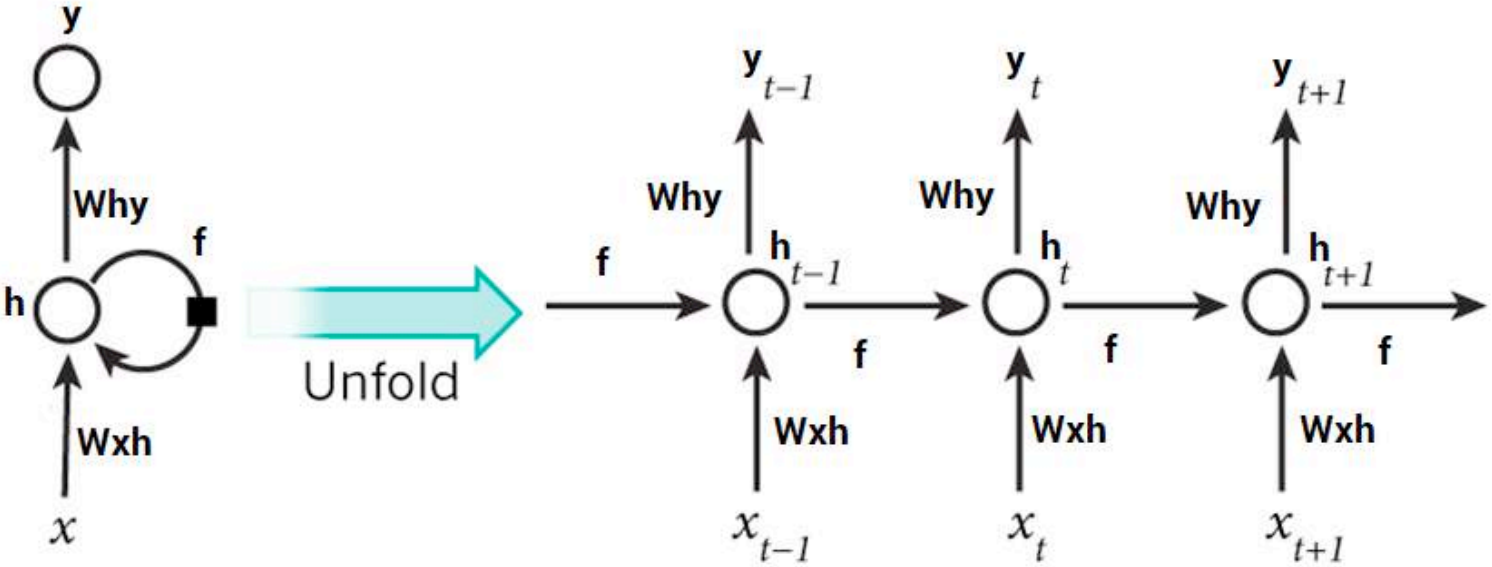
Let me not to the marriage of true minds
        Admit impediments. Love is not love
Which alters when it alteration finds,
        Or bends with the remover to remove:
O no! it is an ever–fixed mark
        That looks on tempests and is never shaken;
It is the star to every wandering bark,
        Whose worth's unknown, although his height be taken.
Love's not Time's fool, though rosy lips and cheeks
        Within his bending sickle's compass come:
Love alters not with his brief hours and weeks,
        But bears it out even to the edge of doom.
If this be error and upon me proved,
        I never writ, nor no man ever loved.

