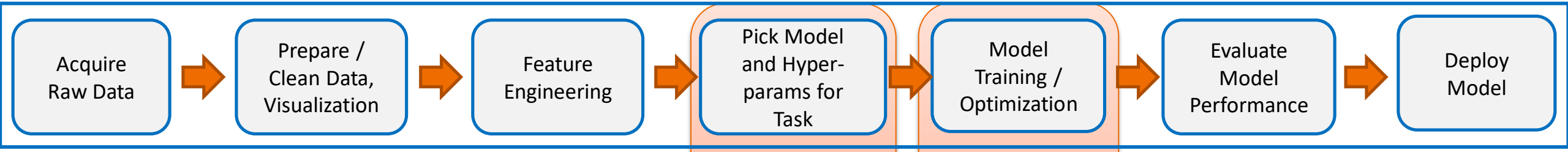
Review Lecture

Unit-3

Focus for this lecture



- Backpropagation
- Convolution Layer
- CNNs
- Auto Encoder

- Time Series
- Appreciating CNNs
- RNN
- LSTM

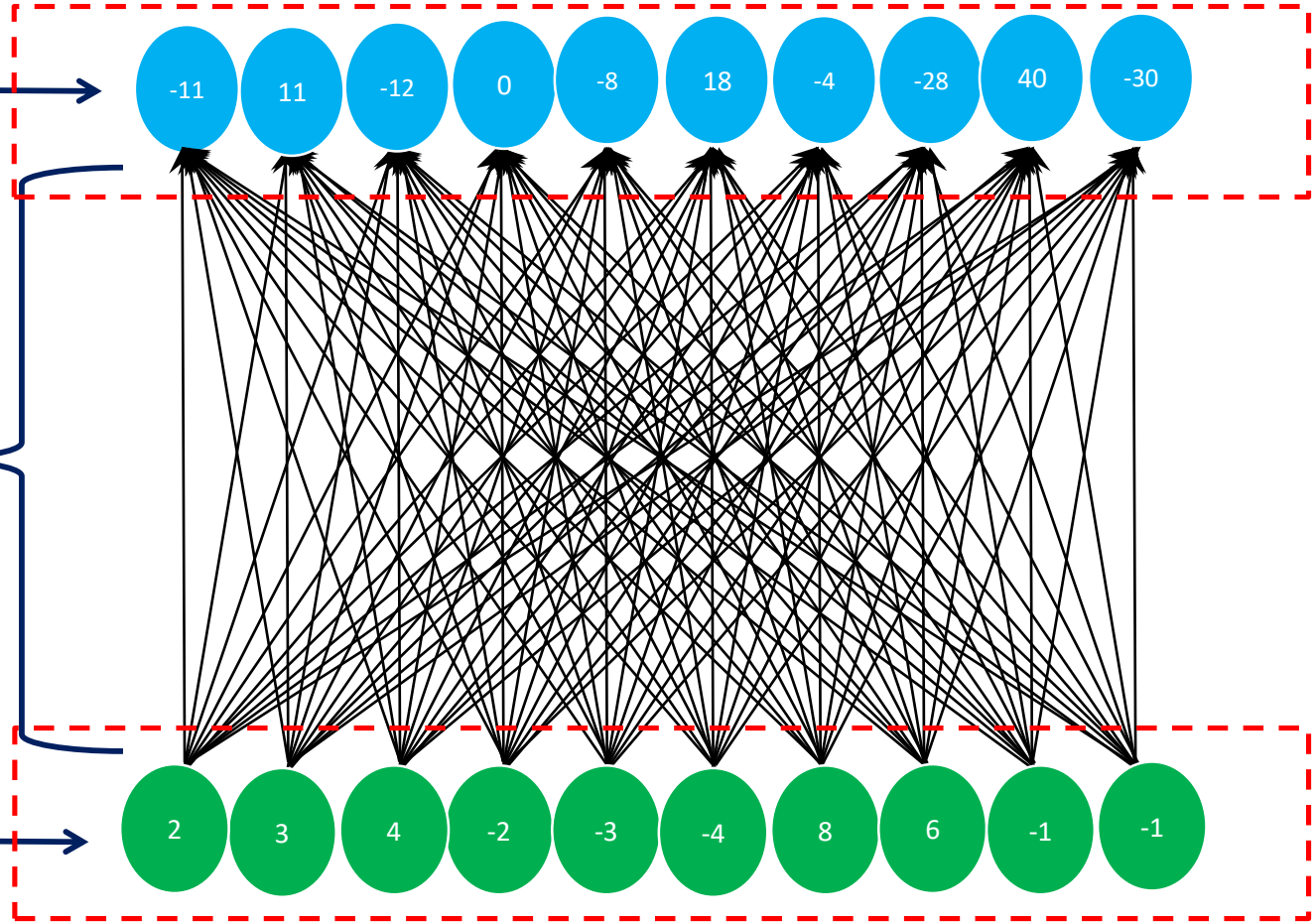
Convolutional Layer

Dense connections

$$Y = W^T X$$

$$W = \begin{bmatrix} -1 & 1 & -2 & 2 & -1 & 1 & 2 & -2 & 3 & -3 \\ -2 & 2 & -1 & 1 & -3 & 3 & -2 & 2 & -1 & 1 \\ -3 & 3 & -1 & 1 & -1 & 1 & -2 & -2 & 2 & 2 \\ -1 & 1 & -2 & 2 & -1 & 1 & 2 & -2 & 3 & -3 \\ -2 & 2 & -1 & 1 & -3 & 3 & -2 & 2 & -1 & 1 \\ -3 & 3 & -1 & 1 & -1 & 1 & -2 & -2 & 2 & 2 \\ -1 & 1 & -1 & 1 & -2 & 2 & -2 & -2 & 3 & -3 \\ -1 & 1 & -1 & -1 & 1 & 1 & +2 & -2 & 3 & -1 \\ -2 & 2 & -1 & 1 & -3 & 3 & -2 & 2 & -1 & 1 \\ -1 & 1 & -1 & -1 & 1 & 1 & +2 & -2 & 3 & -1 \end{bmatrix}$$

$$X$$

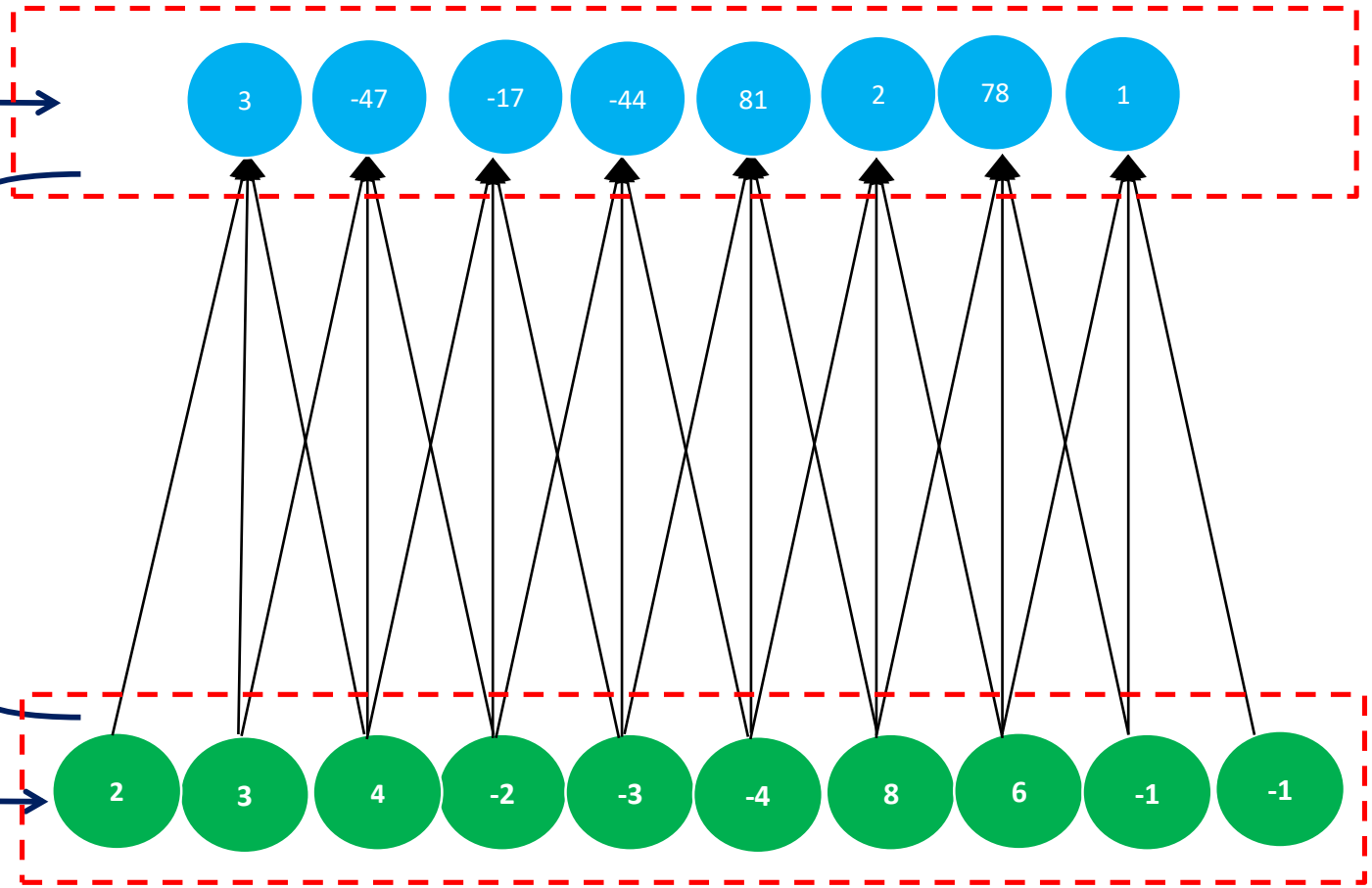


What if connections are only local?

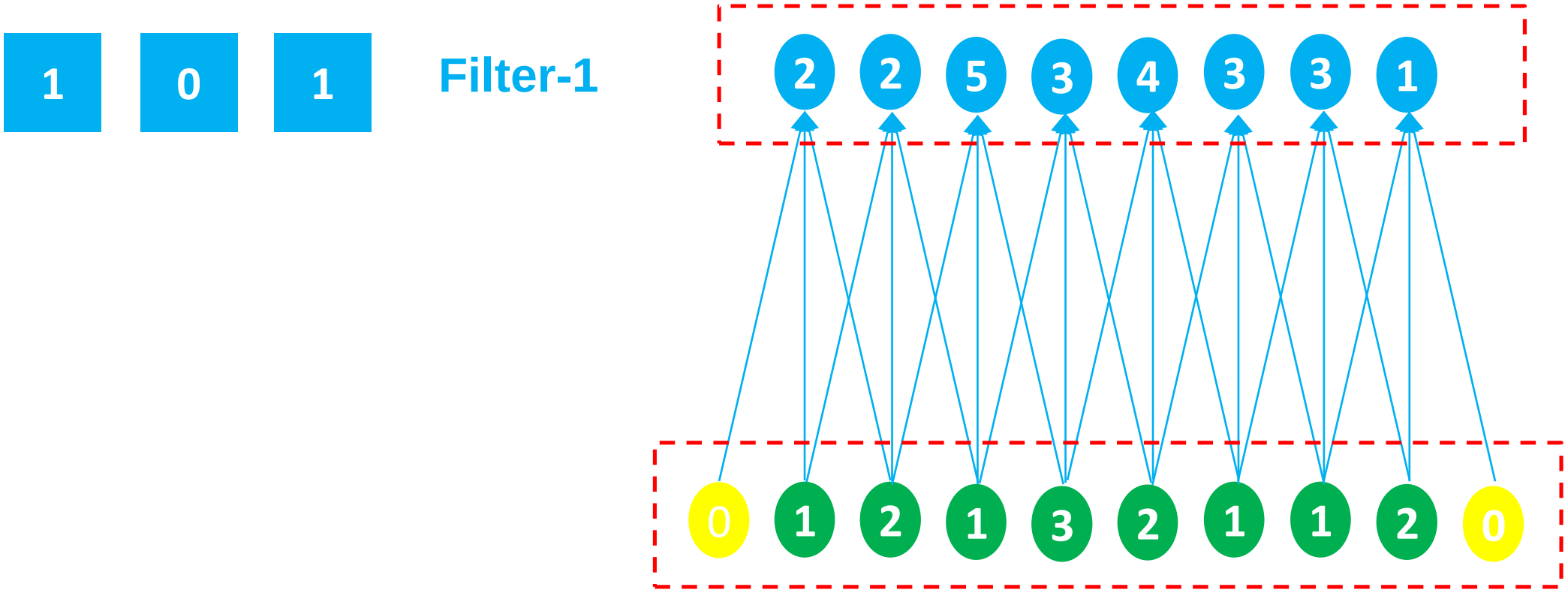
$$Y = W^T X$$

$$W = \begin{bmatrix} 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -7 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & -4 & -9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 4 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & -9 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 & -8 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 8 & -4 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 9 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & -6 & -7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -6 \end{bmatrix}$$

$$X$$



What if weights are same/shared?



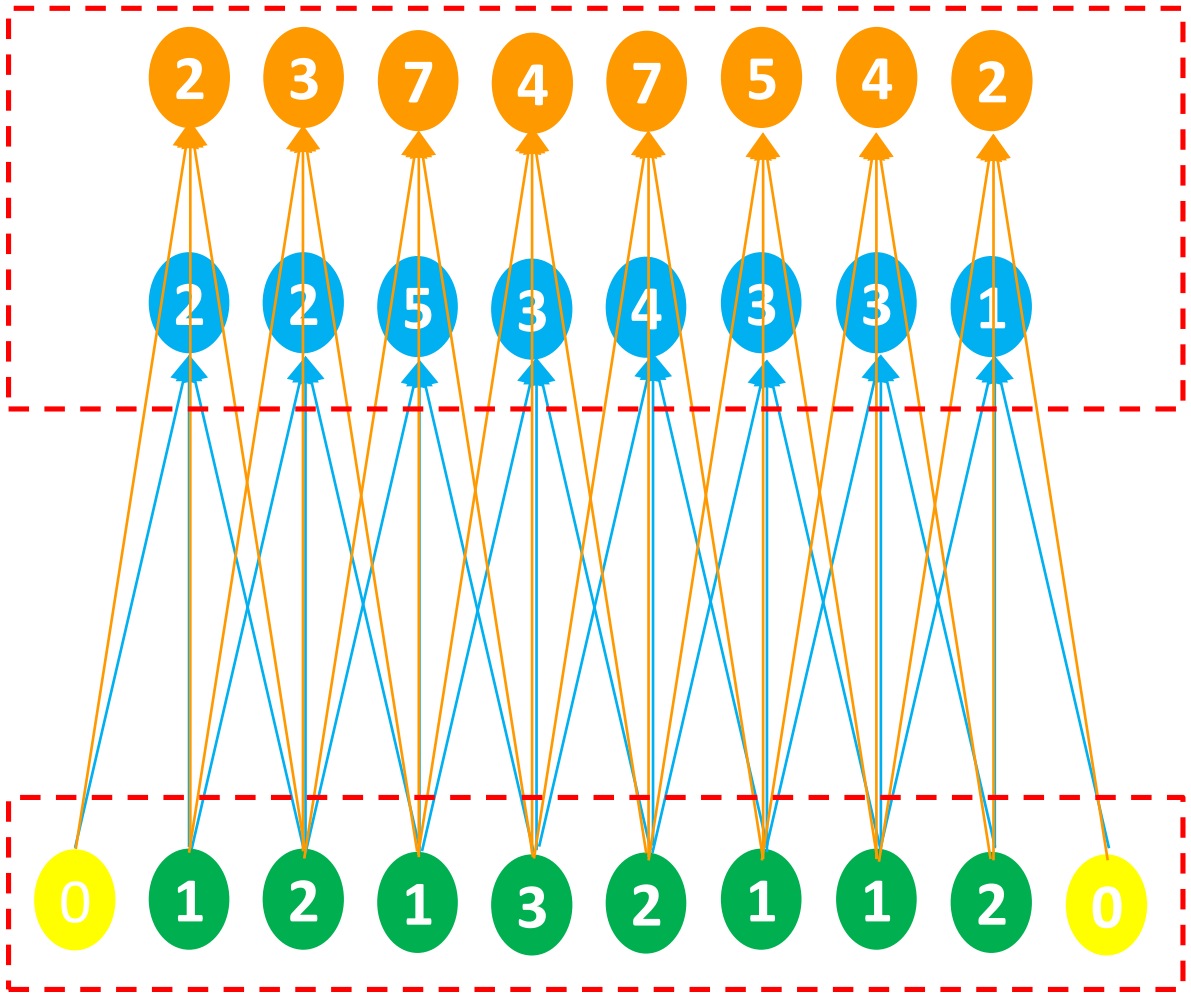
Two such filters/weights

2	0	1

Filter-2

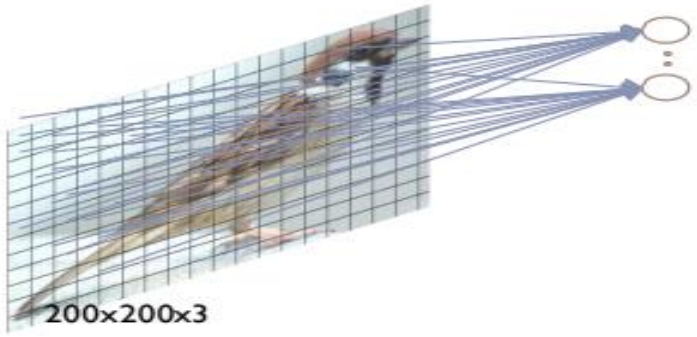
1	0	1

Filter-1



Convolution layer

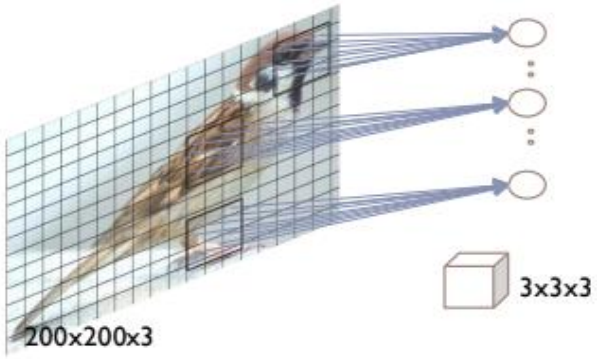
Fully connected layer



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

Parameter Calculations

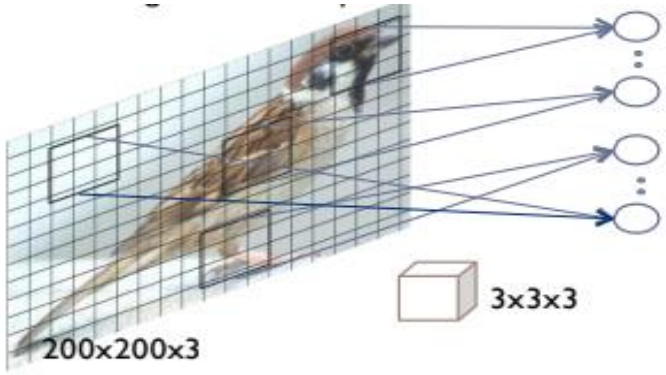
Locally connected layer



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

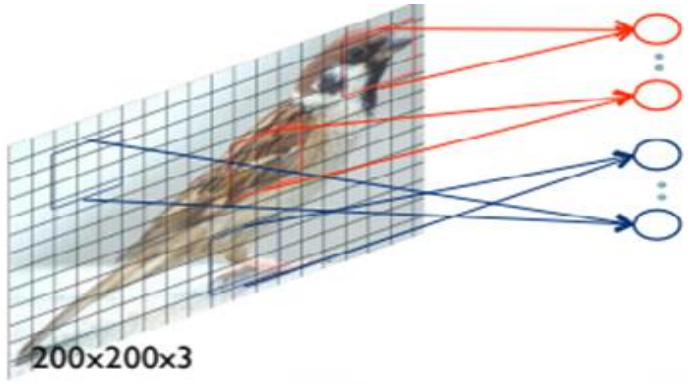
Convolution layer

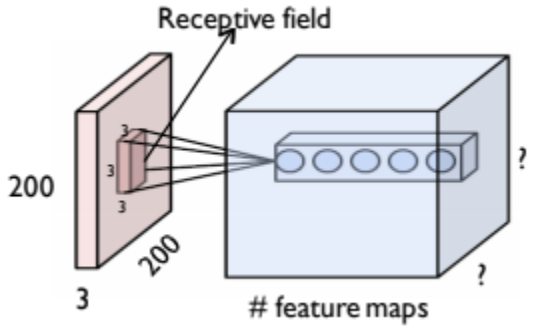
Convolutional layer with single feature map



