# Recall from last time ...

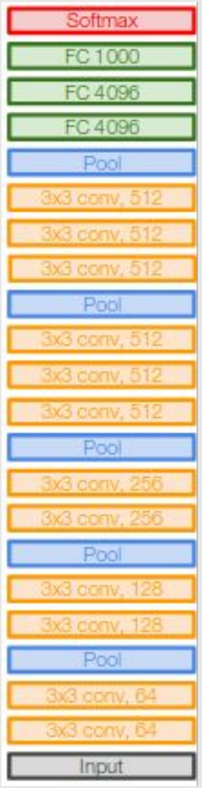
# Transfer Learning

"You need a lot of a data if you want to train/use CNNs"

# Transfer Learning

“You need a [illegible] data if you want to t[illegible]e CNNs”

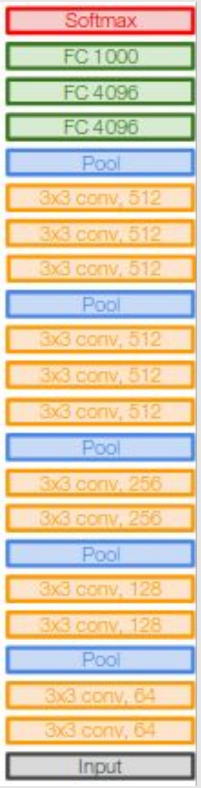
# Transfer Learning with CNNs




**Feature Extractor**

1. Train on ImageNet

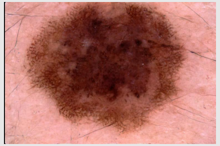
# Transfer Learning with CNNs



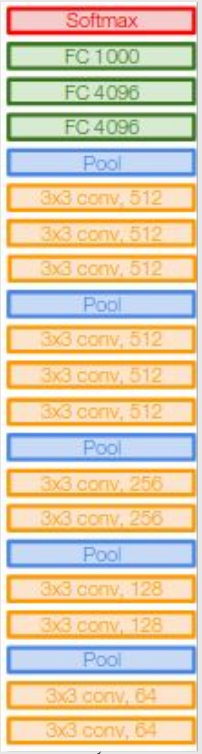