Cell with Memory



RNNs: Overview and Summary

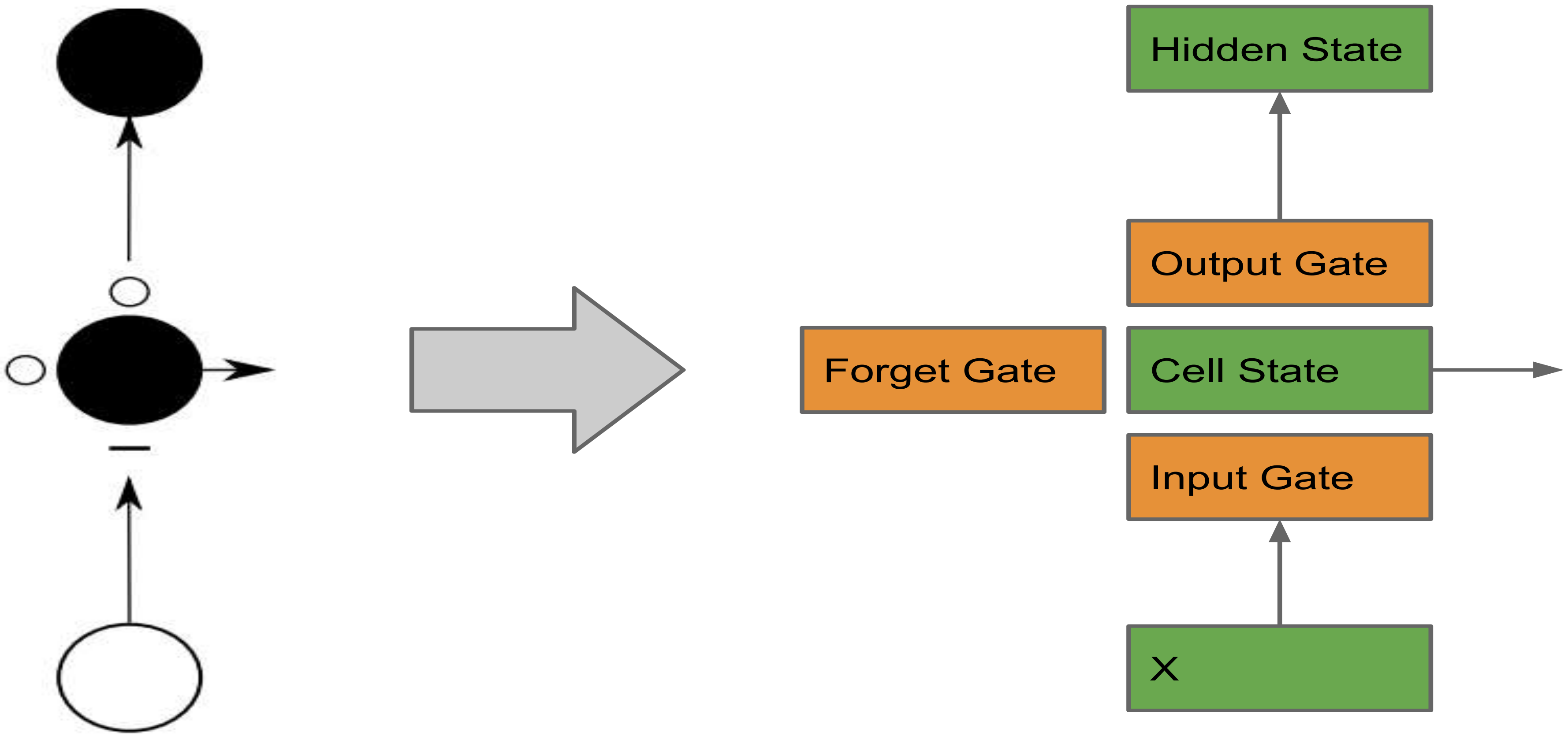




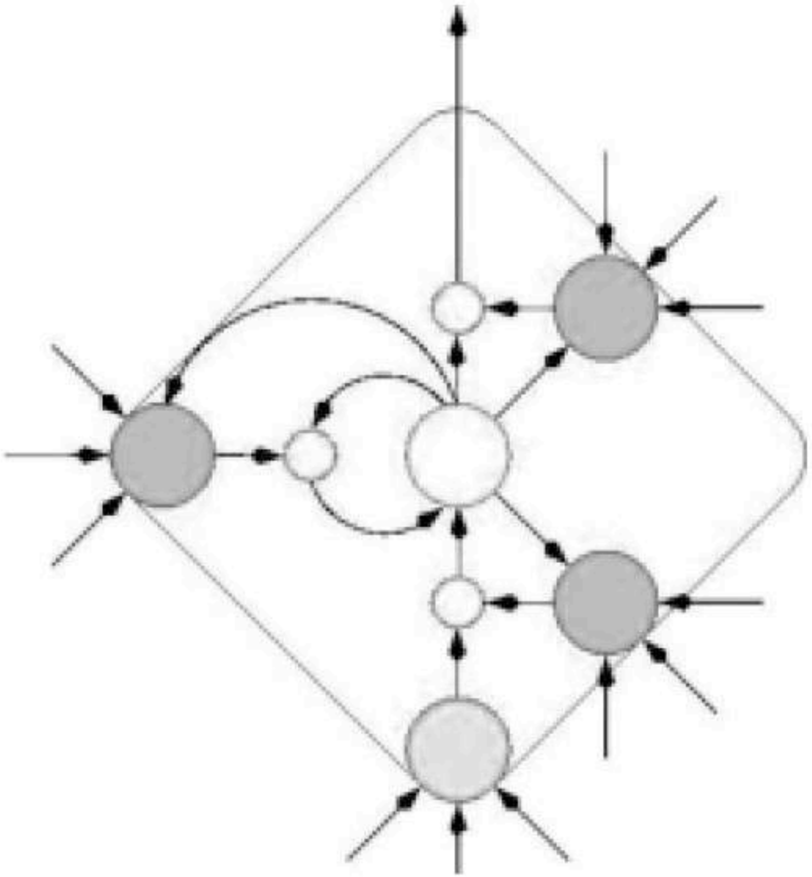
$$h_t = \tanh\left(W_{hh}h_{t-1} + W_{xh}x_t\right)$$