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

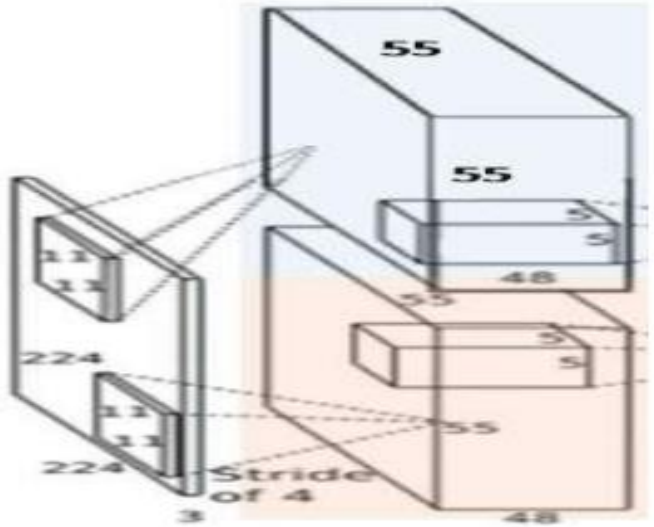
Convolutional layer with multiple feature maps





Parameter Calculation

- Filter Size: F
- Input volume streams: D
- # filters: K
- # parameters in a layer is **(F .F .D) .K**

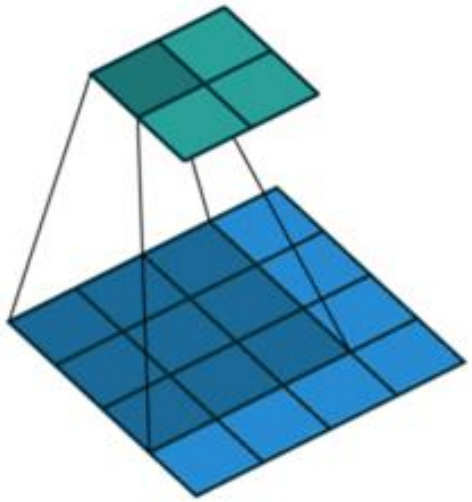


Example:

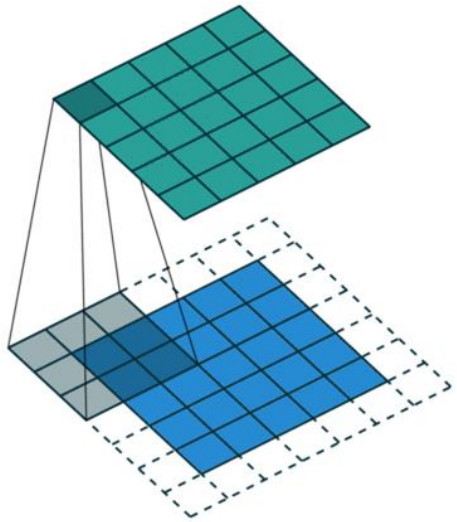
- For layer 1, Input images are 227 x 227 x 3
- F = 11 and K = 96
- Each filter has 11 x 11 x 3 = 363 and 1 (bias) i.e., 364 weights
- # weights = 364 x 96 = **35 K** (approx.)

CNNs

- Window size
- Stride
- Padding



Window size: 3x3
Stride: 1
Padding: 0

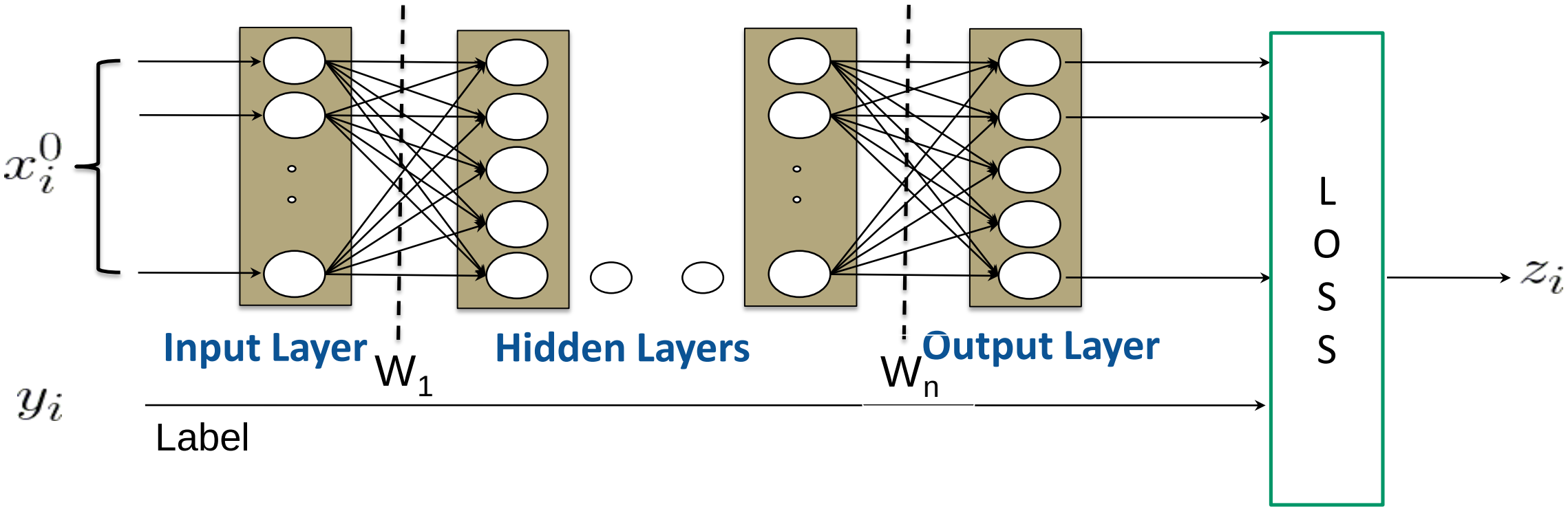


Window size: 3x3
Stride: 1
Padding: 1

Questions?

Backpropagation

Loss or Objective



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

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

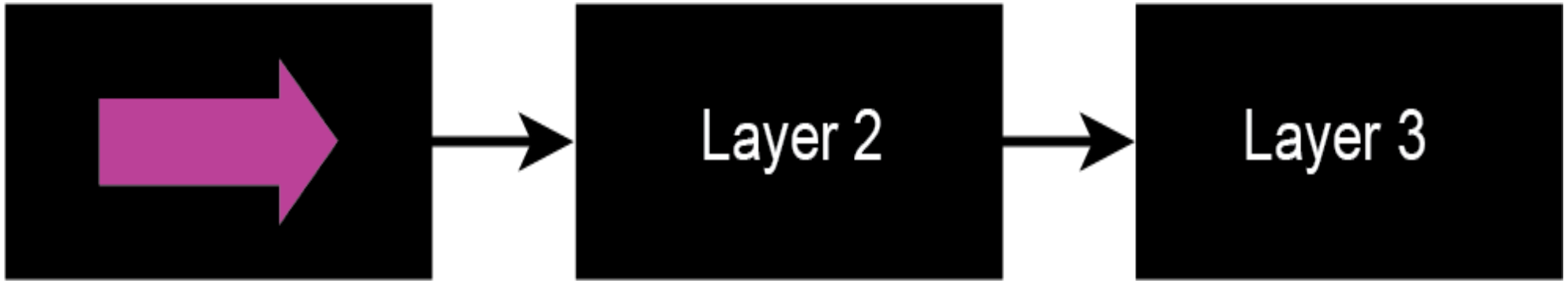
Weight vector

$$z_i = \frac{1}{2} \parallel x_i^n - y_i \parallel_2^2$$

E.g. Squared Loss

Neural Network Training

- Step 1: Compute loss on mini-batch [F-Pass]
- Step 2: Compute gradients w.r.t parameters[B-Pass]



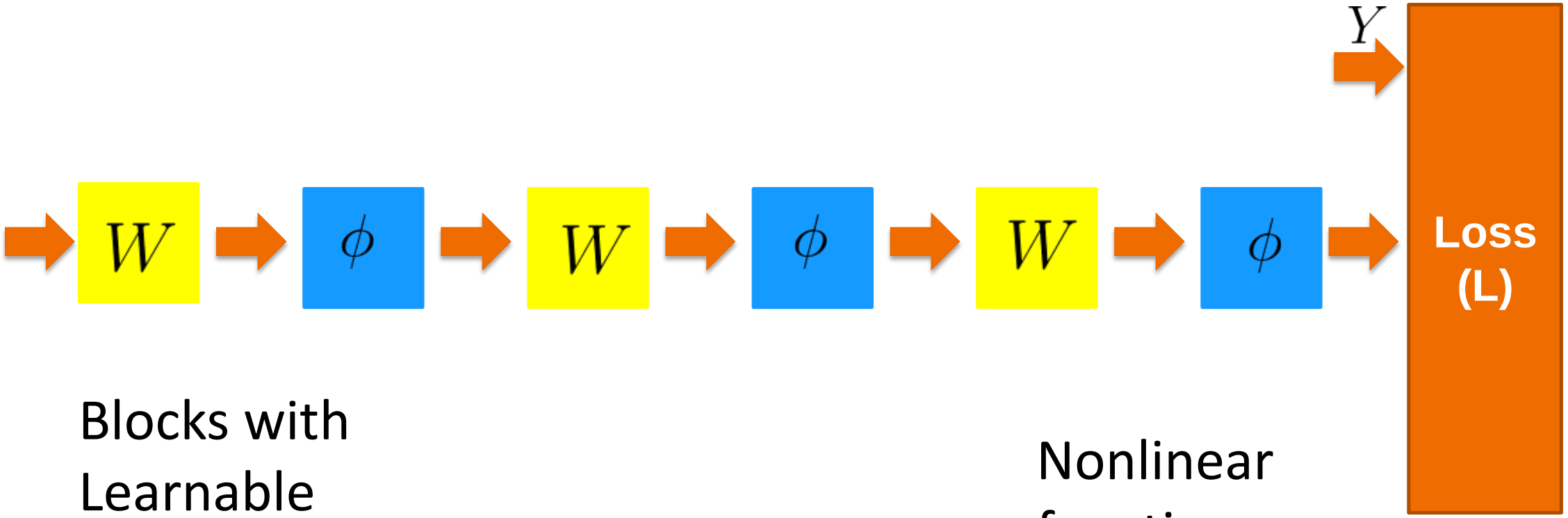
$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial \mathbf{L}}{\partial \mathbf{W}}$$

Chain Rule

- If a variable z depends on the variable y, which itself depends on the variable x, so that y and z are therefore dependent variables, then z, via the intermediate variable of y, depends on x as well. The chain rule then states,

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

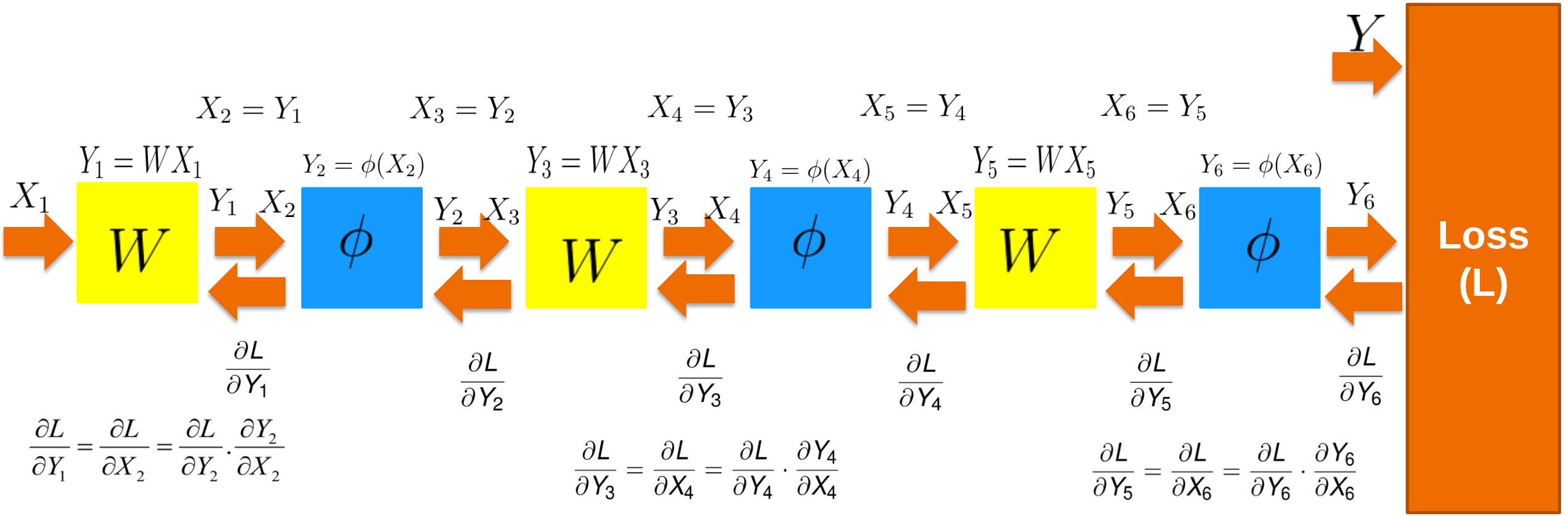
A simpler view point



Blocks with Learnable parameters
Matrix Multiplication

Nonlinear functions
(often non learnable)

Back Propagation (X,Y): Also Learning



$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial Y_1} \cdot \frac{\partial Y_1}{\partial W}$$

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial Y_3} \cdot \frac{\partial Y_3}{\partial W}$$

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial Y_5} \cdot \frac{\partial Y_5}{\partial W}$$

$$W^{n+1} = W^n - \eta \frac{dL}{dW}$$

Back Propagation

$$(1) \rightarrow \frac{\partial L}{\partial x} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial x} \quad (2) \rightarrow \frac{\partial L}{\partial W} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial W} \quad (3) \rightarrow W^{n+1} = W^{n}\,\eta\,\frac{\partial L}{\partial W}$$

- Let there be N stages. For a computational block ℓ,
  - Compute $\frac{\partial L}{\partial x}$ using equation 1
  - If the block as a learnable parameters **W** then,
    - Compute $\frac{\partial L}{\partial W}$ using equation 2
    - Update the parameters using equation 3
  - Set the $\frac{\partial L}{\partial x}$ of stage ℓ as $\frac{\partial L}{\partial y}$ of stage ℓ - 1, and repeat the steps 1-3, until we reach the first block.

Summary

- Step 0:
  - Initialize the Network (MLP), weights

- Step 1:
  - Do forward pass for a batch of randomly selected samples.
  - Predict outputs with the existing weights.

- Step 2:
  - Compute Loss for the set of samples.

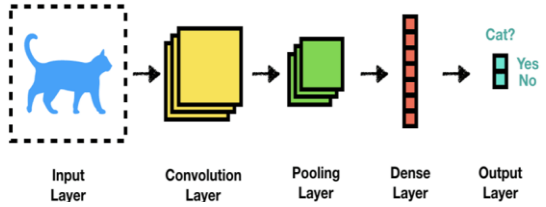
Summary

- Step 3:
  - Update all the weights using gradient descent.

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial \mathbf{L}}{\partial \mathbf{W}}$$

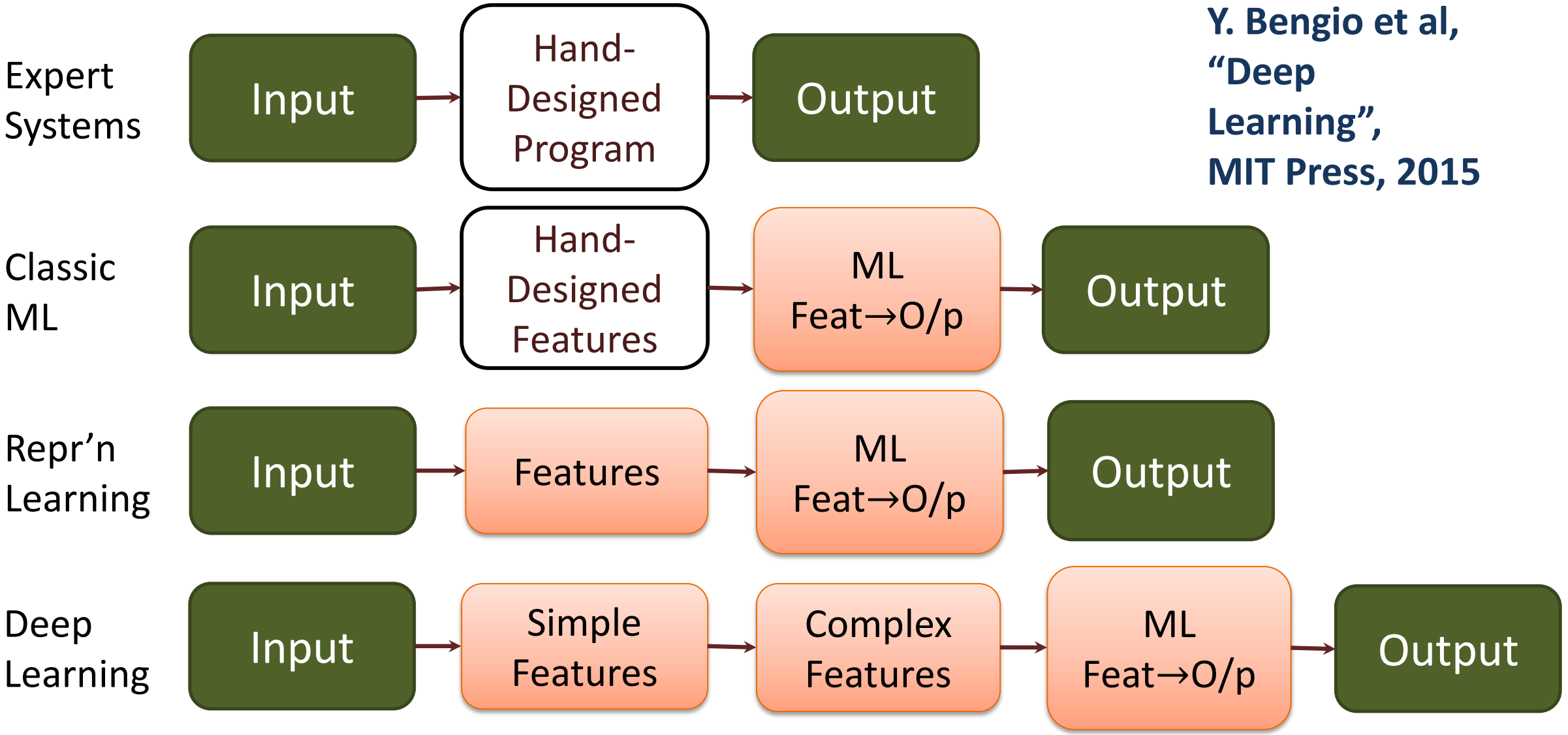
- Step 4:
  - Repeat Steps 1, 2 and with updated W till the Loss is less than a threshold ϵ

Questions?