**Feature Extractor**

1. Train on ImageNet

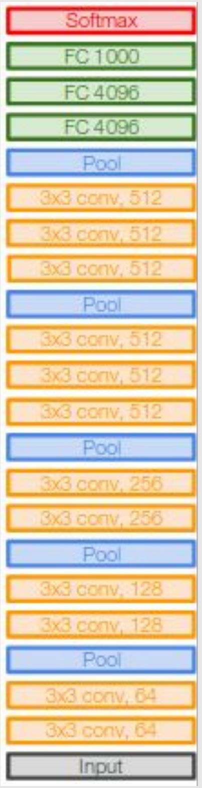
Skin lesion



**input**



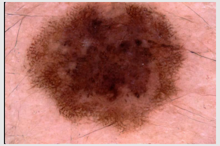
# Transfer Learning with CNNs



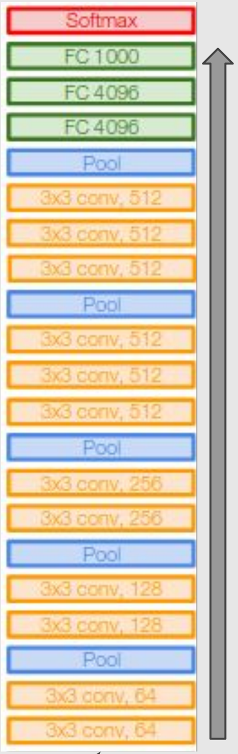

**Feature Extractor**

1. Train on ImageNet

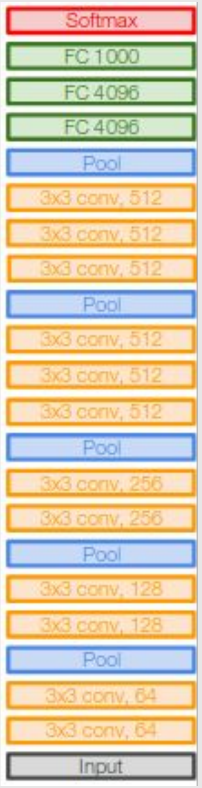
Skin lesion



**input**



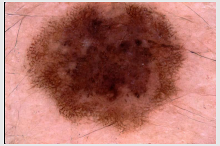