$$y_t = W_{hy}h_t$$

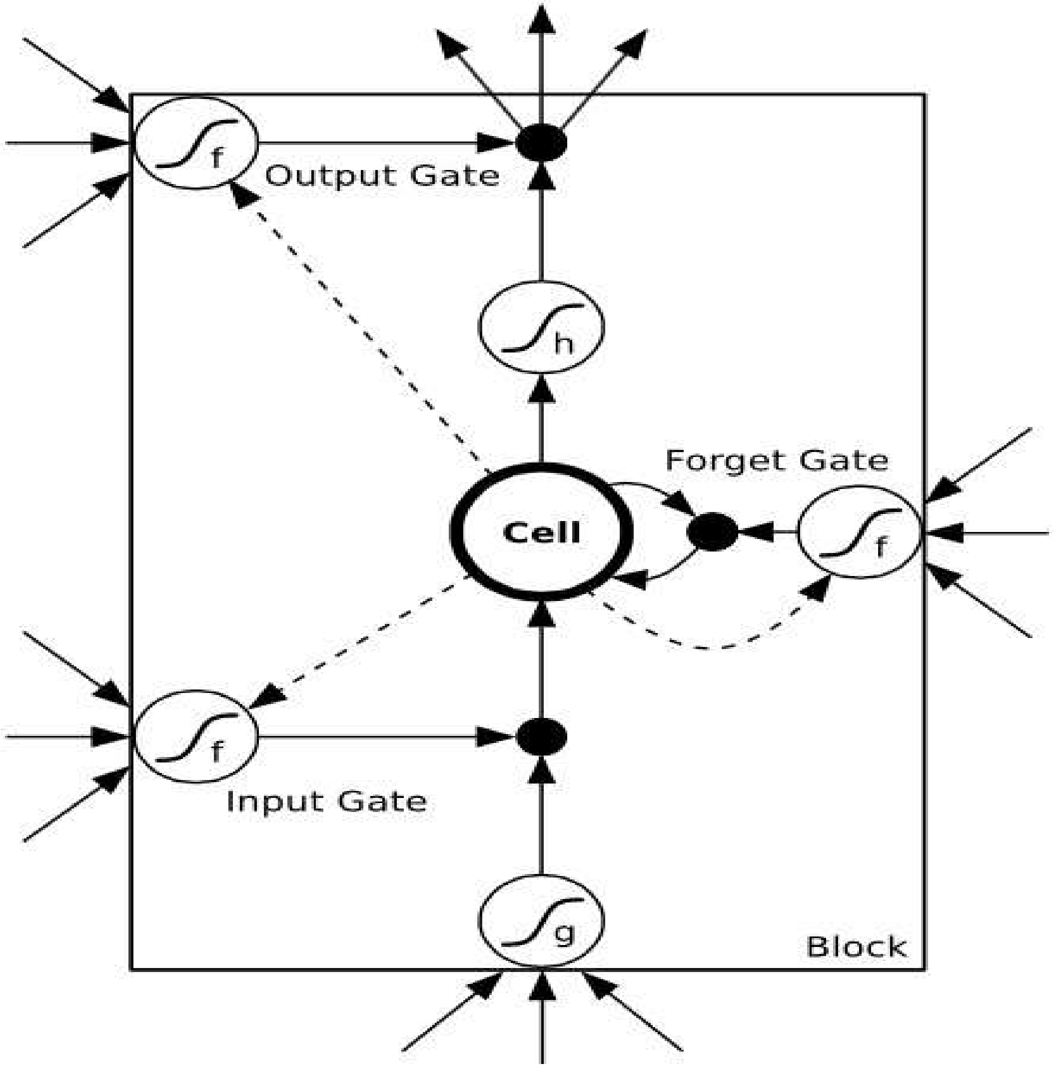
LSTM Node


Hidden State
Output Gate
Forget Gate
Cell State
Input Gate
X

Gates/Switches are “controllable”