Cat?
Yes
No
Input Layer
Convolution Layer
Pooling Layer
Dense Layer
Output Layer

CNNs

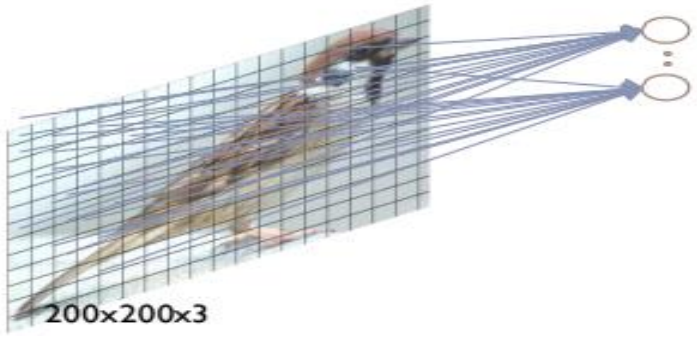
Evolution of Learning



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

Revisit: Convolution layer

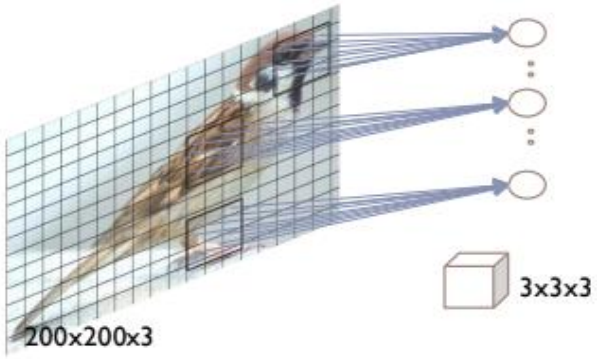
Fully connected layer



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

Parameter Calculations

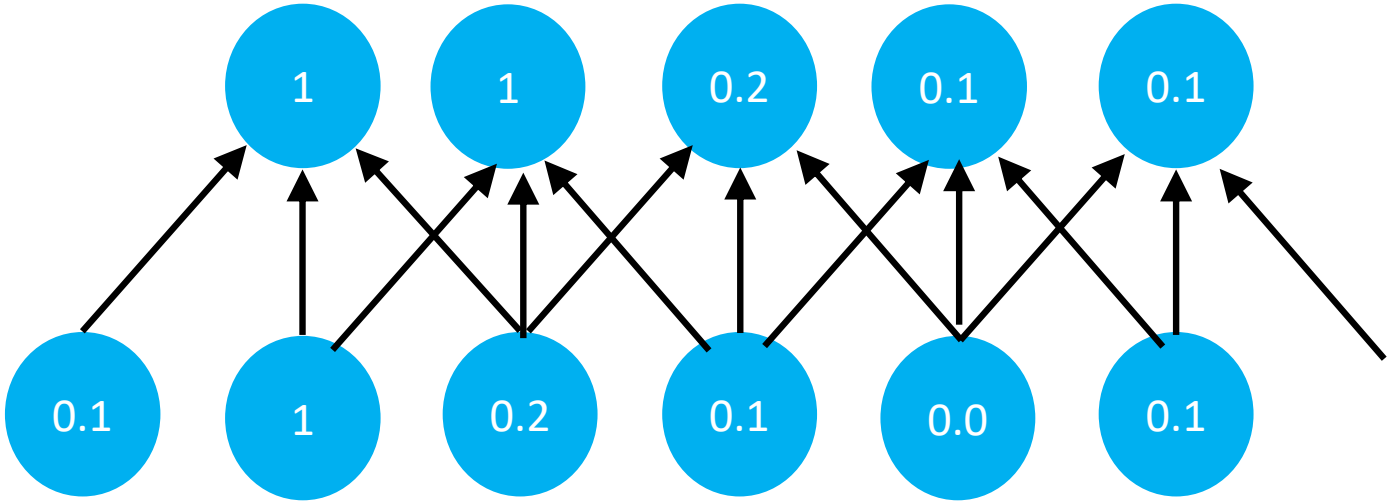
Locally connected layer



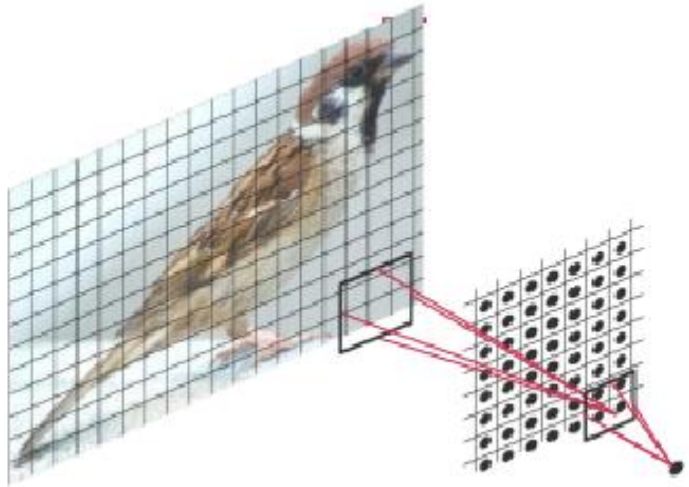
- #Hidden Units: 120,000
- #Params: 3.2 Million (= 120K X 27)
- Useful when the image is highly registered

Max Pooling

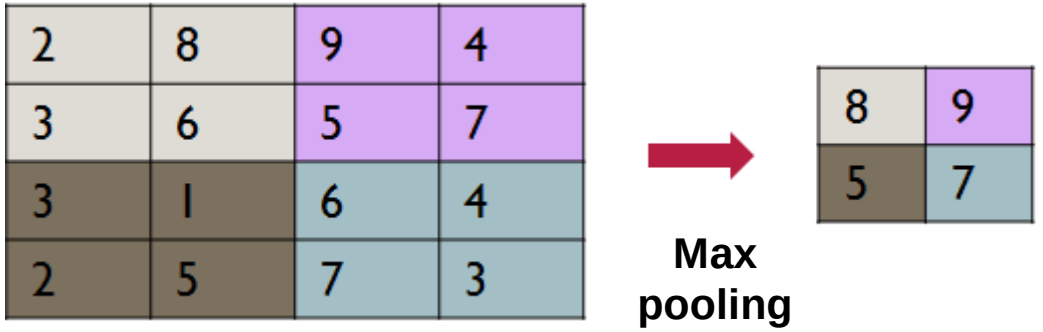
- Window size : 3



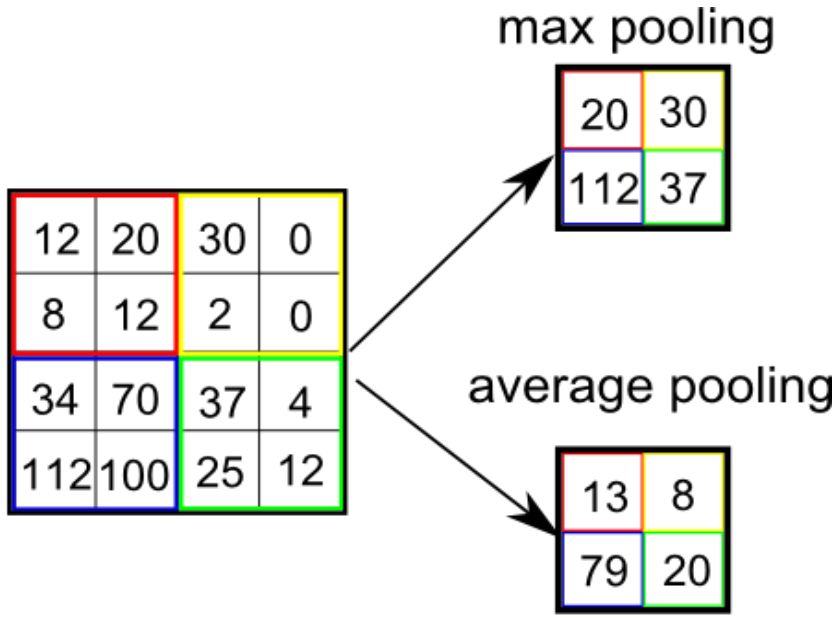
Pooling Layer



Pool Size: 2x2
Stride: 2
Type: Max



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

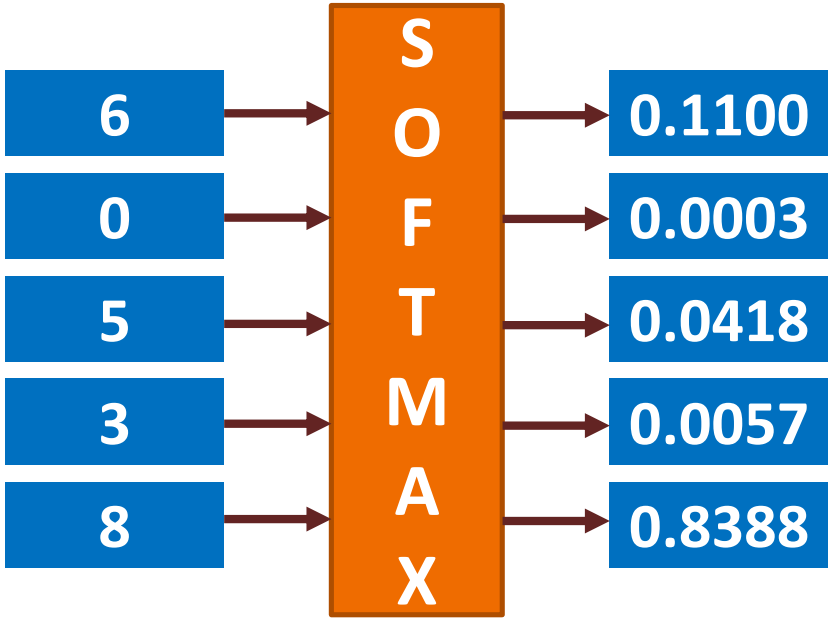


Softmax

```
Out[12]: array([ 6.,  0.,  5.,  3.,  8.])

In [8]:  exp = (np.e)**(x)
         exp
         executed in 6ms, finished 01:47:23 2018-08-21

Out[8]:  array([  4.03428793e+02,   1.00000000e+00,   1.48413159e+02,
                  2.00855369e+01,   2.98095799e+03])

In [9]:  sigma_e = np.sum(exp)
         sigma_e
         executed in 9ms, finished 01:47:25 2018-08-21

Out[9]:  3553.8854765602264

In [11]: z = exp/sigma_e
         z
         executed in 8ms, finished 01:47:34 2018-08-21

Out[11]: array([  1.13517669e-01,   2.81382168e-04,   4.17608165e-02,
                  5.65171192e-03,   8.38788421e-01])
```

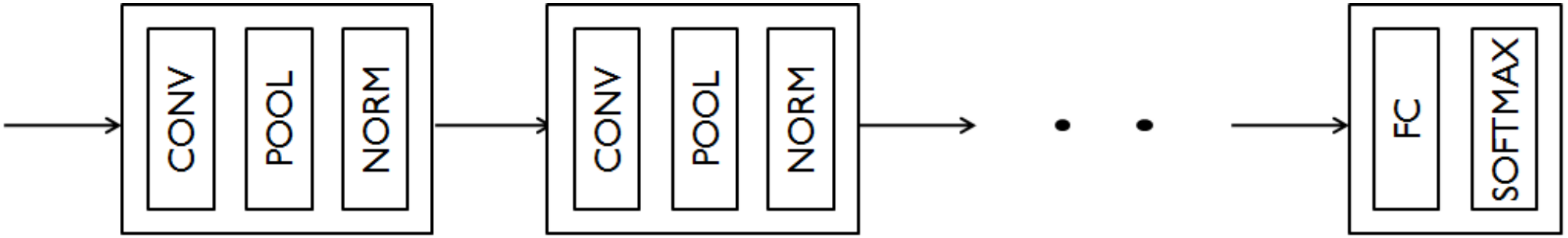
- Normalizes the output.
- K is total number of classes

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



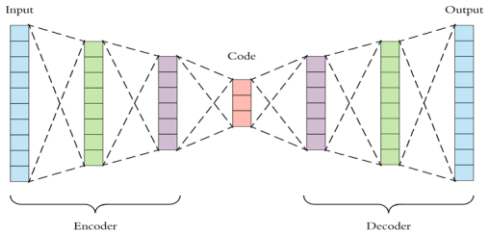
Typical Architecture

- A typical deep convolutional network



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

Questions?

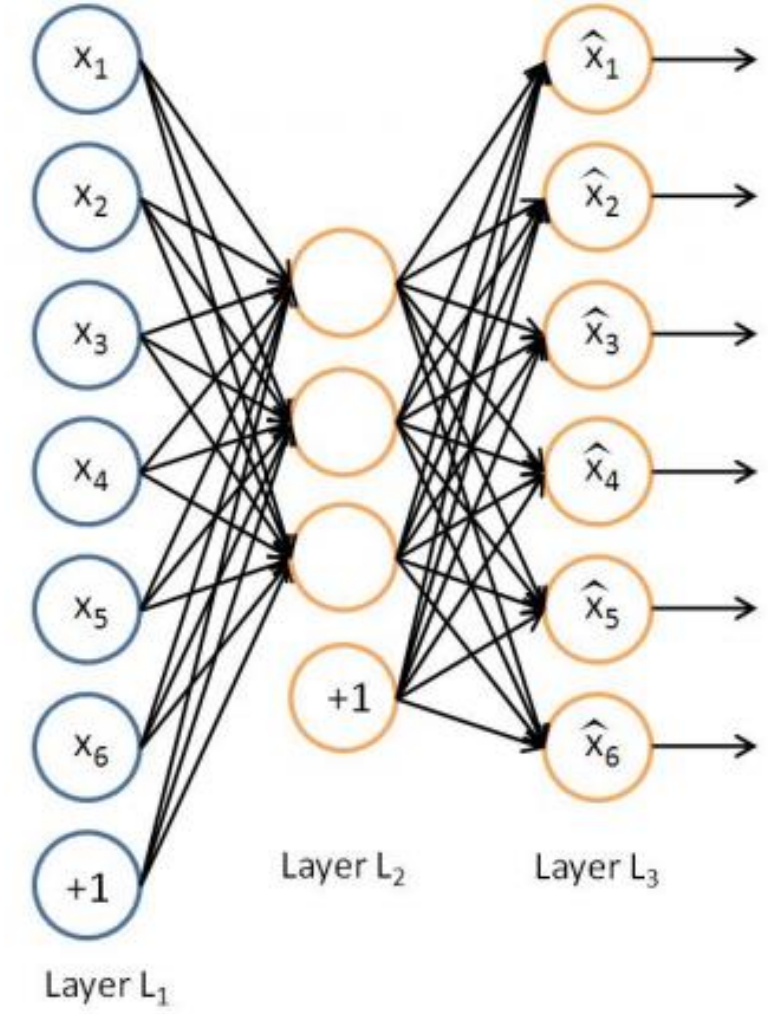


Auto Encoder

What if we do not have labels?

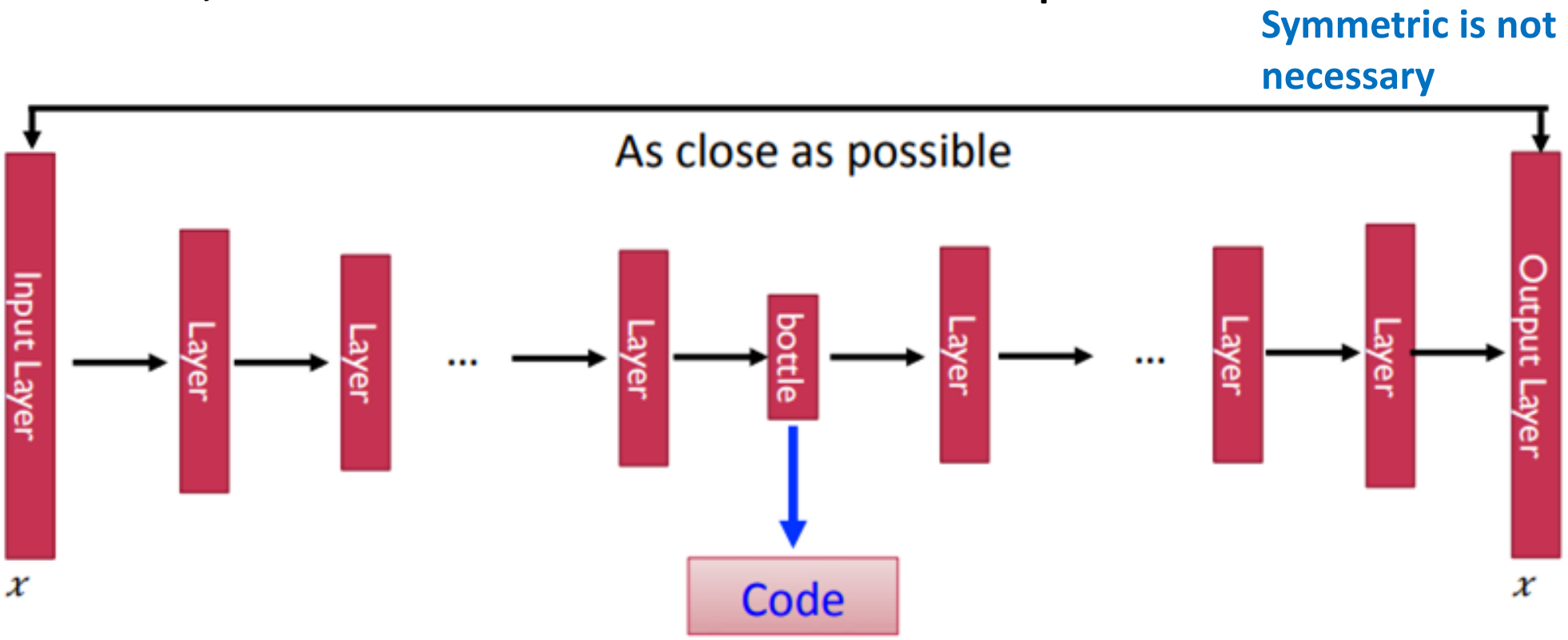
Auto-encoder

- Similar to MLP
- Input is the same as the output
- Network learns to reconstruct.
- “Bottleneck” layer learns a compact representation.



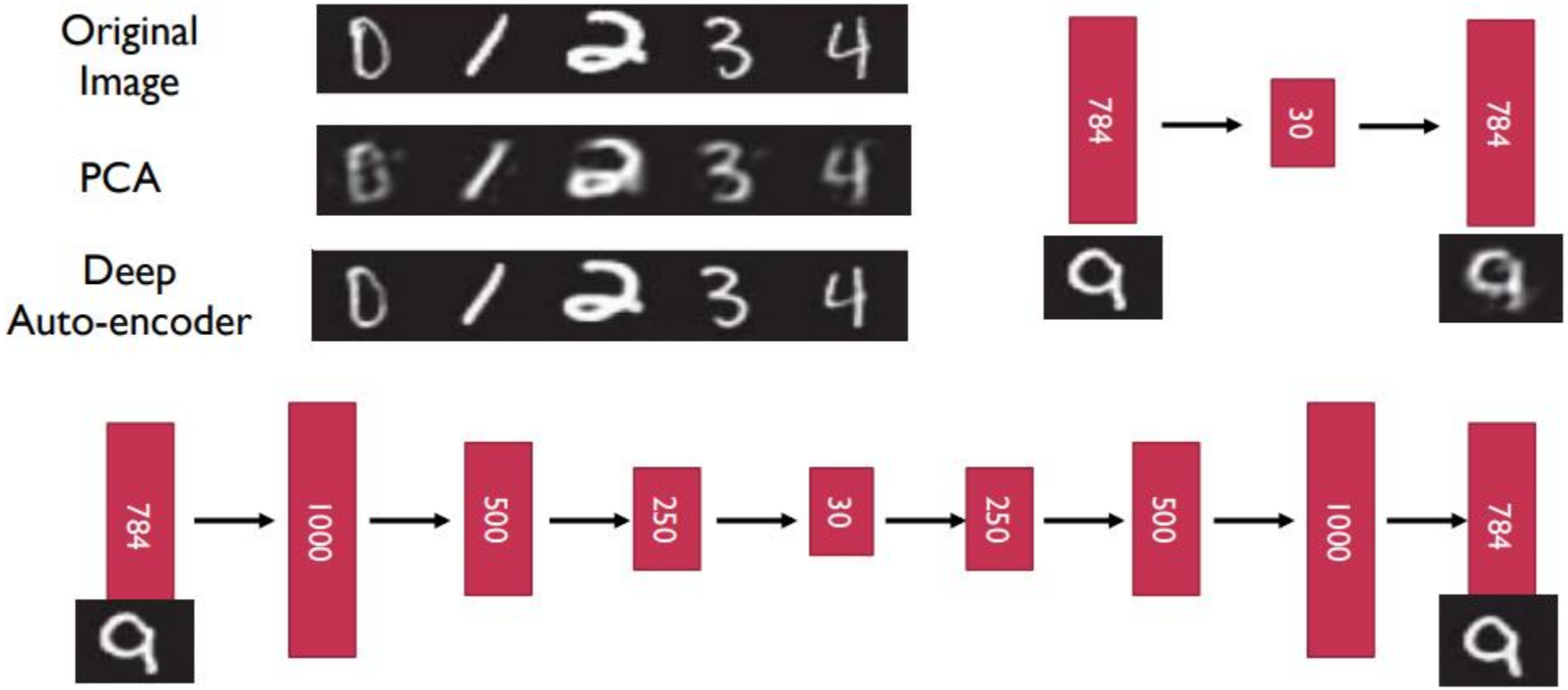
Deep Auto-encoder

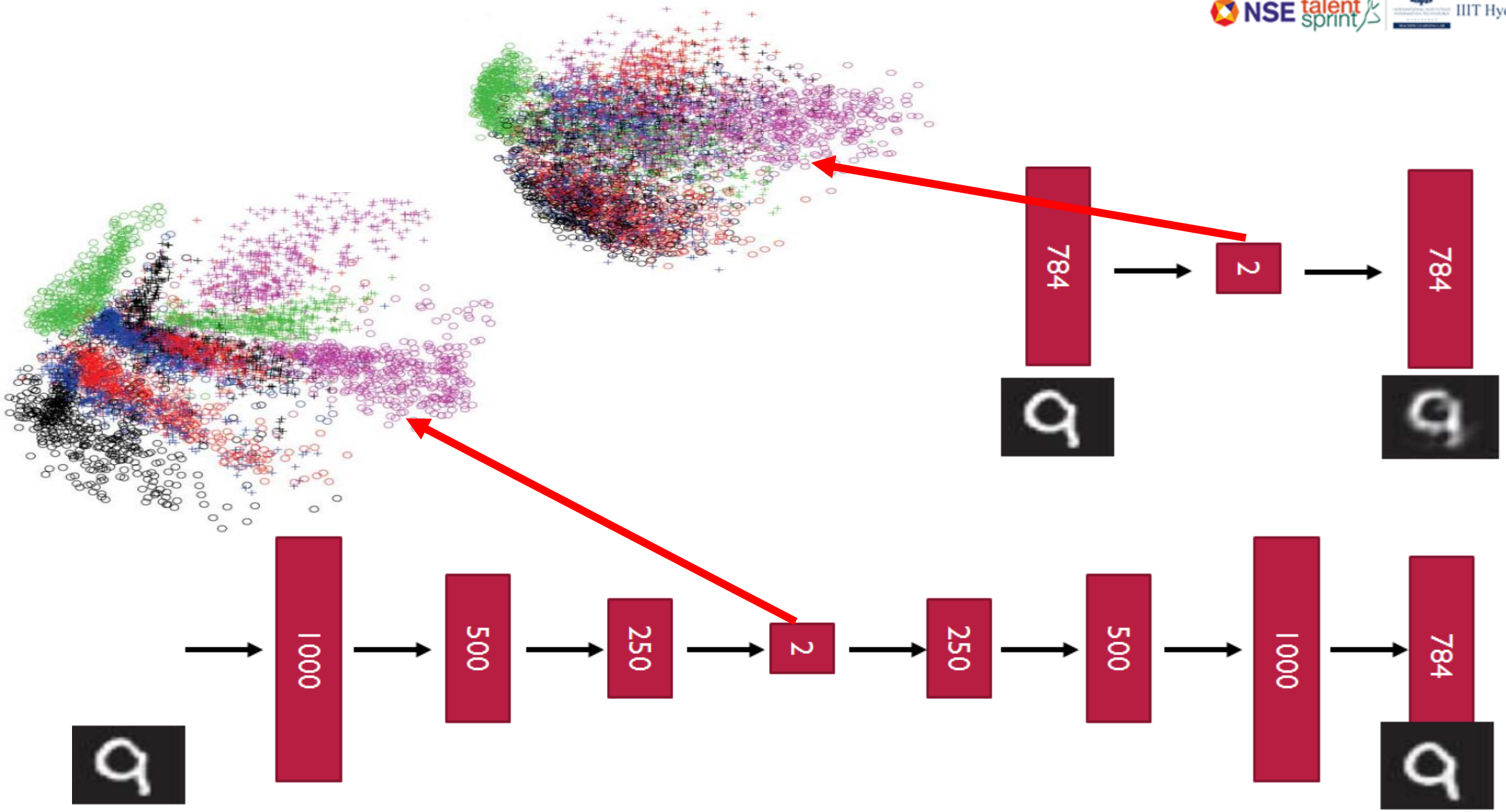
- Of course, the auto-encoder can be deep



Reference: Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507

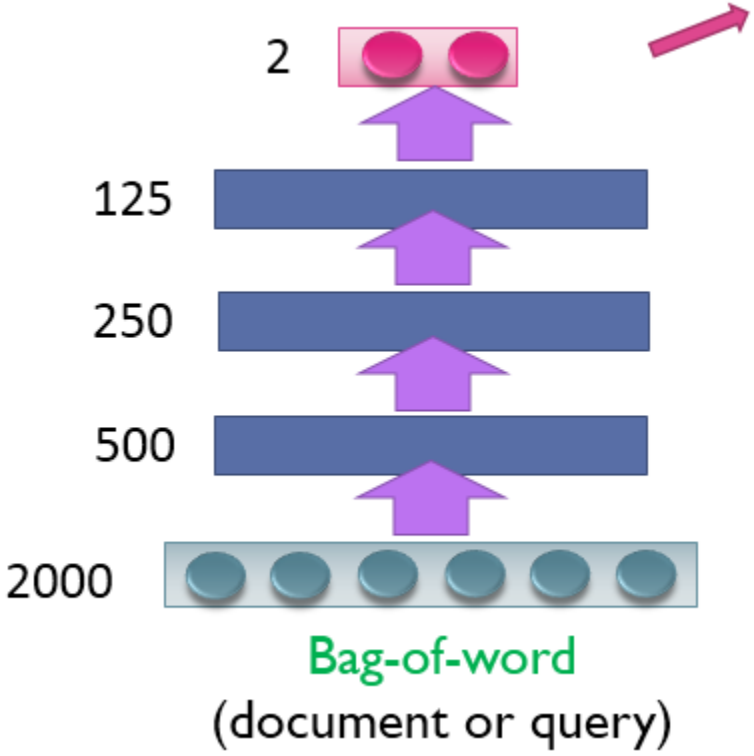
Deep Auto-encoder

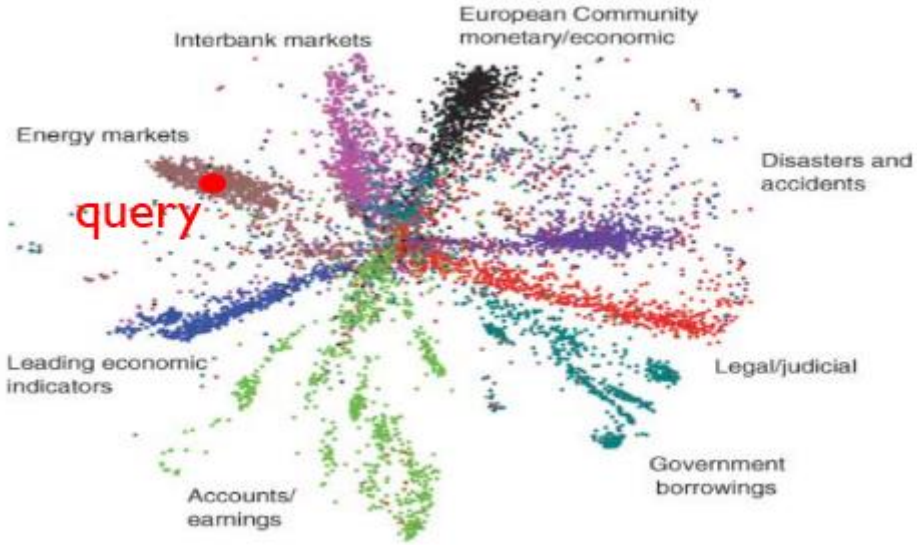



784
2
784
1000
500
250
2
250
500
1000
784

Auto-encoder – Text Retrieval

- The documents talking about the same thing will have close code.

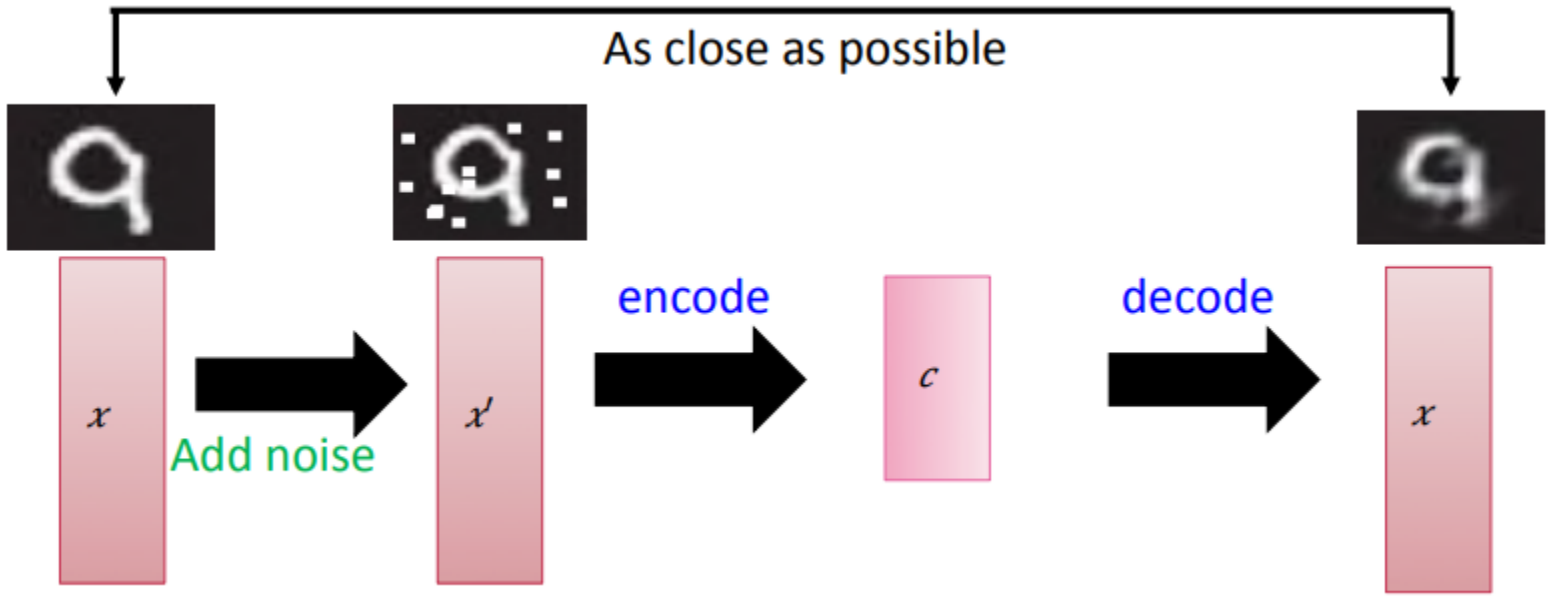




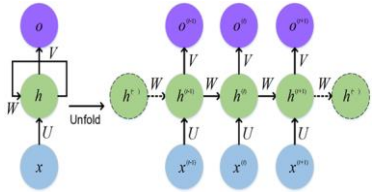
LSA: project documents to 2 latent topics

Auto-encoder

- De-noising auto-encoder



Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." *ICML,* 2008.

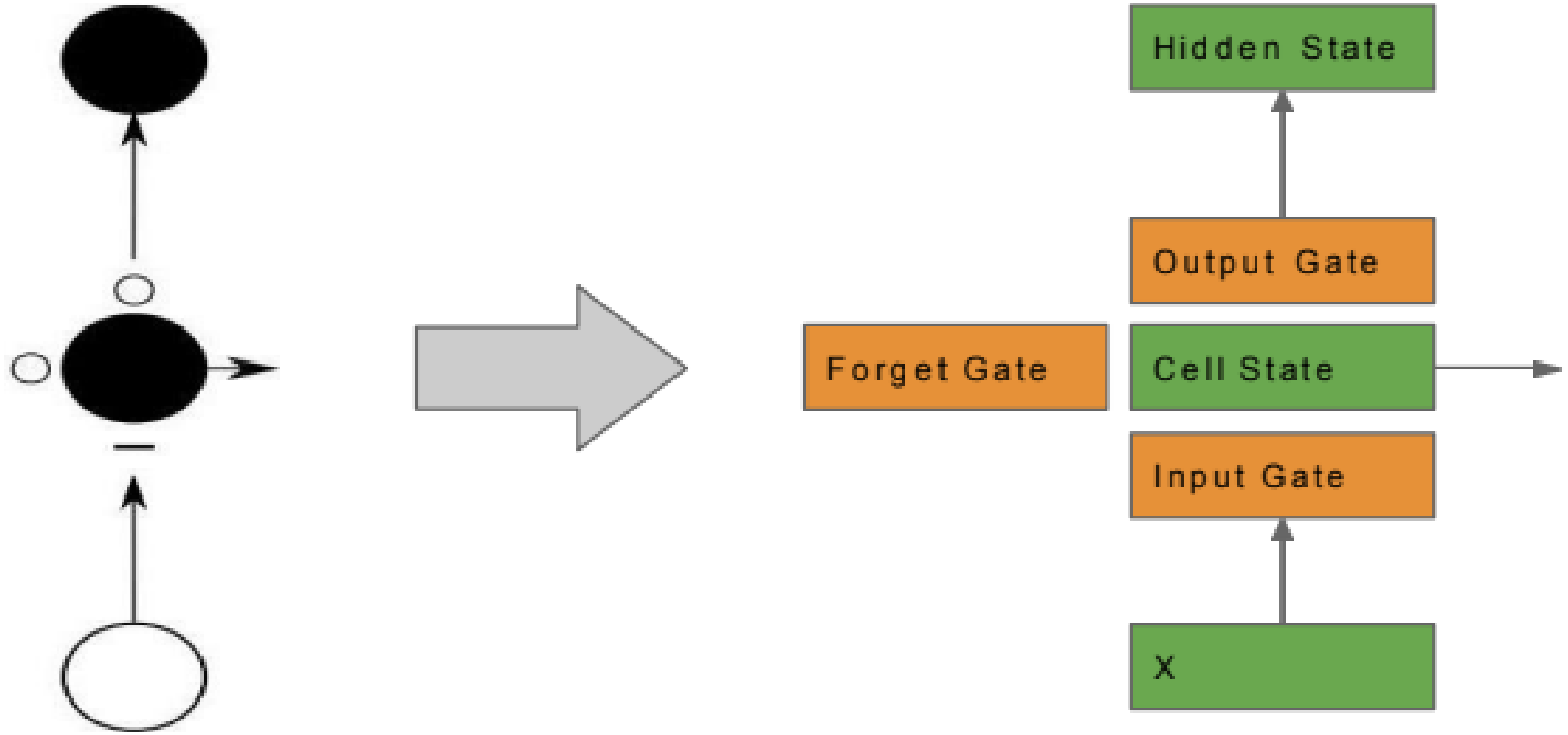


Introduction to RNNs

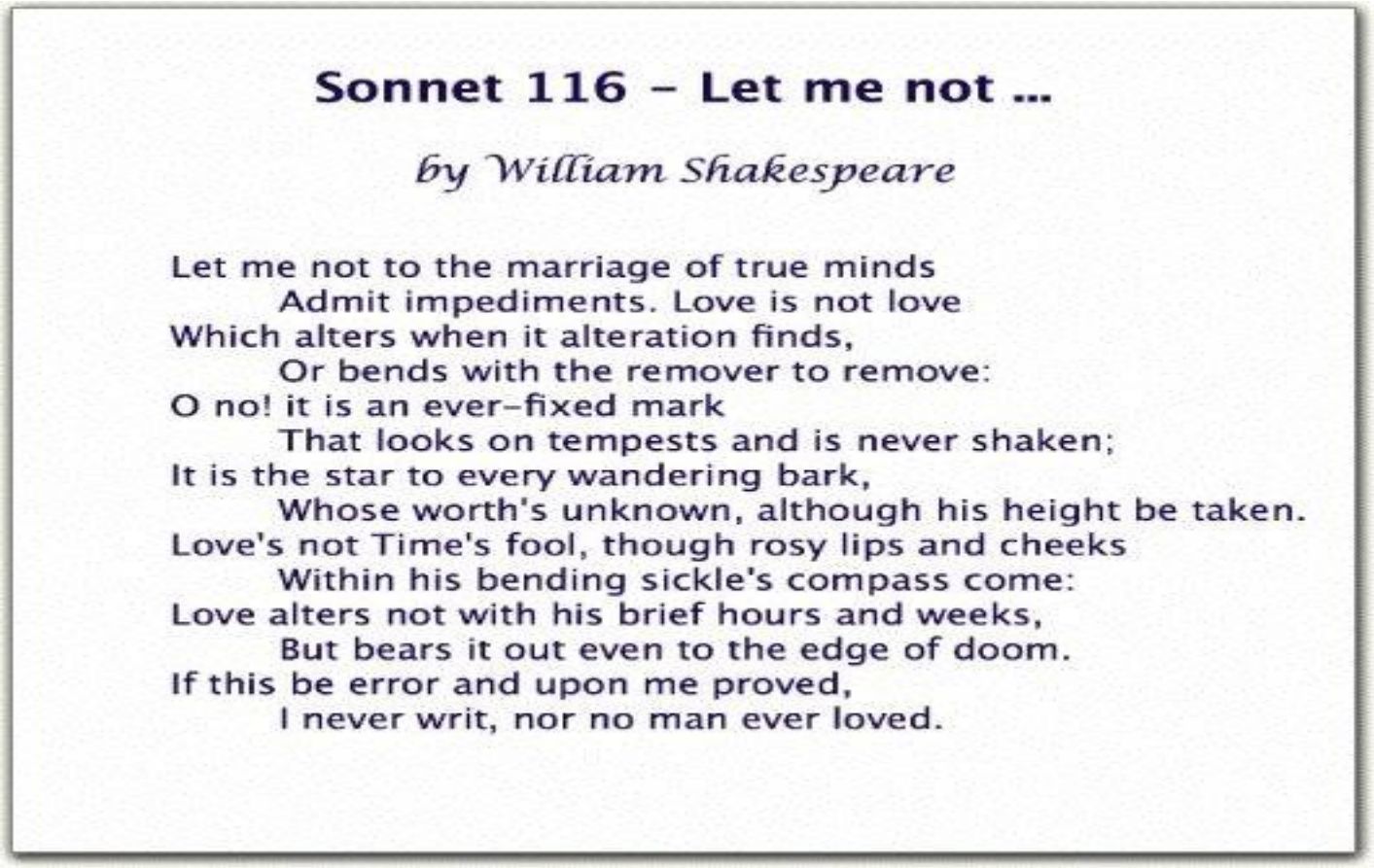
Why RNNs ?

- 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 tomato price of tomorrow?
- Popular networks (Eg. CNNs) do not have cycles.

Cell with Memory



Generating poetry with RNNs



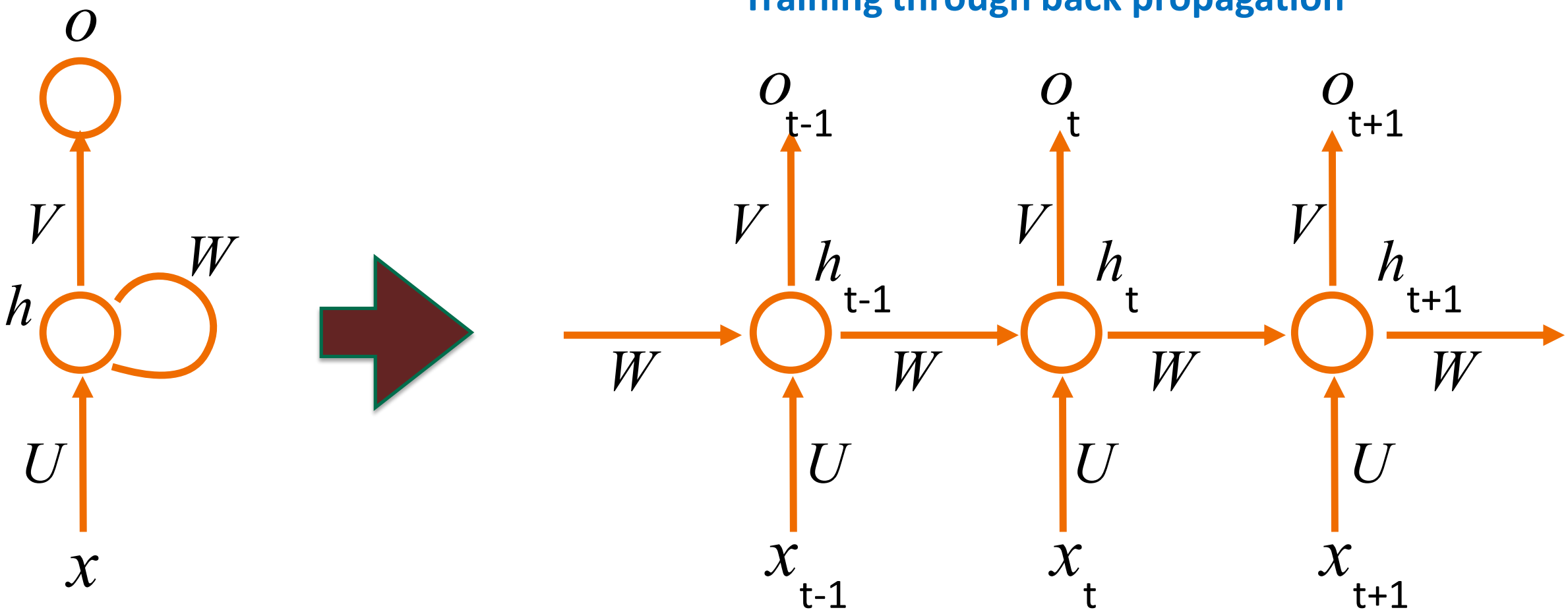
Sonnet 116 – Let me not …

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.