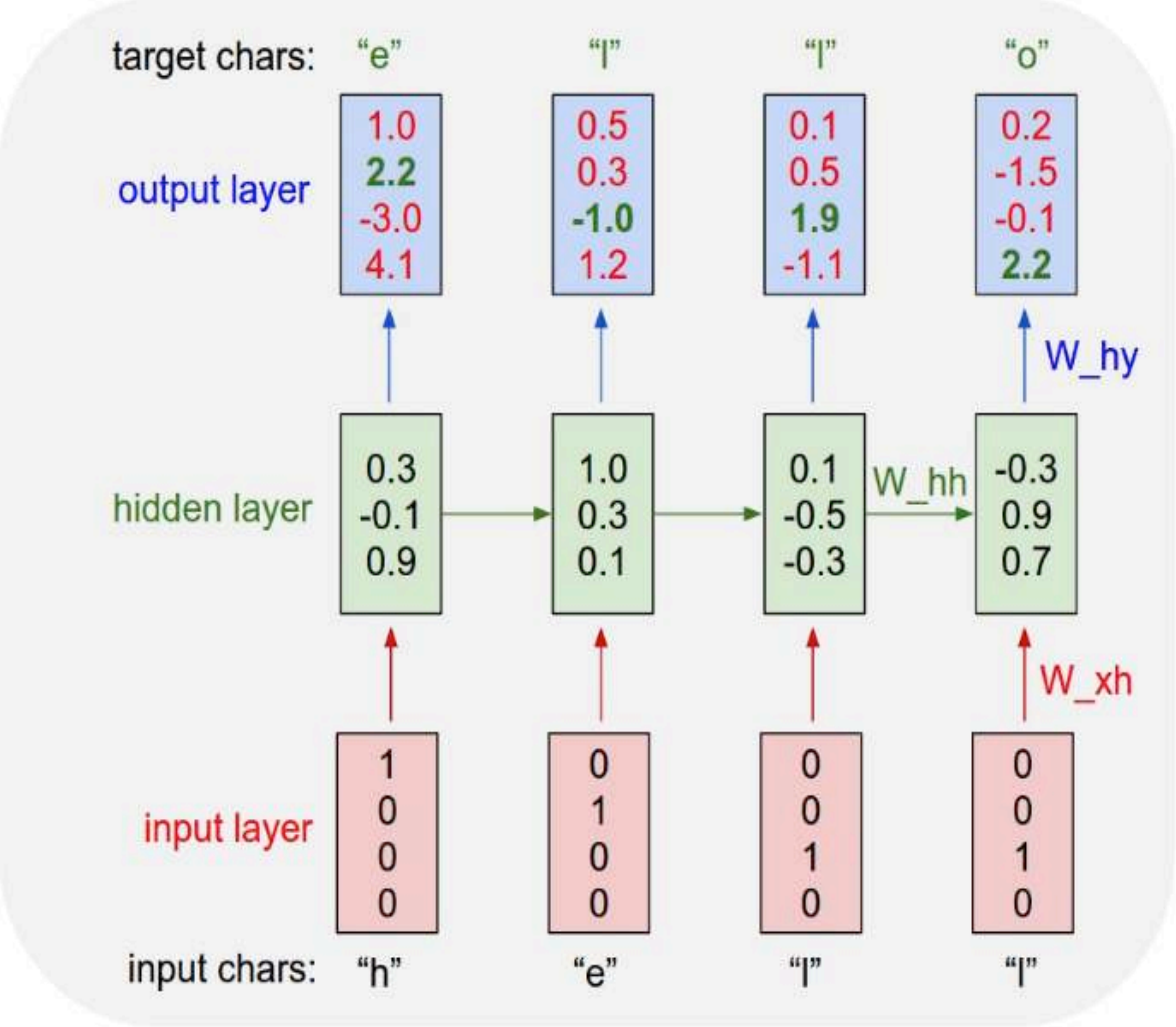


LSTM: Gates are “soft” controllable



Character Level Language Modelling

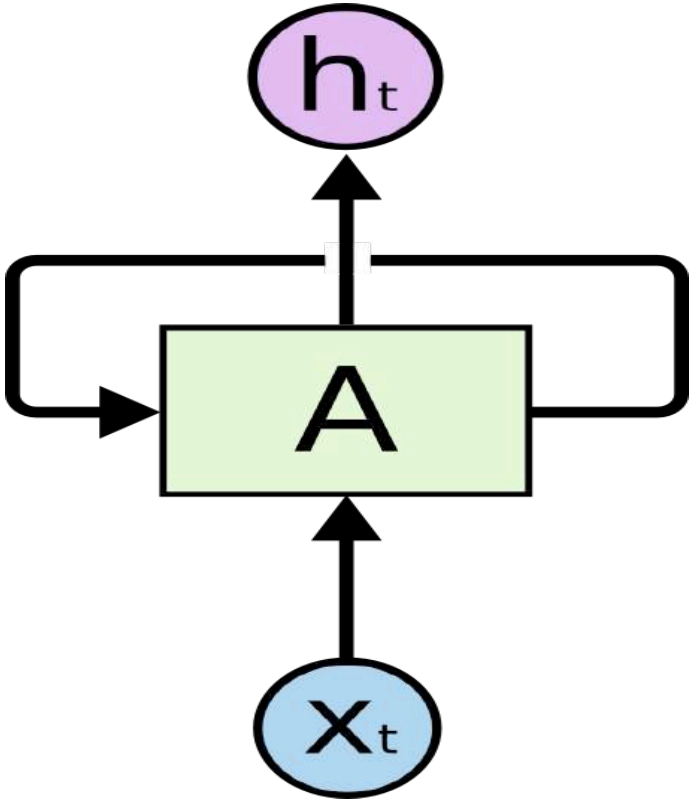
Task: Predicting the next character given the current character



Andrej Karpathy, Blog on "Unreasonable Effectiveness of Recurrent Neural Networks"

It may be fun today ☺

- RNNs




```
*
* Increment the size file of the new incorrect UI_FILTER
* of the size generatively.
*/
tatic int indicate_policy(void)

 int error;
 if (fd == MARN_EPT) {
   /*
    * The kernel blank will coeld it to userspace.
    */
   if (ss->segment < mem_total)
     unblock_graph_and_set_blocked();
   else
     ret = 1;
   goto bail;
 }
 segaddr = in_SB(in.addr);
 selector = seg / 16;
 setup_works = true;
 for (i = 0; i < blocks; i++) {
   seq = buf[i++];
   bpf = bd->bd.next + i * search;
   if (fd) {
     current = blocked;
```

A. Karpathy http://karpathy.github.io/2015/05/21/rnn-effectiveness/

and Shakespeare

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

and Algebraic Geometry!!

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ over U compatible with the complex

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S. Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) *f is locally of finite type. Since $S = \mathrm{Spec}(R)$ and $Y = \mathrm{Spec}(R)$.*

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\ldots,0}$.

Lemma 0.2. *Let X be a locally Noetherian scheme over S, $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \overline{A}_2$ works.*