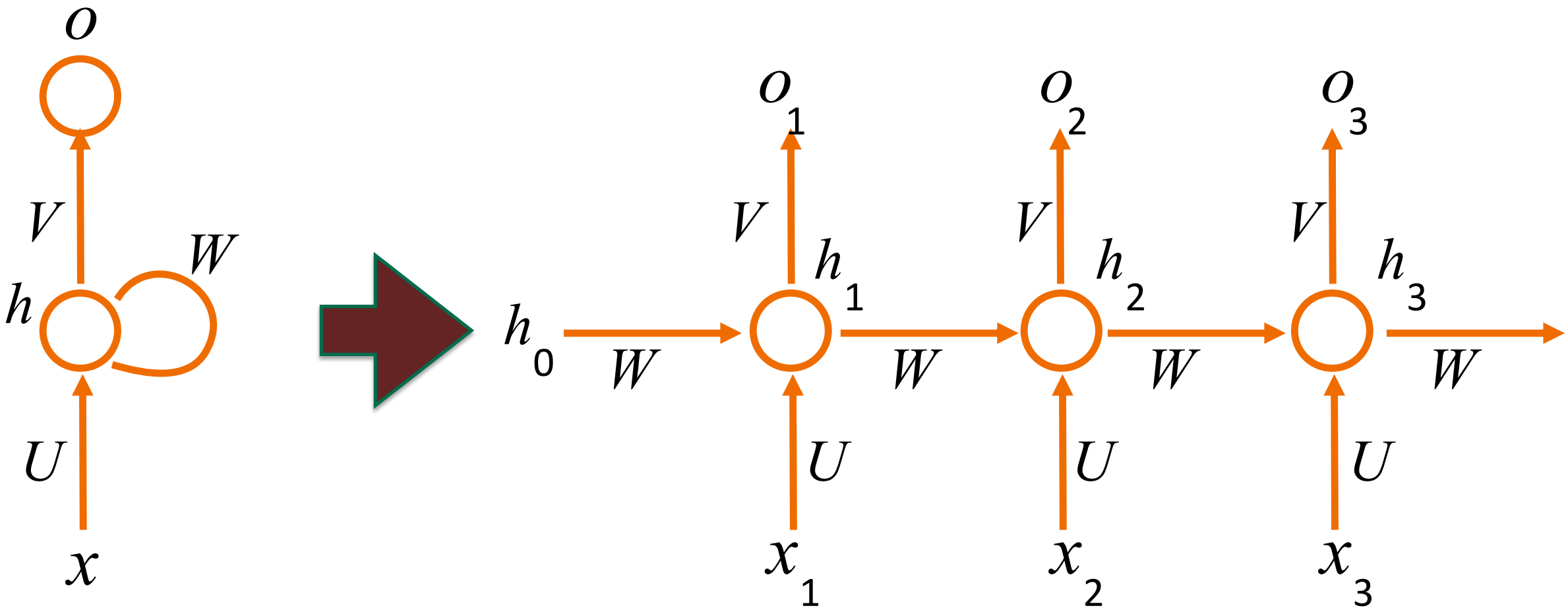
Andrej Karpathy

RNN basic architecture



$$h_t = f(Ux_t + Wh_{t-1}) \qquad o_t = \text{softmax}(Vh_t)$$

RNN basic architecture



$$h_t = f(Ux_t + Wh_{t-1}) \qquad o_t = \text{softmax}(Vh_t)$$

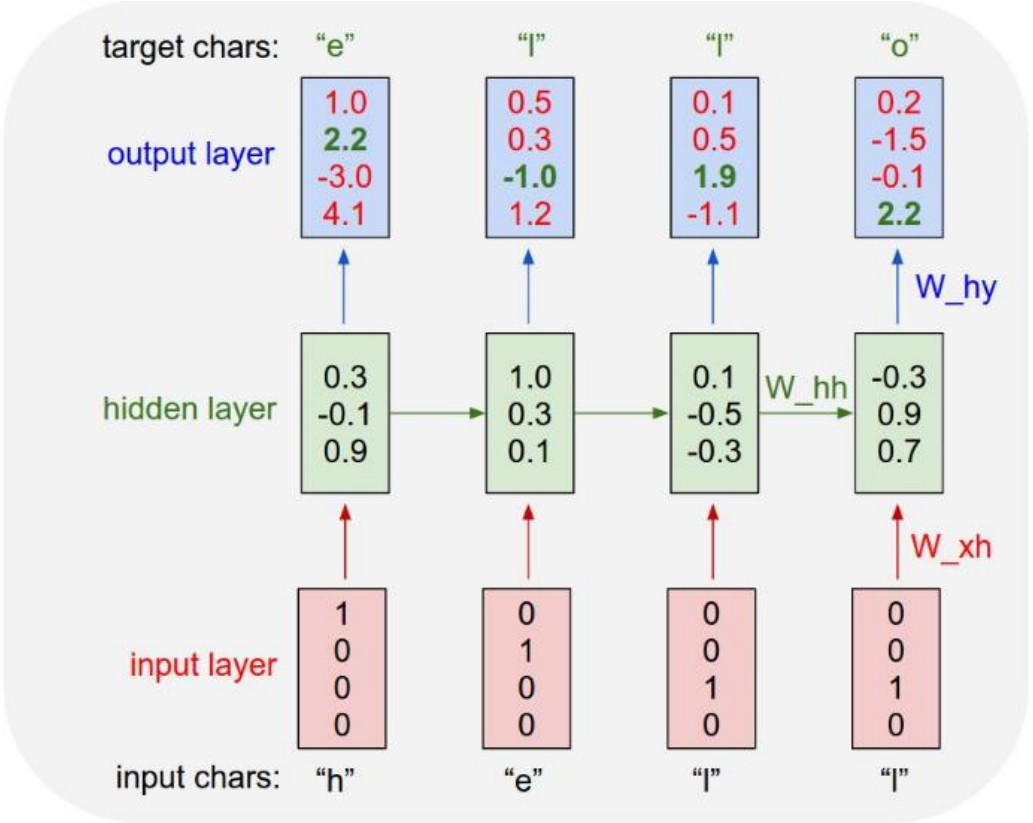
RNN basic architecture

- x_t- input at time step t
- h_t - hidden state at time step t (memory of the network)
- o_t - output at time step t
- U, V, W are parameters (shared across all layers)

Character Level Language Modelling

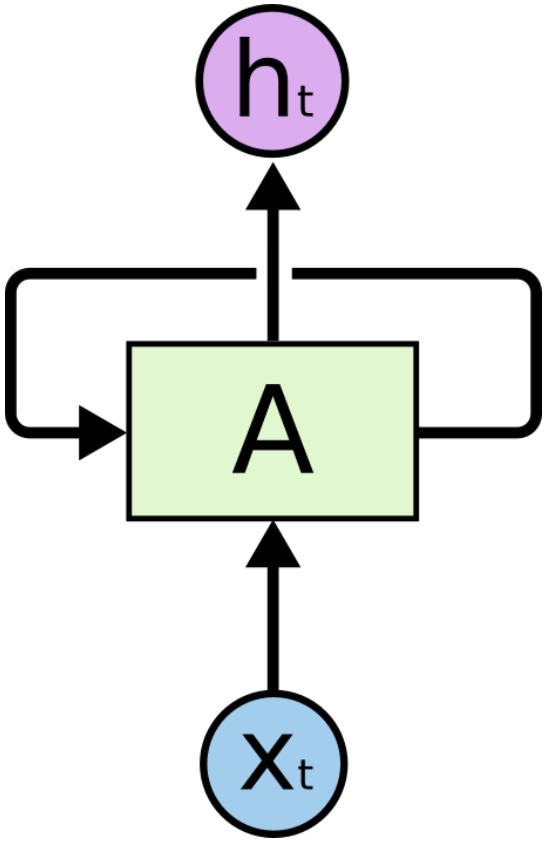
Task:

- Predicting the next character given the current character



It may be 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. □

at first:

```
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

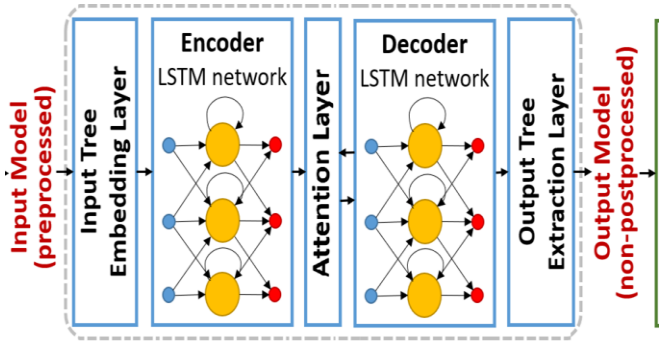
```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

```
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

```
"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.
```



Why RNN's?

Remember the History

Machine learning with sequential data

- Turn an input sequence into an output sequence that lives in a different domain or similar domain
  - e.g. sequence of sound pressures into a sequence of words
  - e.g. sequence of images into sequence of words
  - e.g. language translation

Machine learning with sequential data

- Learn a model to predict the next term in the input sequence
  - e.g. predict next word given a set of words (we use this on everyday basis)
  - e.g. stock market time series data

- Predicting the next term in a sequence is supervised or unsupervised learning?

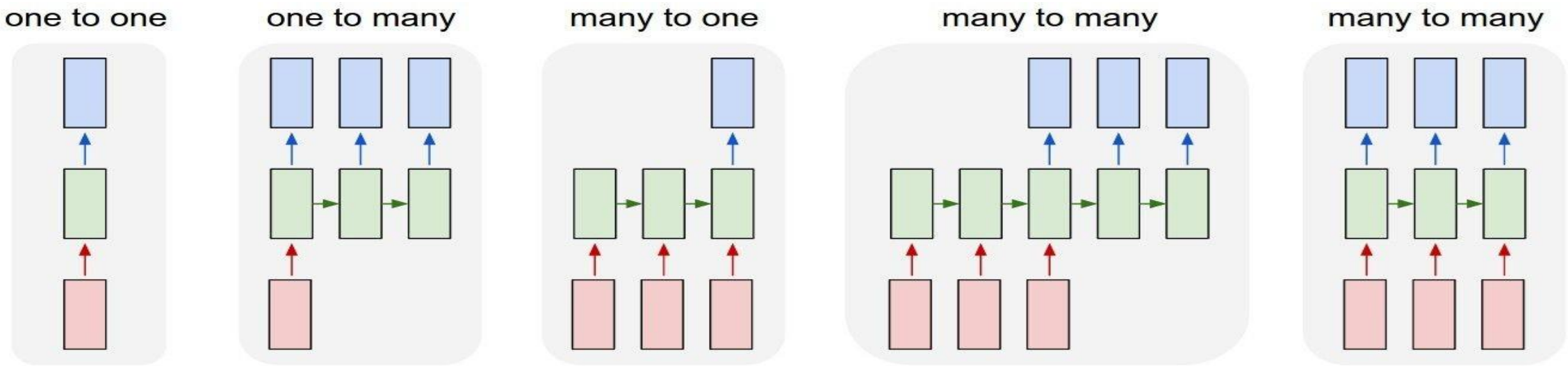
- Uses methods designed for supervised learning but doesn't require a separate teaching signal

RNN has been a natural choice

- In many NLP tasks
- In time series data (stock prices, web logs, weather etc.)
- Fill in the blanks or predict what happens next in time
- Sequence to sequence (translation, speech recognition etc.)
- Sometimes for classifying a sequence (sentiment analysis)

Applications and Use cases

Recurrent Networks offer a lot of flexibility:



vanilla neural networks

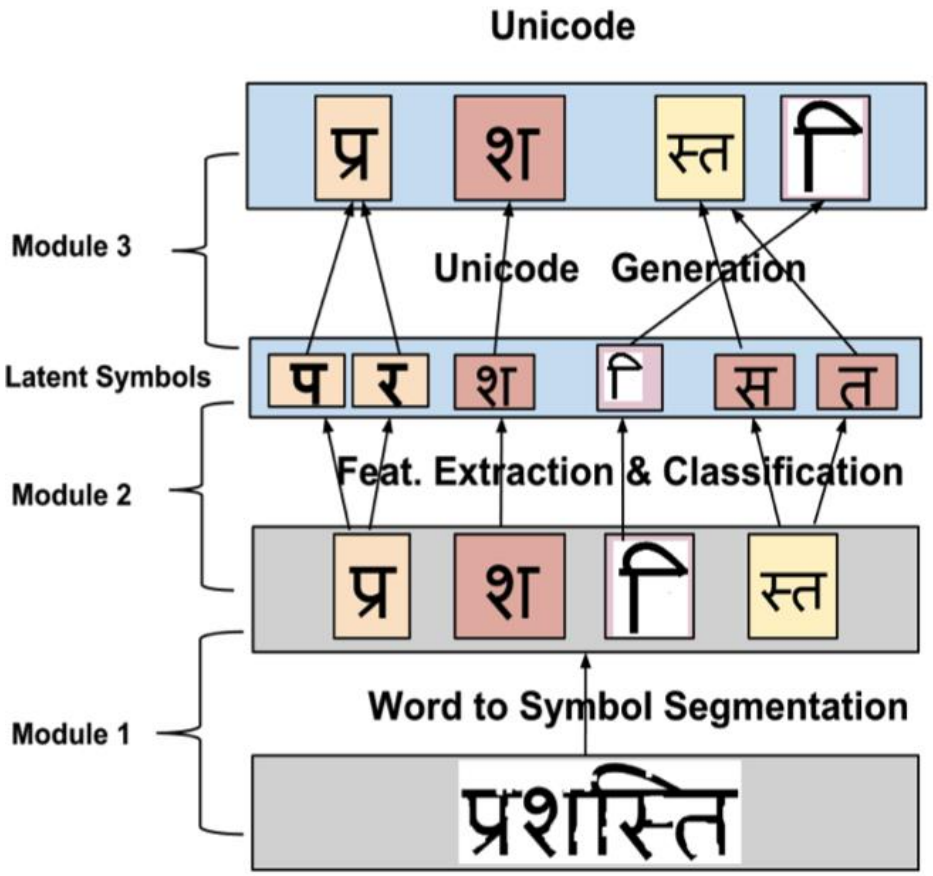
e.g. **sentiment classification**
sequence of words -> sentiment

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

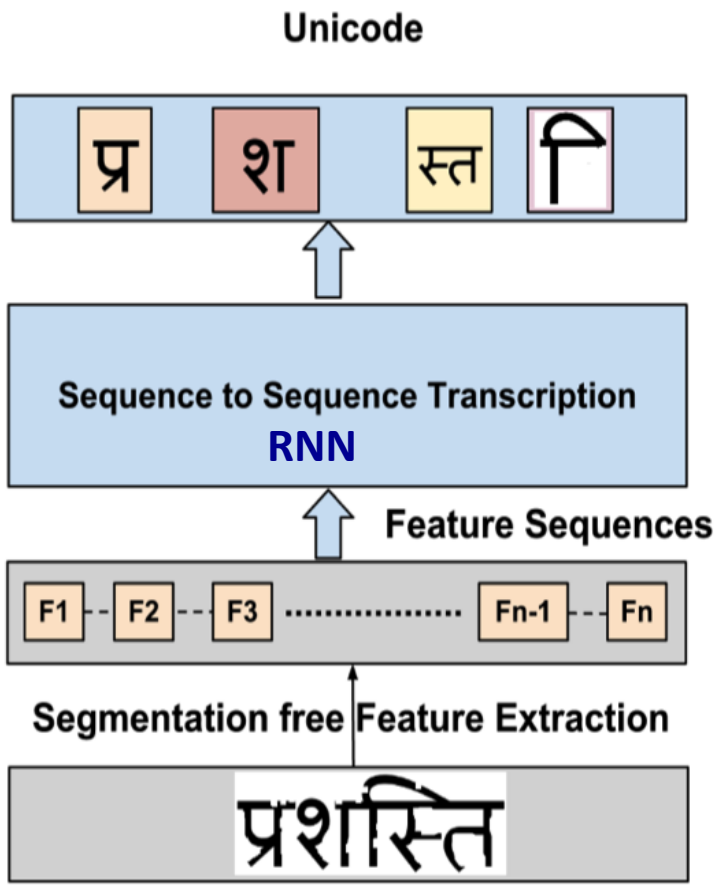
e.g. **image captioning**
image -> sequence of words

e.g. **machine translation**

OCR as Translation: RNN-OCR



Traditional OCR

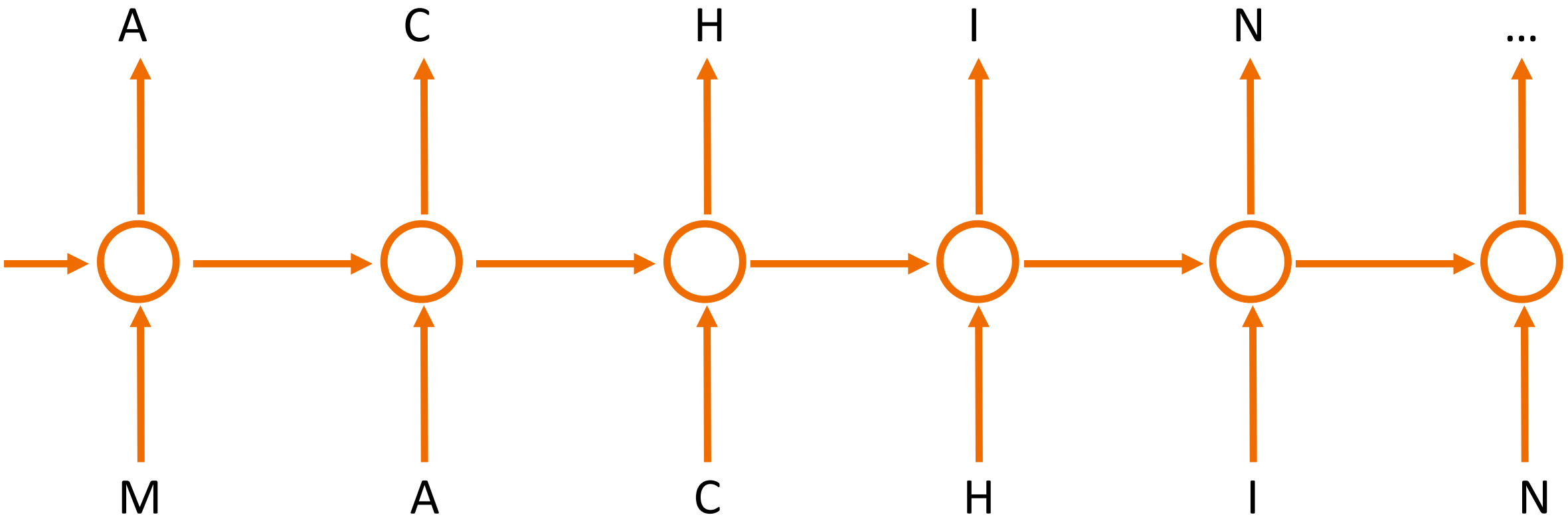


RNN OCR

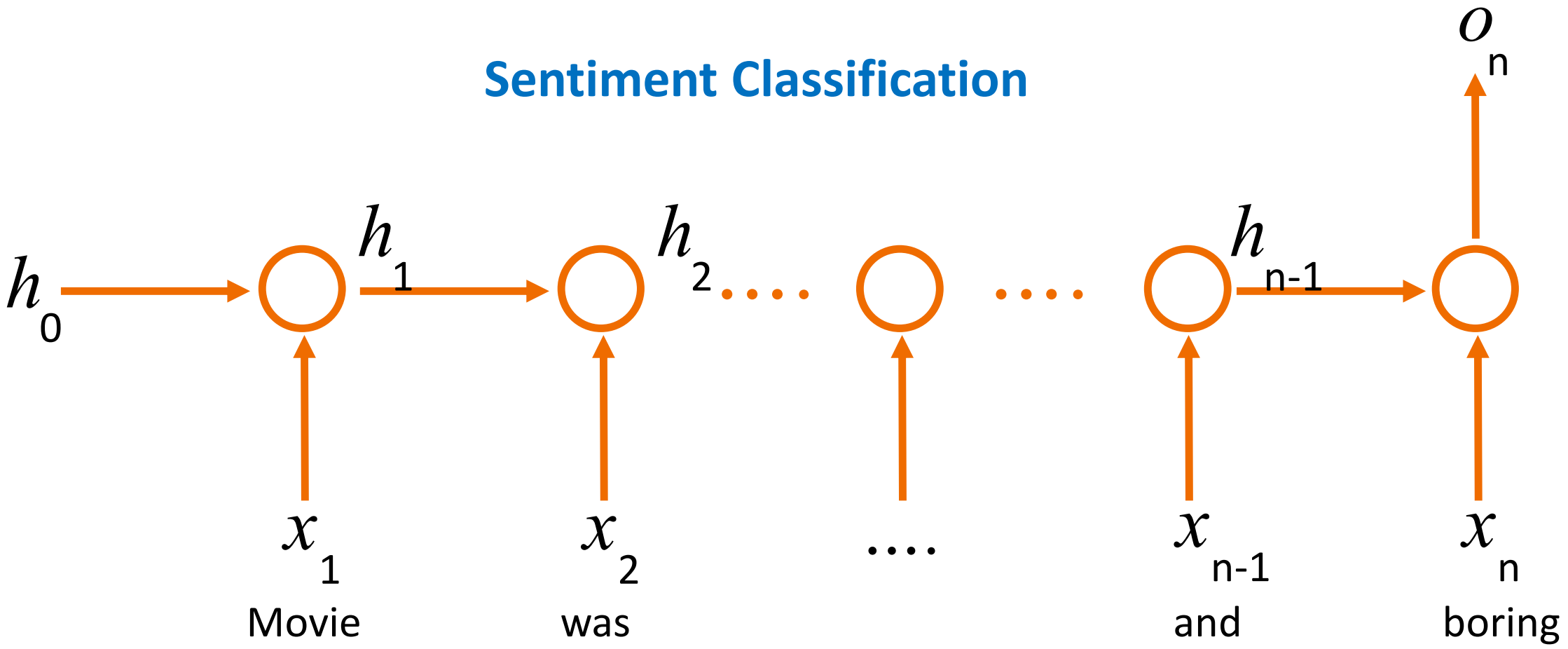
Overcome: Segmentation, Unicode Reordering,

Sankaran and Jawahar, Devanagari Text Recognition: A Transcription Based Formulation **ICDAR 2013**

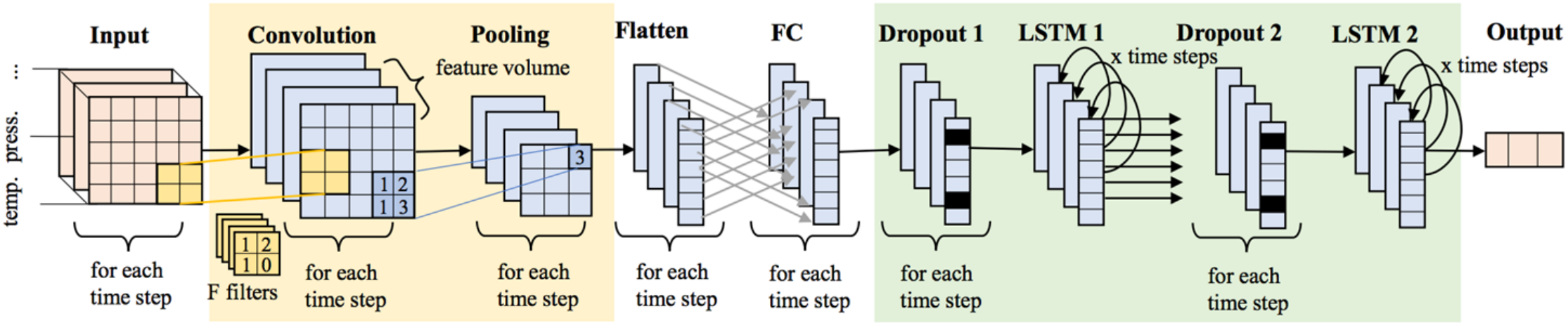
Lets design some Recurrent networks



Lets design some Recurrent networks

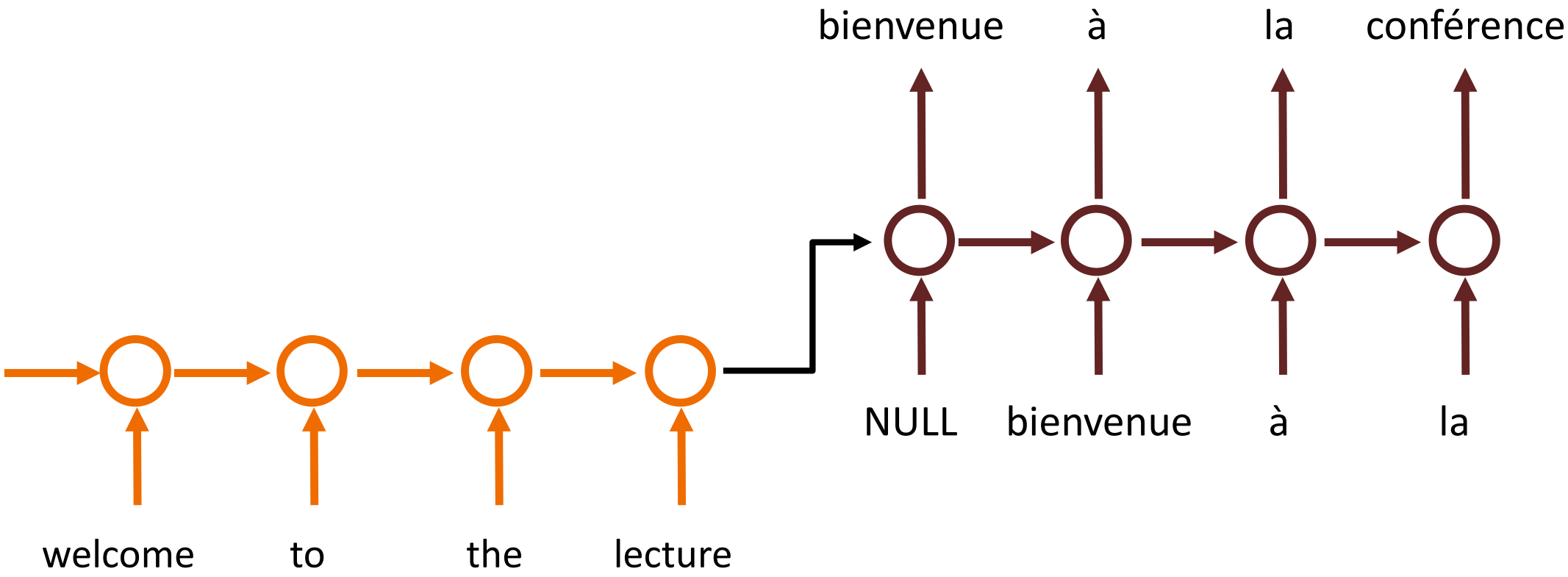


Weather Prediction



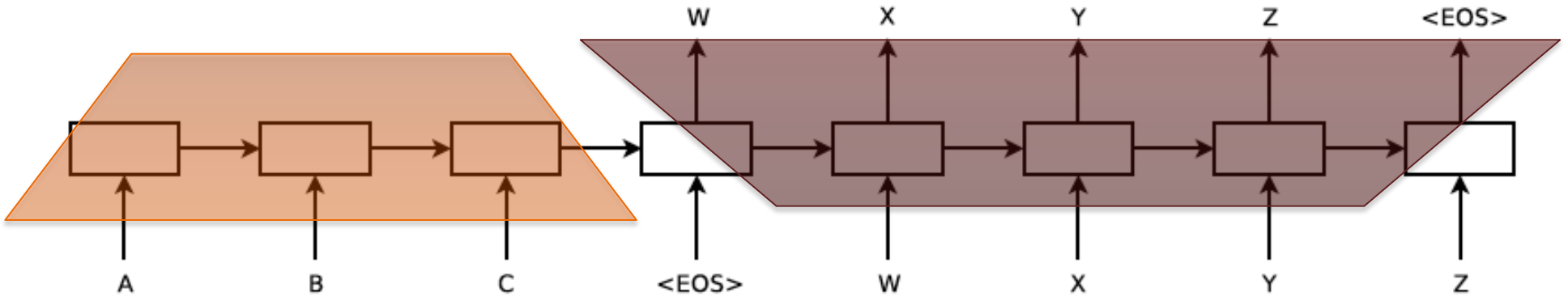
Roesch et al. Computer Graphics Forum, 2017