Lemma 0.3. *In Situation ??. Hence we may assume $\mathfrak{q}' = 0$.*

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

100 th iteration

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

↓ train more

300 th iteration

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

↓ train more

700 th iteration

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

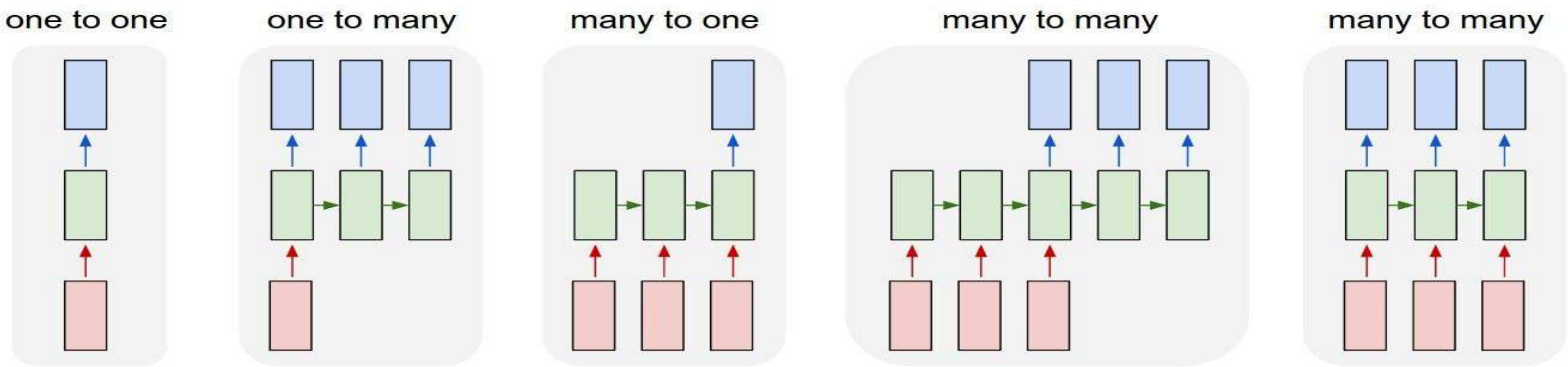
↓ train more

2000 th iteration

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

Andrej Karpathy, Blog on “Unreasonable Effectiveness of Recurrent Neural Networks”

Recurrent Networks offer a lot of flexibility:



Vanilla neural networks

e.g. **image captioning**
image -> sequence of words

e.g. **sentiment classification**
sequence of words -> sentiment

e.g. **machine translation**
seq of words -> seq of words

e.g. **video classification on frame level**

Summary

- RNNs are powerful networks
  - With feedback
  - With memory
  - Hard to “fully” understand.
- CNNs are very useful for a class of tasks
  - Both in Image and Text.
- Shall revisit them again.

Thanks. Questions?

Vanishing Gradients

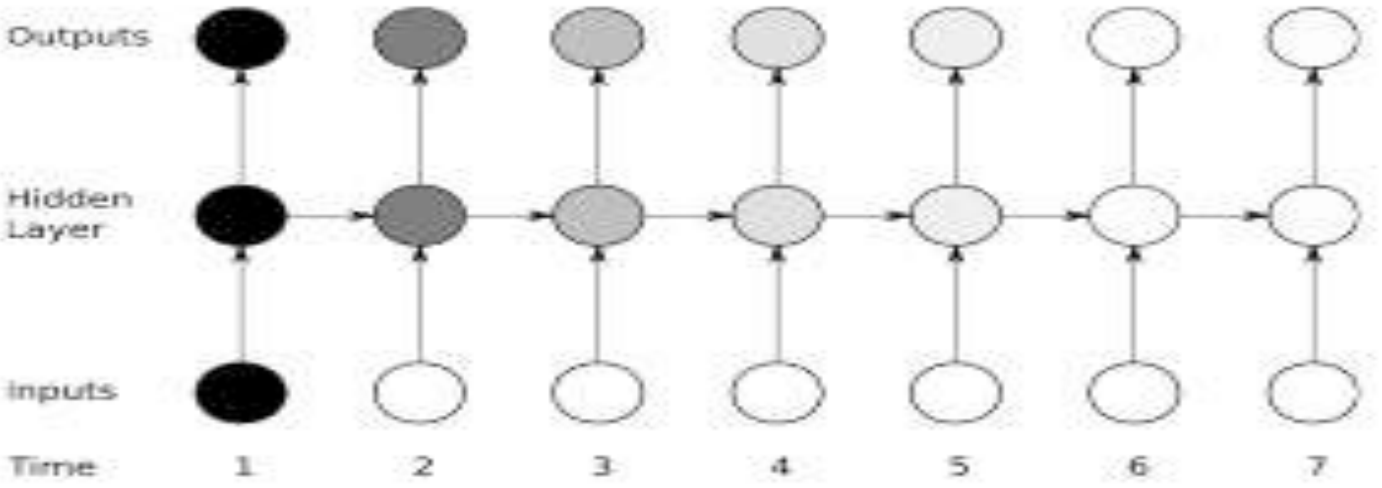


Figure 4.1: **The vanishing gradient problem for RNNs.** The shading of the nodes in the unfolded network indicates their sensitivity to the inputs at time one (the darker the shade, the greater the sensitivity). The sensitivity decays over time as new inputs overwrite the activations of the hidden layer, and the network 'forgets' the first inputs.