Lets design some Recurrent networks



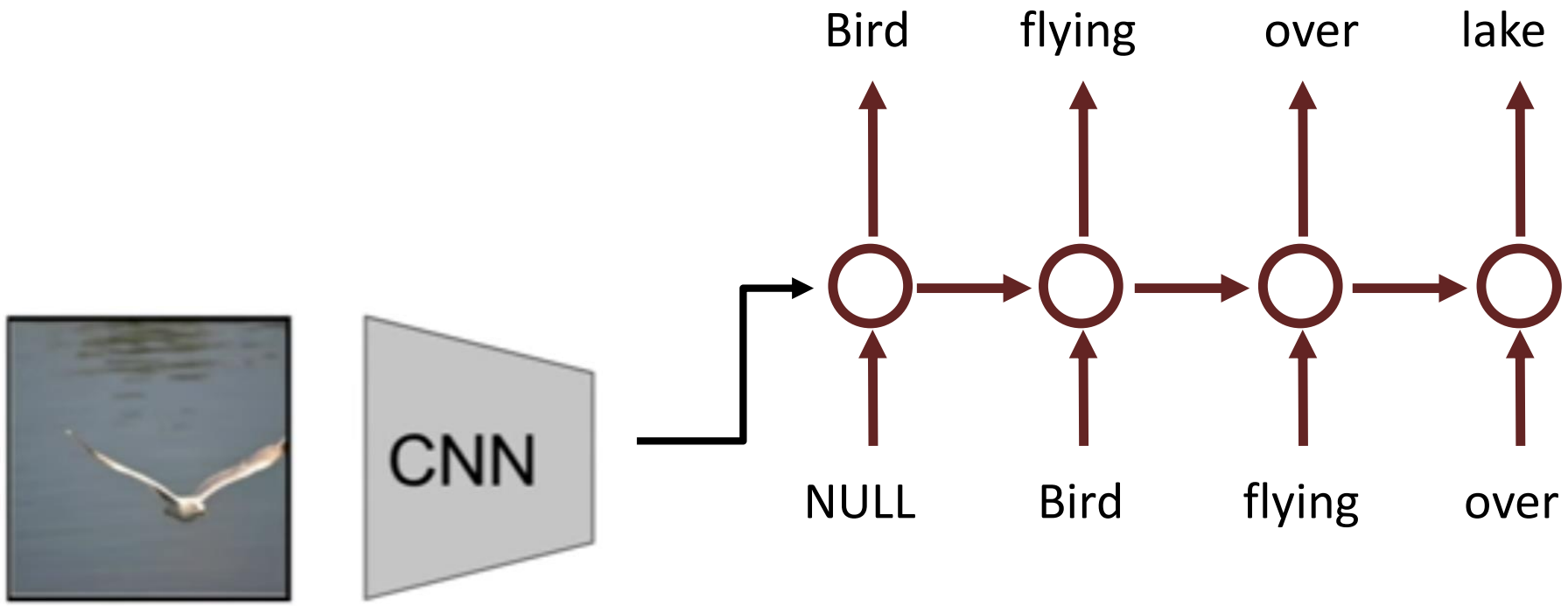
Neural Machine Translation

- Model



Sutskever et al. 2014

Lets design some Recurrent networks



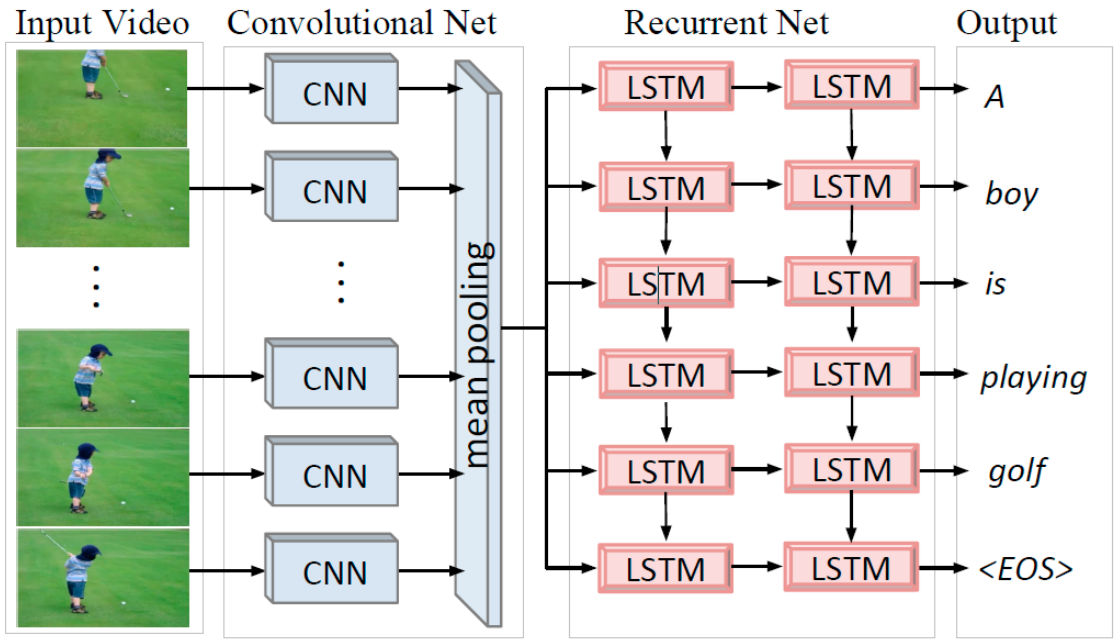
Machine learning with sequential data

- Input is an image and output is a sequence of words

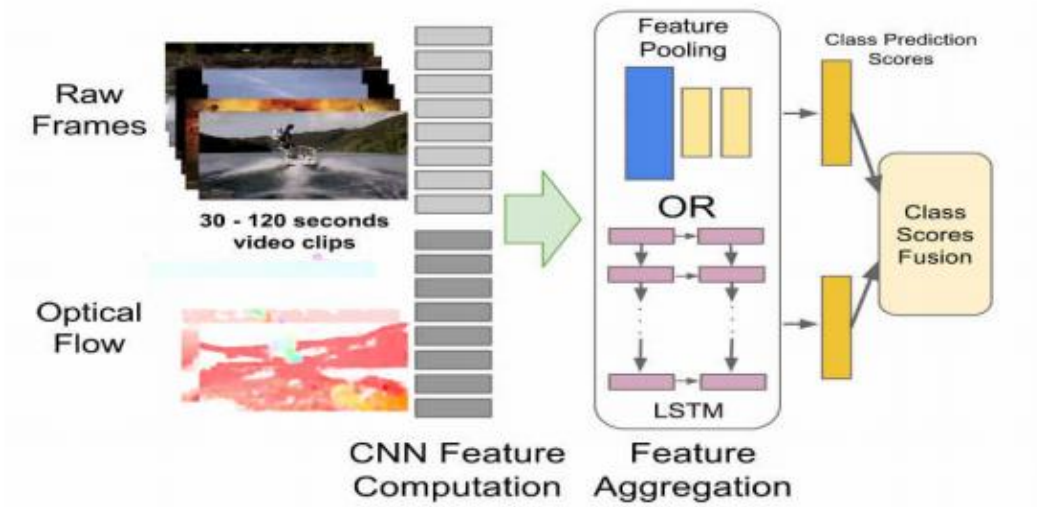
A horse carrying a large load of hay and two people sitting on it

Hybrid Architectures

Video Captioning

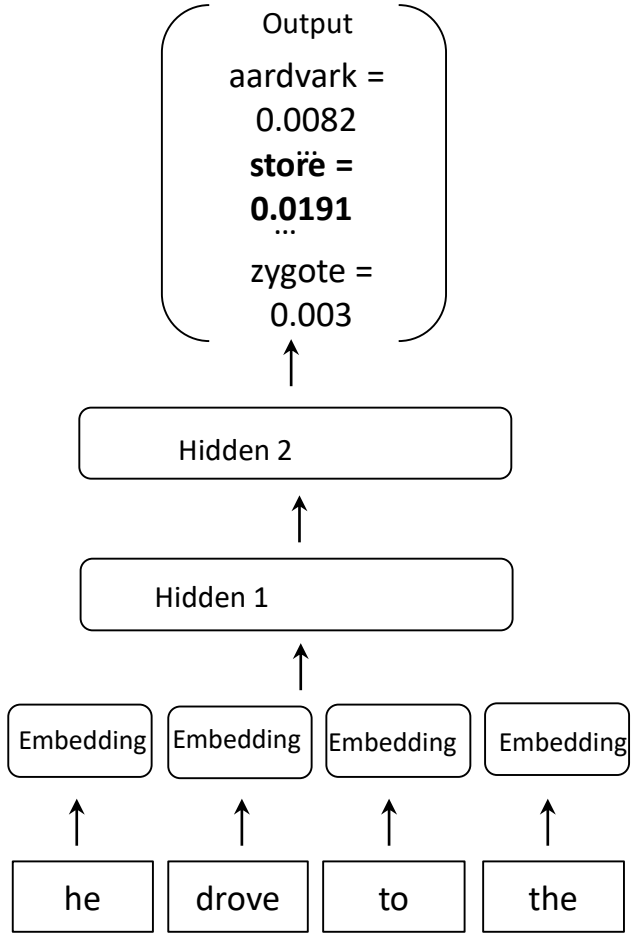


Video Classification

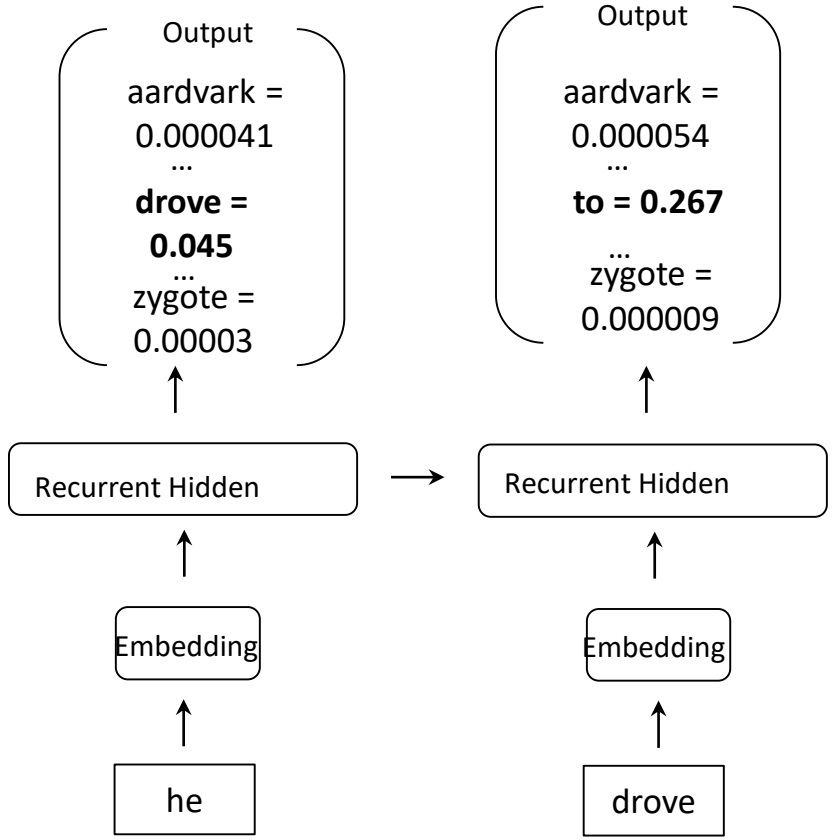


Neural Network Language Models (NNLMs)

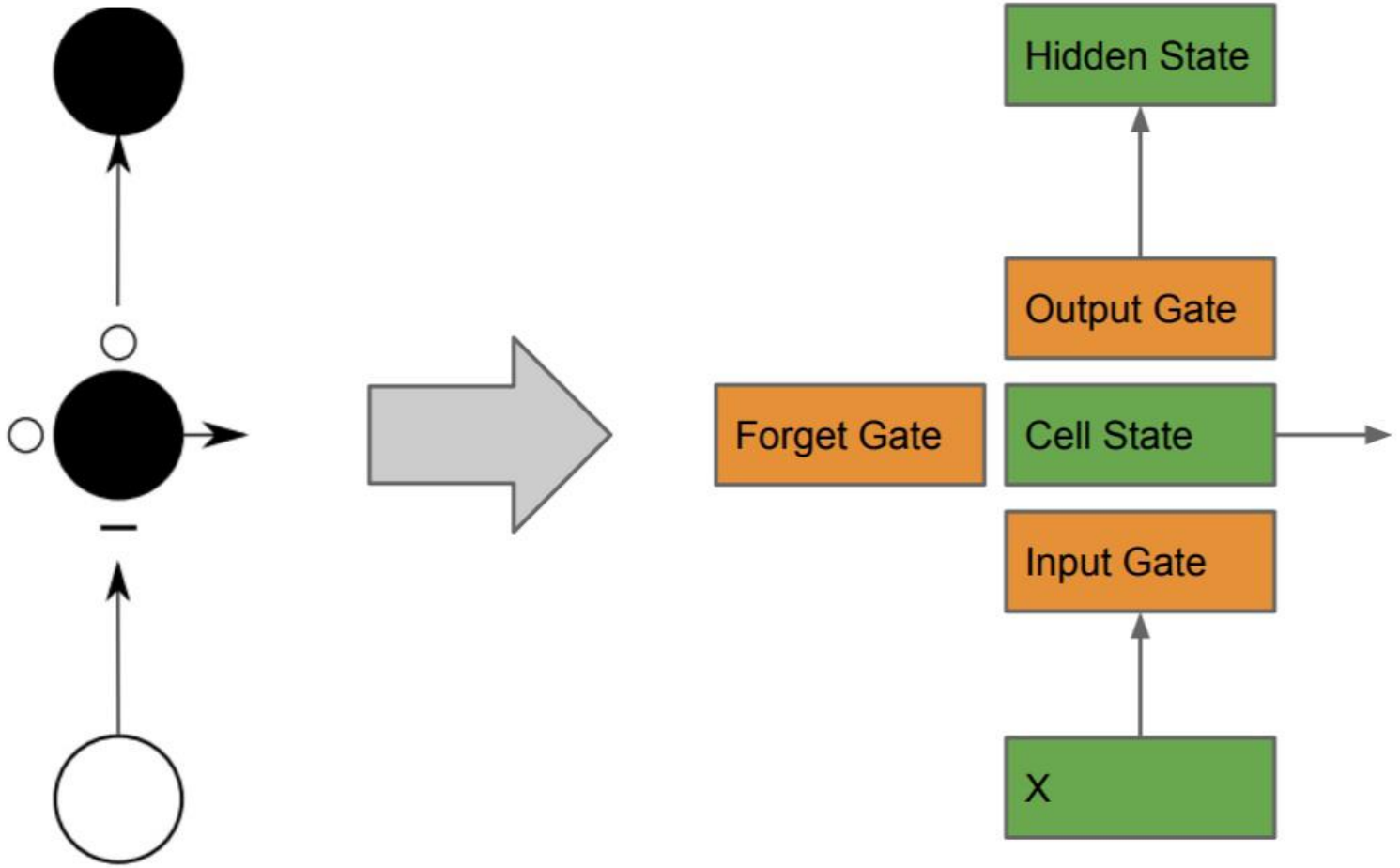
Feed-forward NNLM



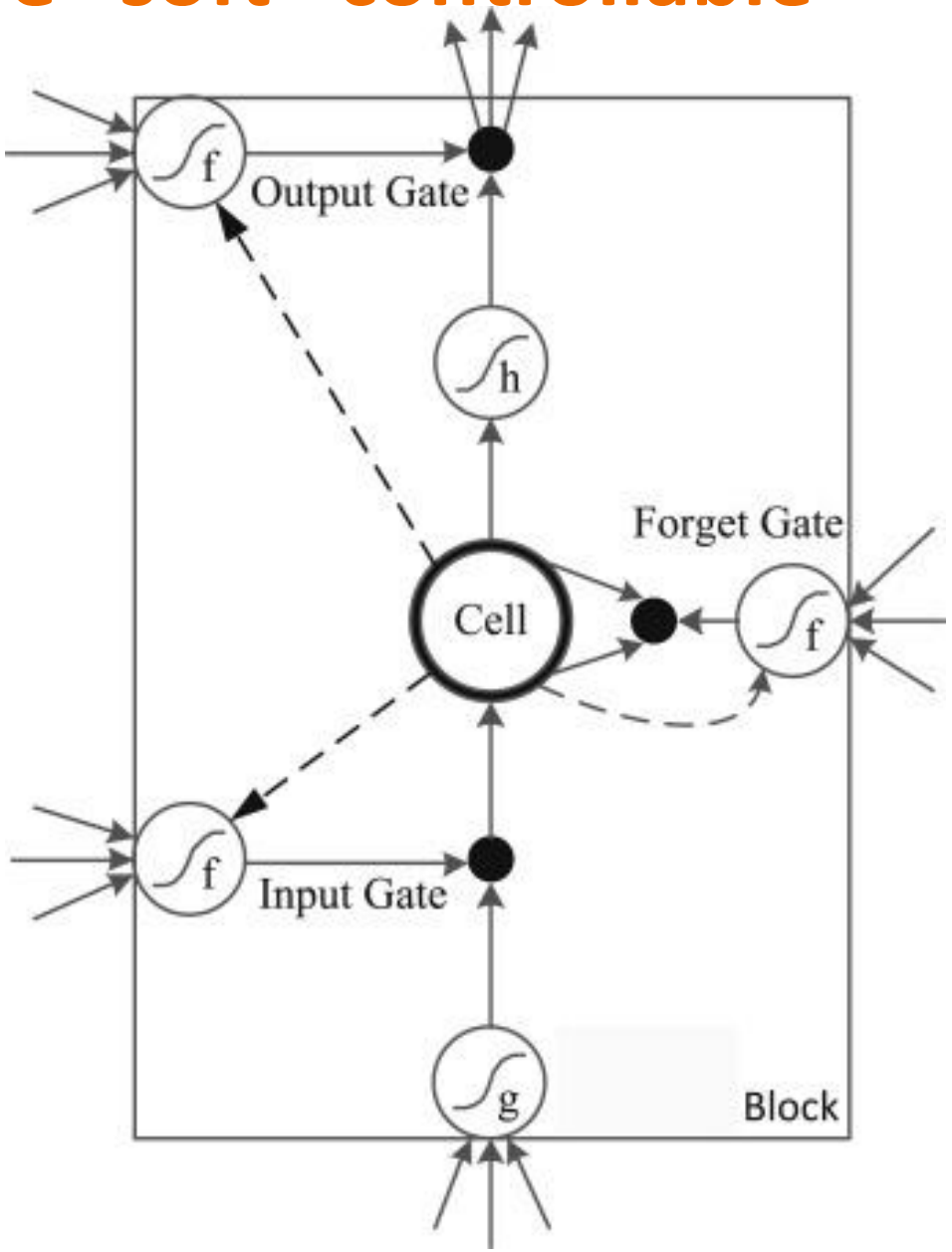
Recurrent NNLM



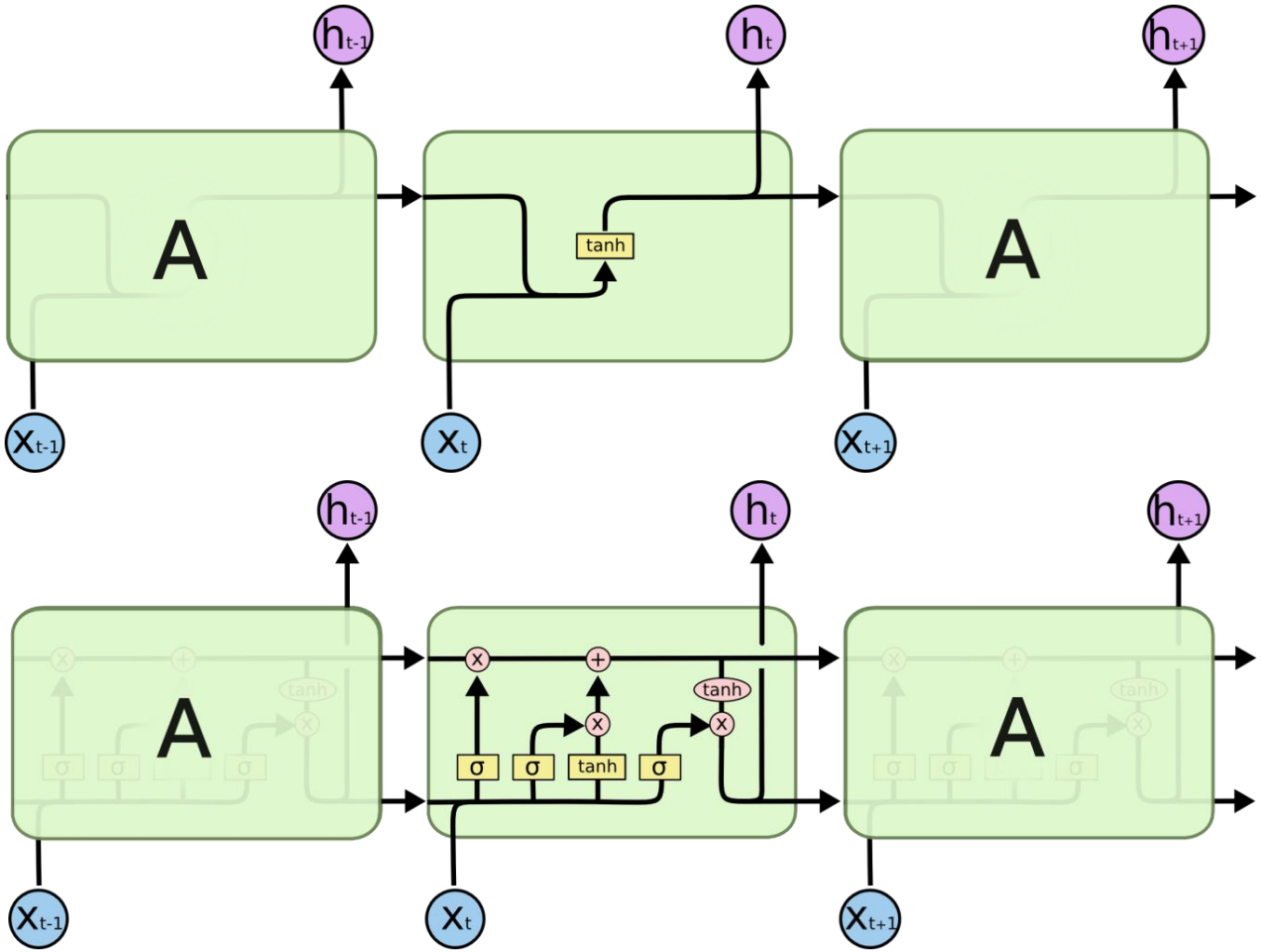
LSTM Node


Hidden State
Output Gate
Forget Gate
Cell State
Input Gate
X

LSTM : Gates are “soft” controllable



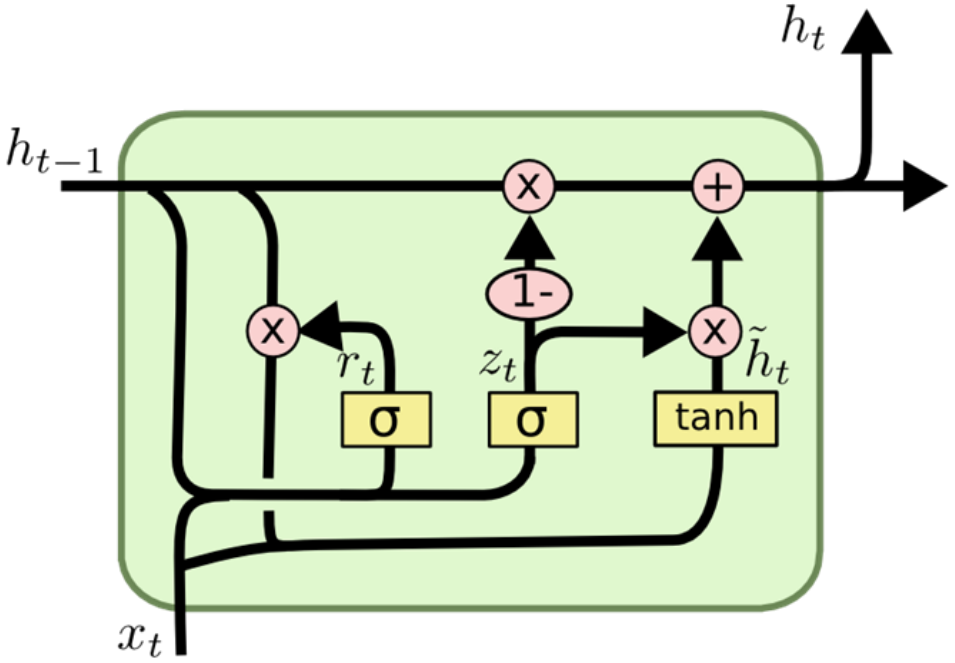
RNN vs LSTM's



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM's

Key take away: LSTMs are essentially the same thing as the RNN, they just have a different way of computing the hidden state.



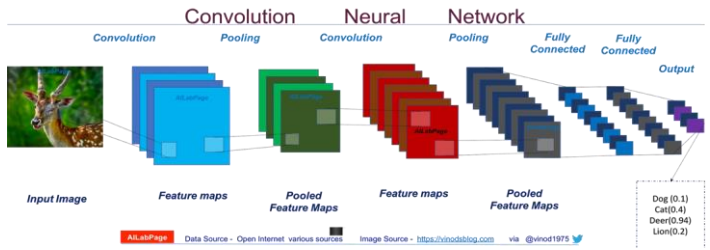
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

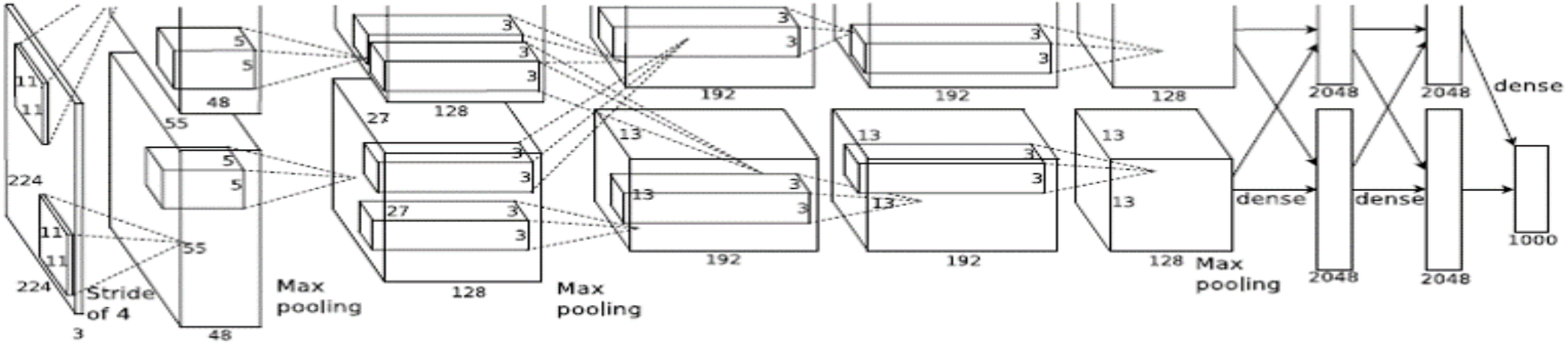
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/



Appreciating CNNs

Turning Point: AlexNet



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

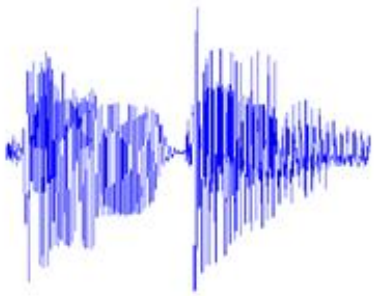
ImageNet Classification Task:

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

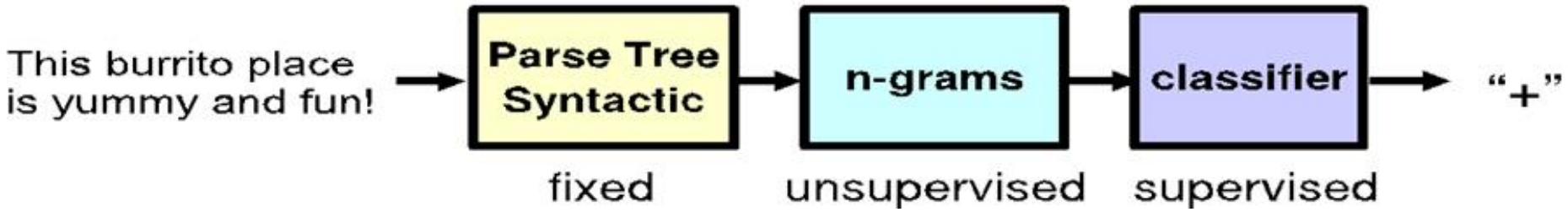
Common Pipeline: Till Then

- VISION:

- SPEECH:

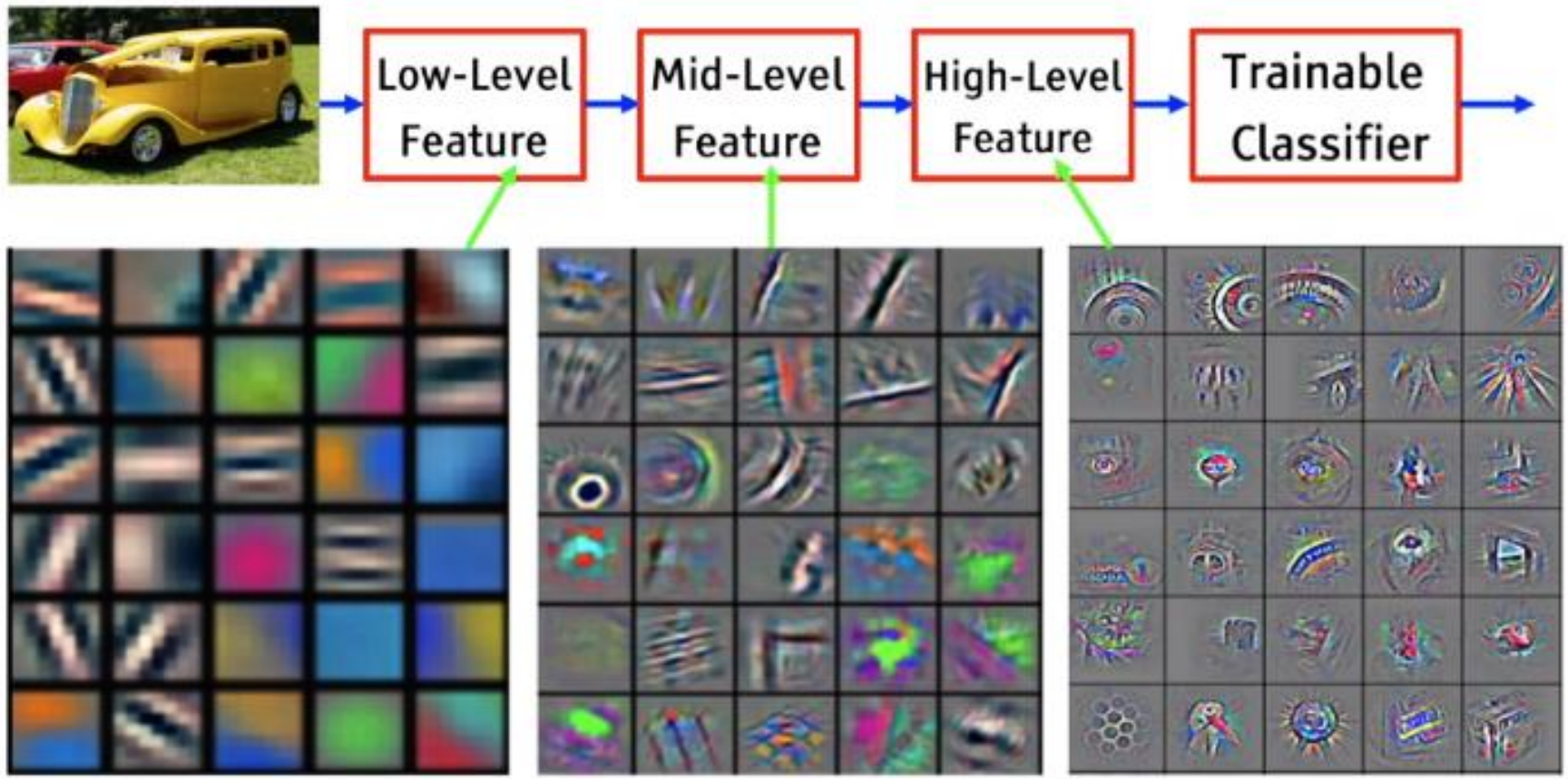


- NLP:



Deep Learnt Features

- It's deep if it has more than one stage of non-linear feature transformation.

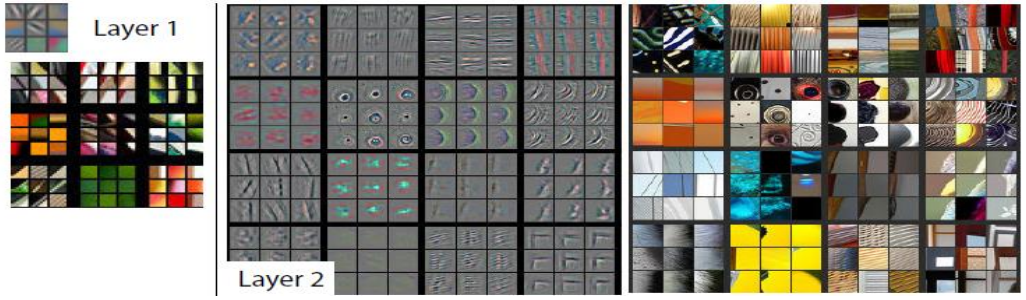


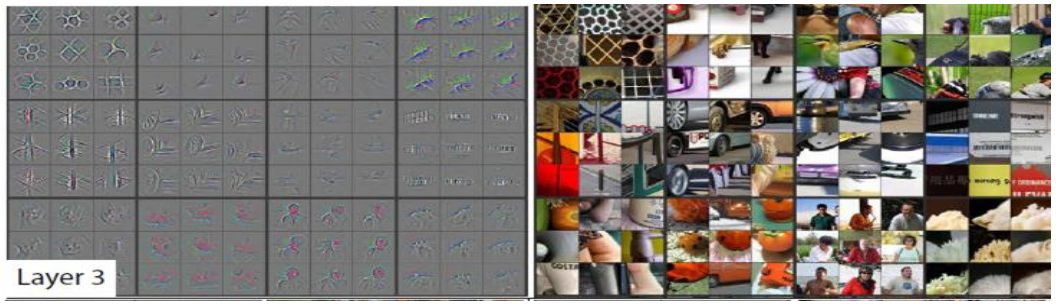
Learn the full pipeline

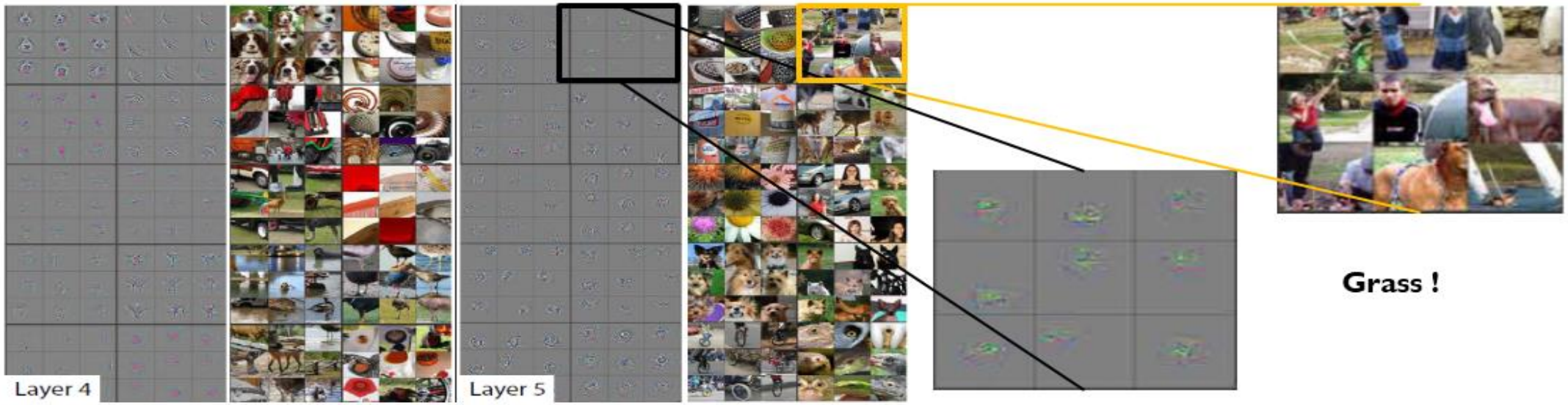
- VISION:
  - Pixels → edge → texton → motif → part → object
- SPEECH:
  - Sample → spectral → band → formant → motif → phone → word
- NLP:
  - Character → word → NP/VP/.. → Clause → sentence → story

Visualizing CNNs

A. How do I interpret the learned filters?







Source: Zeiler e.t. al. ECCV'14

Zeiler and Fergus , Visualizing and Understanding Convolutional Networks,. ECCV 2014

Early Layers Converge Faster

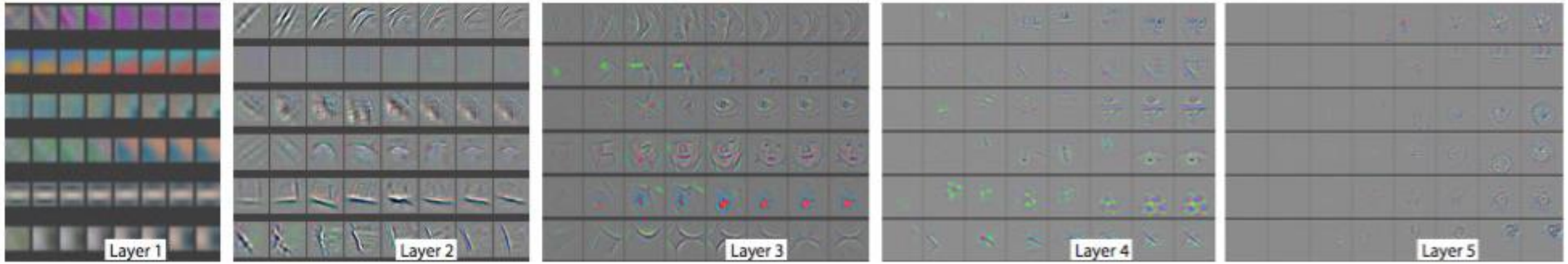
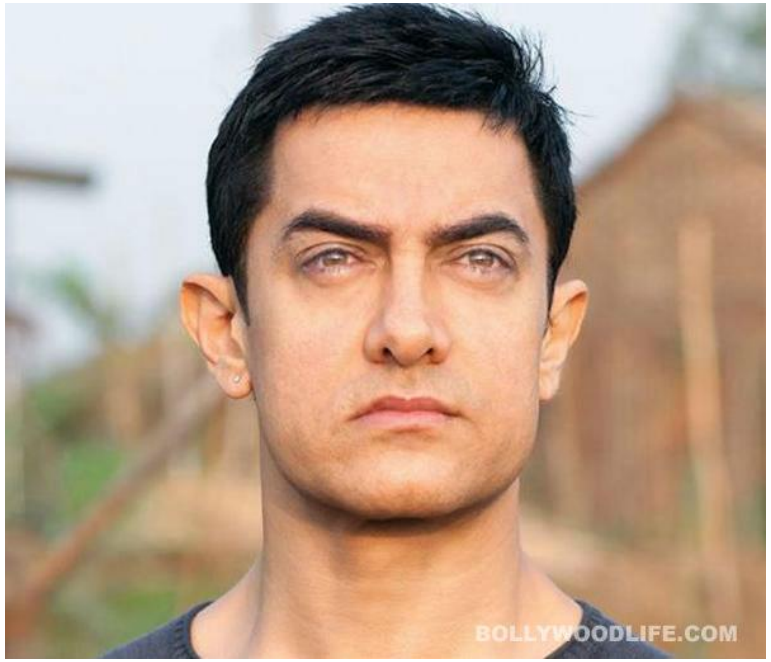
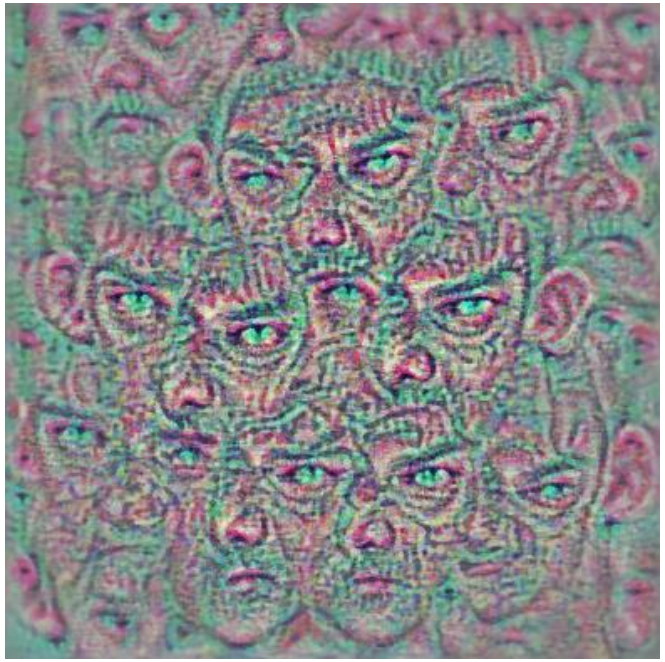


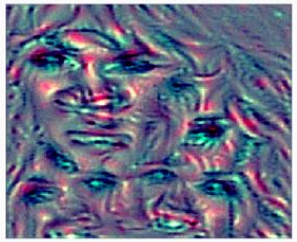
Figure: Evolution of randomly chosen subset of model features generated using deconvnet through training at epoch 1, 2, 5, 10, 20, 30, 40, 64.

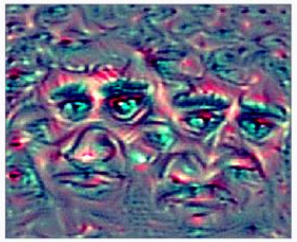
Deep Dream






(a) Class 93


(b) Class 301

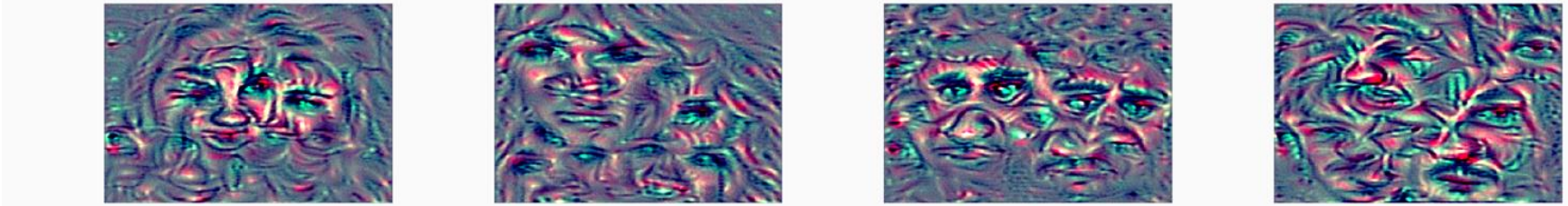

(c) Class 404


(d) Class 509

Simonyan et al. NIPS 2014,
Mahendran et al. CVPR 2015

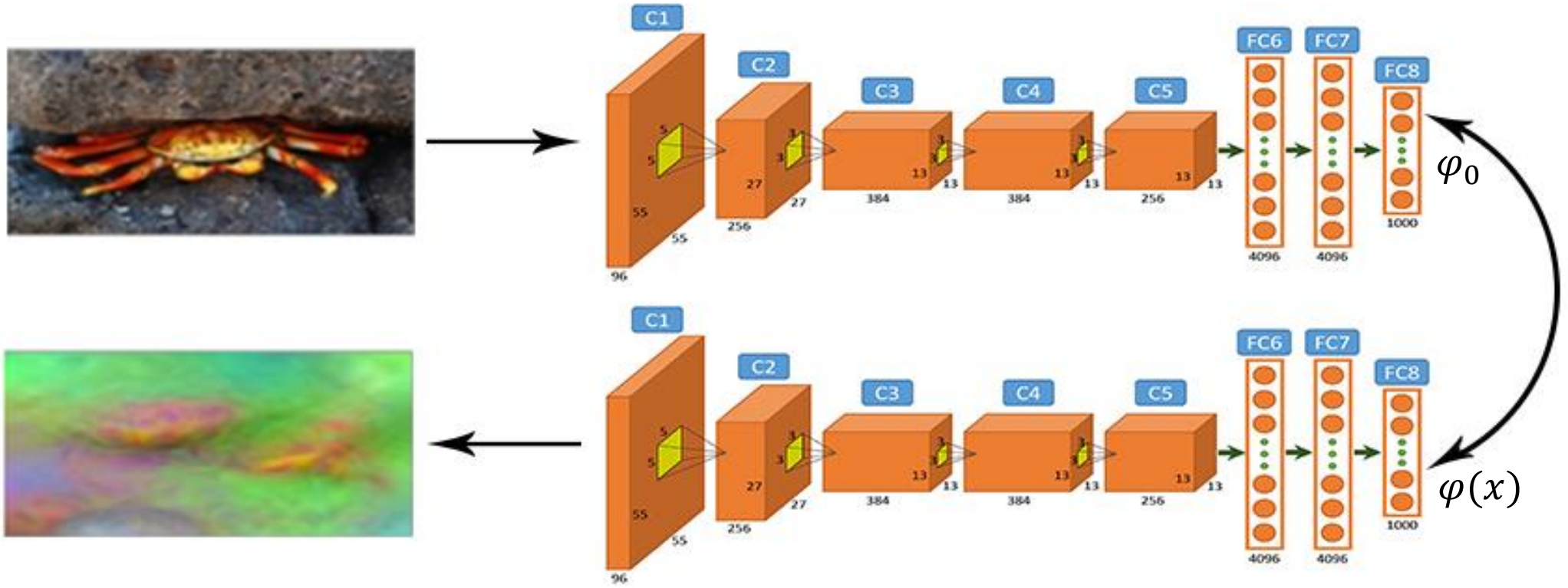
Visualizing CNNs

- Class Model Visualization
  - Find an L_2 normalized image which maximizes the C_l class score
  - Initialize with mean image.
  - Back-propagate.



Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. CoRR 2014

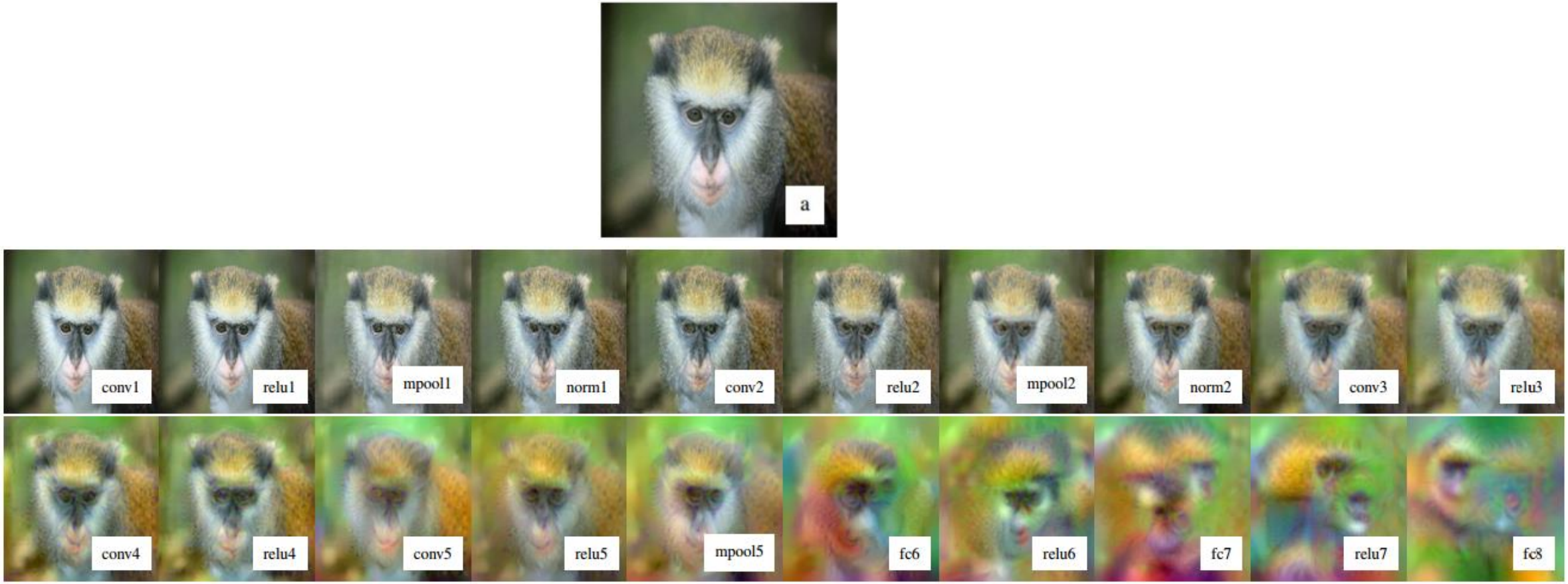
Inverting Specific Representation



$$x^* = \underset{x \epsilon R\ H \times W \times C}{argmin}\ L(\varphi(x), \varphi_0) + \lambda R(x)$$

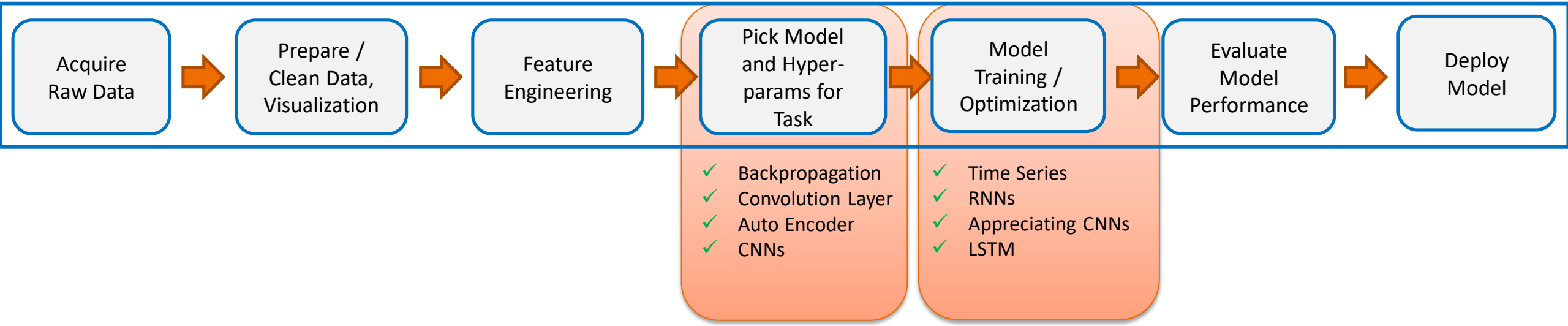
Aravindh Mahendran and Andrea Vedaldi, Understanding Deep Image Representations by Inverting Them, CVPR'15

Inverting at Different Stages



Reconstructions from intermediate layers

Summary


Acquire Raw Data
Prepare / Clean Data, Visualization
Feature Engineering
Pick Model and Hyper-params for Task
Model Training / Optimization
Evaluate Model Performance
Deploy Model
✓ Backpropagation
✓ Convolution Layer
✓ Auto Encoder
✓ CNNs
✓ Time Series
✓ RNNs
✓ Appreciating CNNs
✓ LSTM

Thanks!!

Questions?

What’s ahead: Unit-4

Unit-4

- Transfer Learning
- Fast API Deployment
- Beyond AlexNet
- Advances in Training
- Deployment and Practical Issues
- Model Compression
- Siamese Networks
- Advanced Topics: GANs
- Computer Vision