Vanishing Gradients

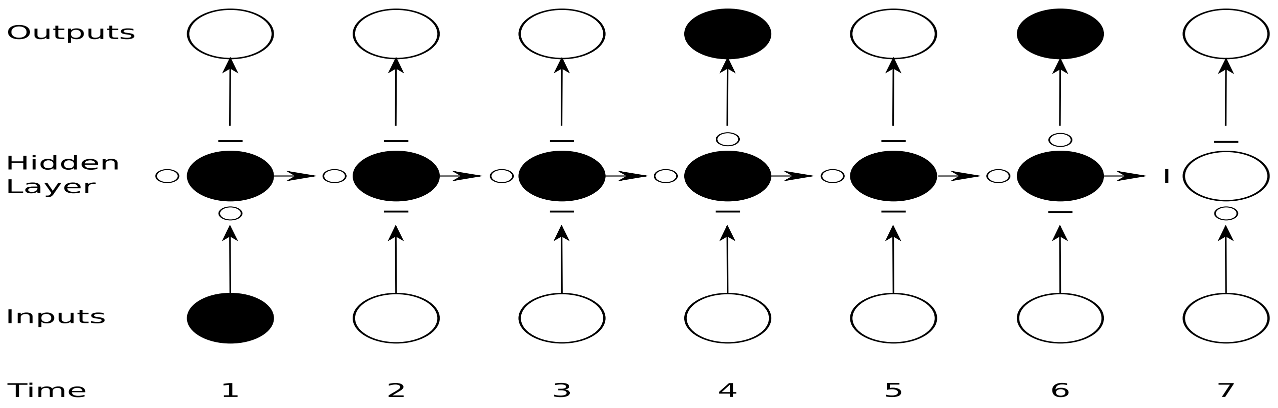


Figure 4.4: **Preservation of gradient information by LSTM.** As in Figure 4.1 the shading of the nodes indicates their sensitivity to the inputs at time one; in this case the black nodes are maximally sensitive and the white nodes are entirely insensitive. The state of the input, forget, and output gates are displayed below, to the left and above the hidden layer respectively. For simplicity, all gates are either entirely open ('O') or closed ('—'). The memory cell 'remembers' the first input as long as the forget gate is open and the input gate is closed. The sensitivity of the output layer can be switched on and off by the output gate without affecting the cell.

Vanishing Gradients

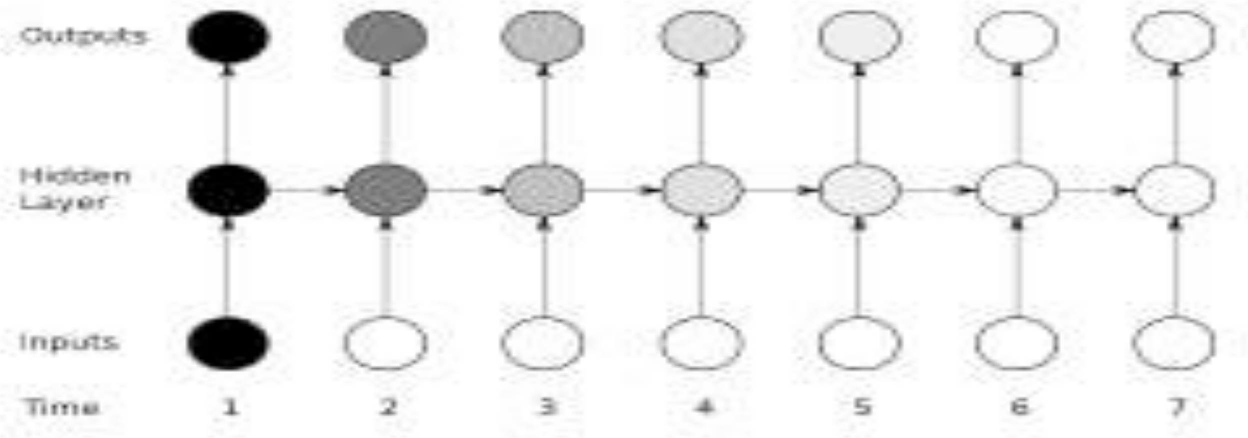
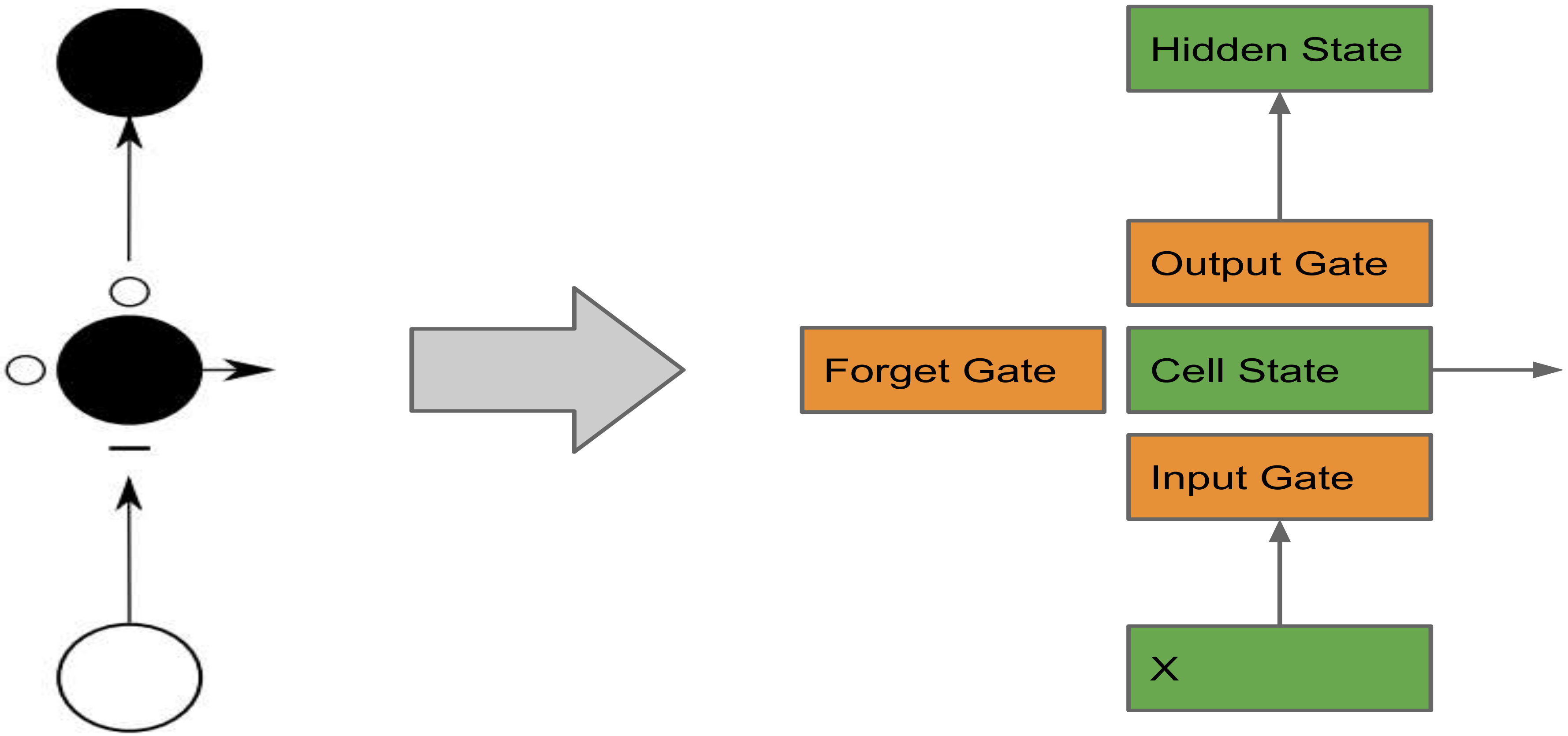


Figure 4.1: **The vanishing gradient problem for RNNs.** The shading of the nodes in the unfolded network indicates their sensitivity to the inputs at time one (the darker the shade, the greater the sensitivity). The sensitivity decays over time as new inputs overwrite the activations of the hidden layer, and the network 'forgets' the first inputs.

LSTM Node


Hidden State
Output Gate
Forget Gate
Cell State
Input Gate
X

Vanishing Gradients

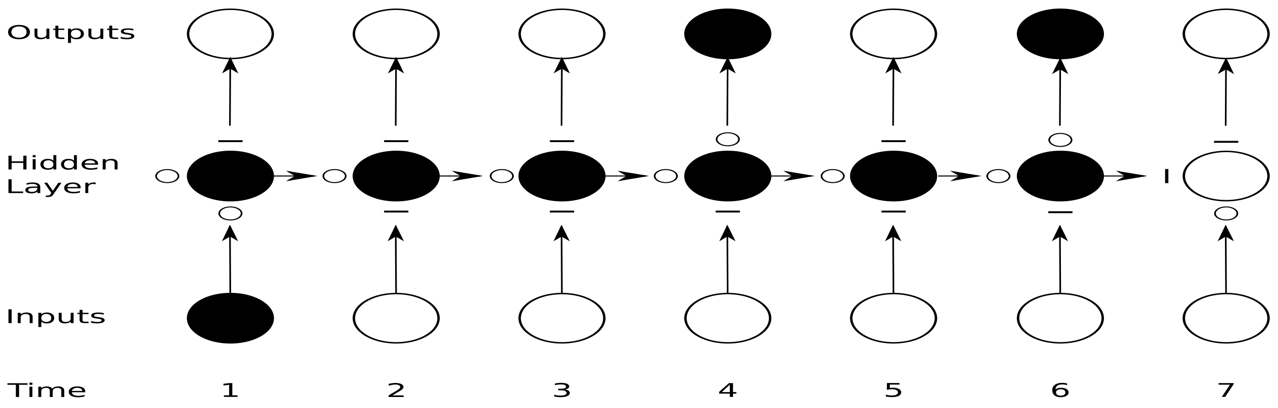
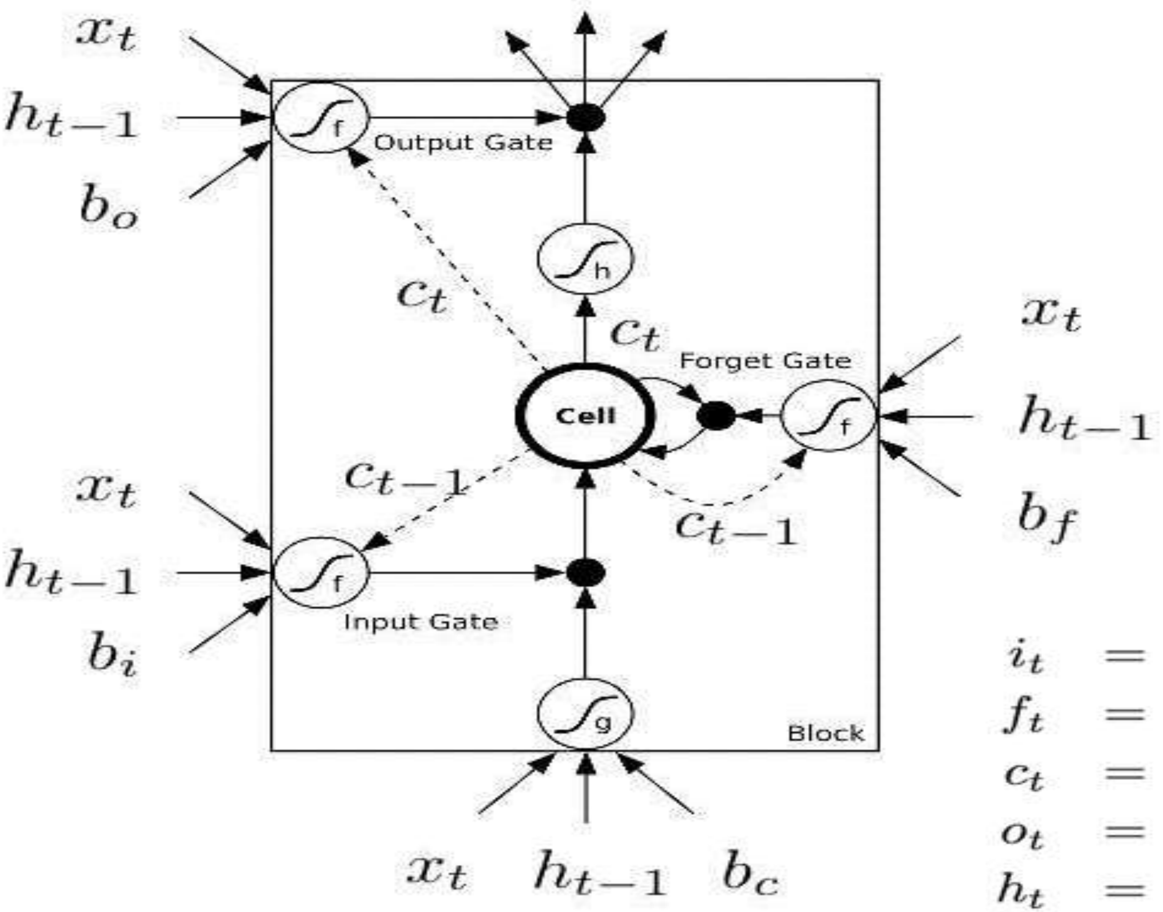


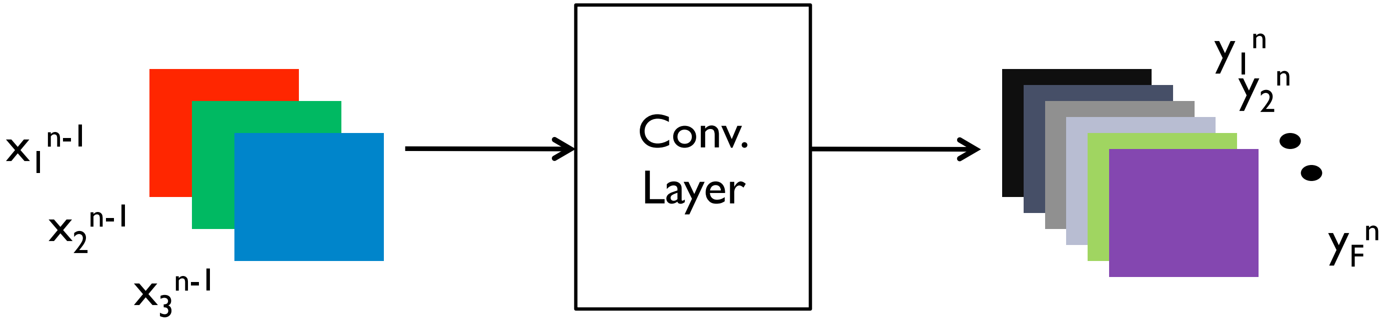
Figure 4.4: **Preservation of gradient information by LSTM.** As in Figure 4.1 the shading of the nodes indicates their sensitivity to the inputs at time one; in this case the black nodes are maximally sensitive and the white nodes are entirely insensitive. The state of the input, forget, and output gates are displayed below, to the left and above the hidden layer respectively. For simplicity, all gates are either entirely open ('O') or closed ('—'). The memory cell 'remembers' the first input as long as the forget gate is open and the input gate is closed. The sensitivity of the output layer can be switched on and off by the output gate without affecting the cell.

LSTM Node



$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t &= f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
h_t &= o_t tanh(c_t)
\end{aligned}
$$

Convolutional Layer

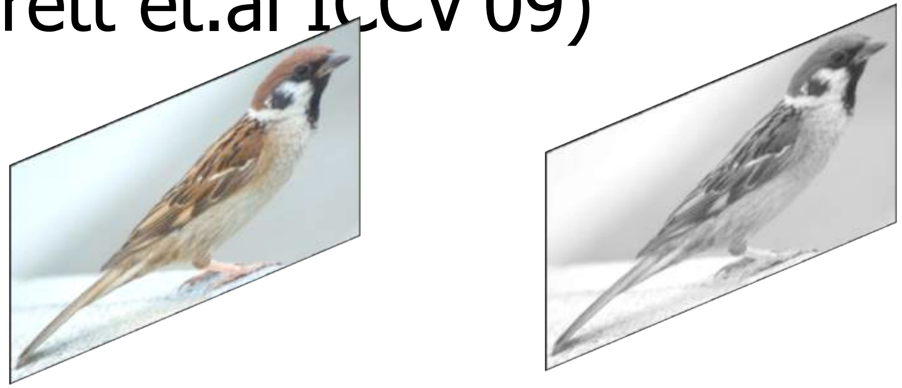


$$y_j^n = f\left(\sum_{k=1}^{F} x_k^{n-1} * w_{kj}^n\right)$$

Here "f" is a non-linear activation function.
F= no. of feature maps
n= layer index
"*" represents element-by-element multiplication

Normalization

- Local contrast normalization (Jarrett et.al ICCV'09)
  - Improves invariances
  - Improves sparsity



Need similar responses

- Local response normalization (Krizhevesky et.al. NIPS'12)
  - Kind of "lateral inhibition" and performed across the channels
- Batch normalization
  - Activation of the mini-batch is centered to zero-mean and unit variance to prevent internal covariate shifts.

Recurrent Neural Networks