# Transfer Learning with CNNs




**Feature Extractor**

1. Train on ImageNet

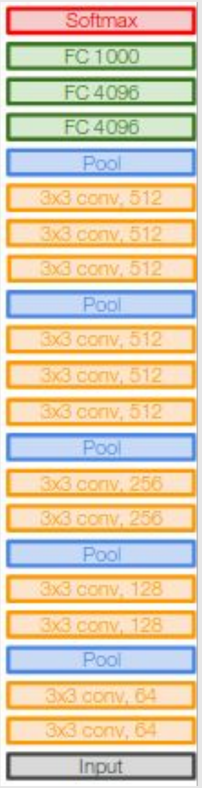
Skin lesion



**input**

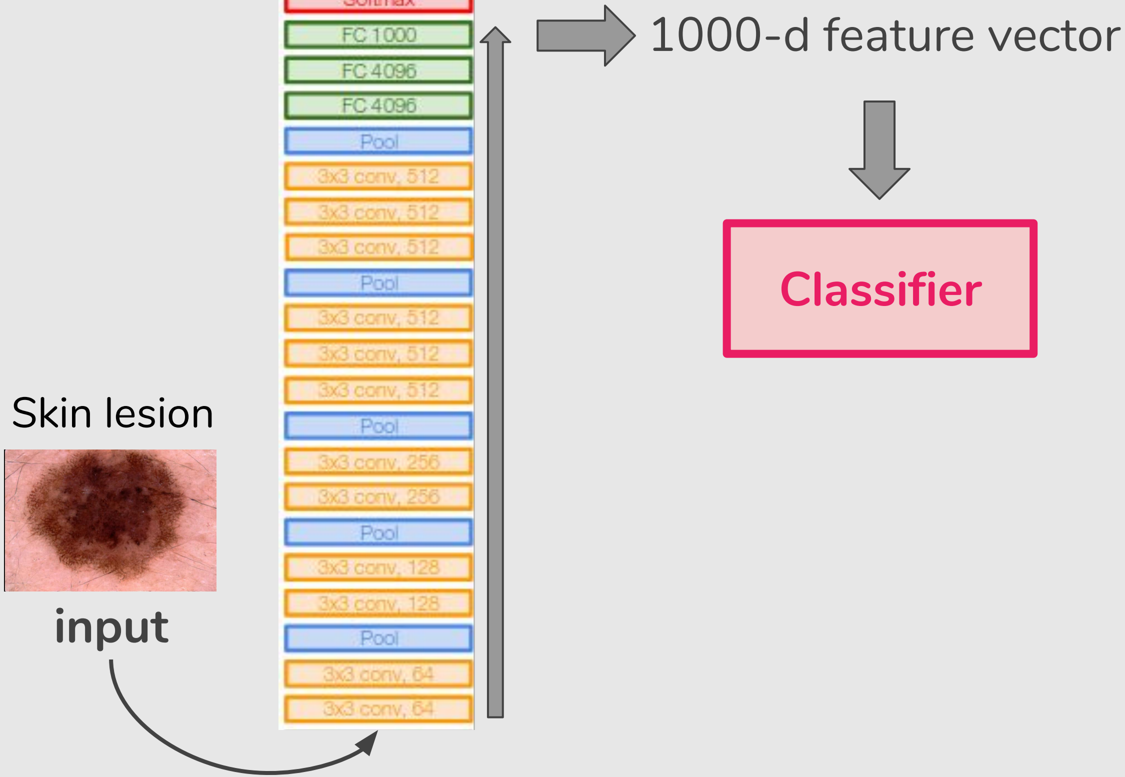

1000-d feature vector

# Transfer Learning with CNNs

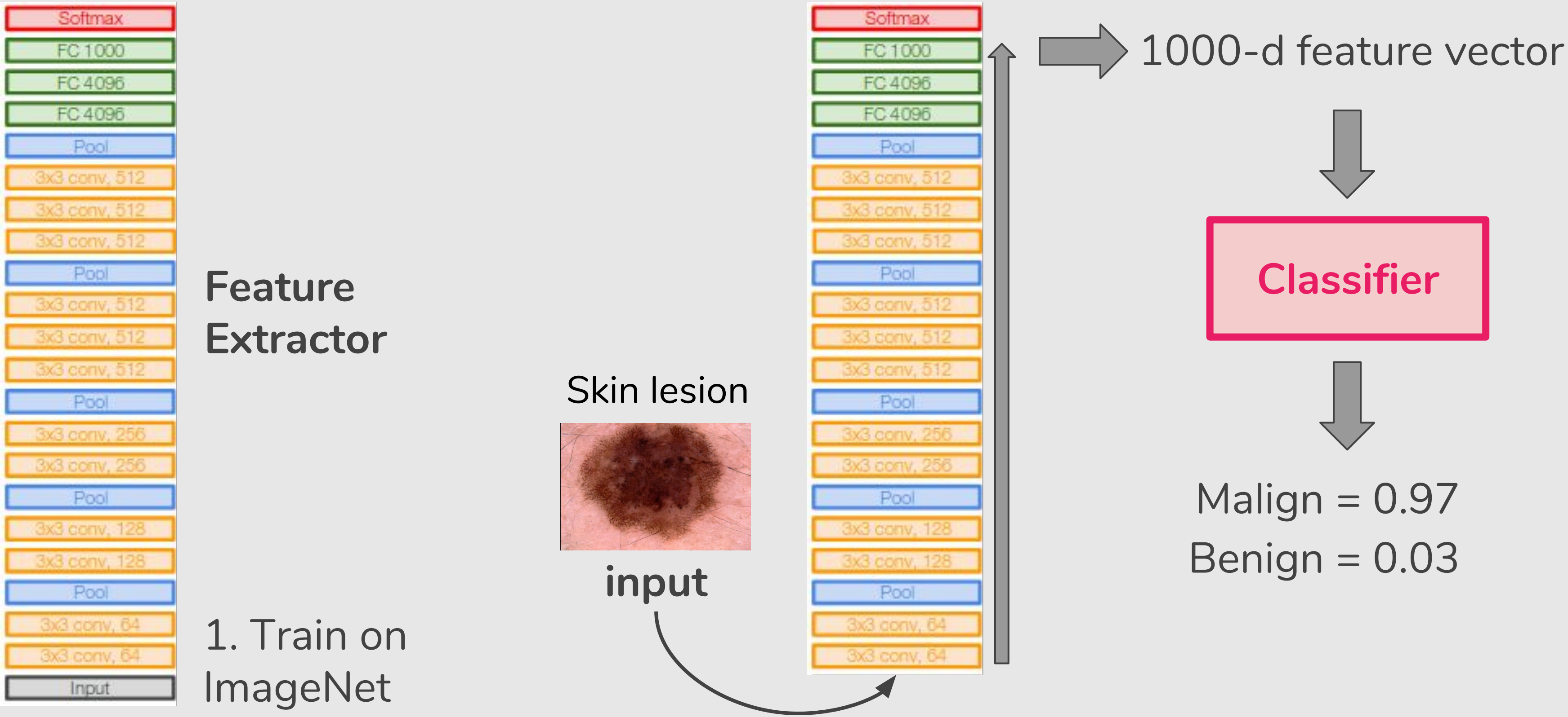



**Feature Extractor**

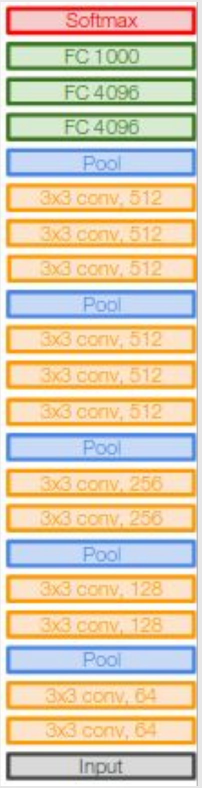
1. Train on ImageNet



# Transfer Learning with CNNs



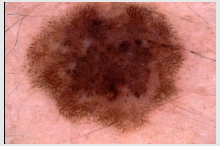
# Transfer Learning with CNNs



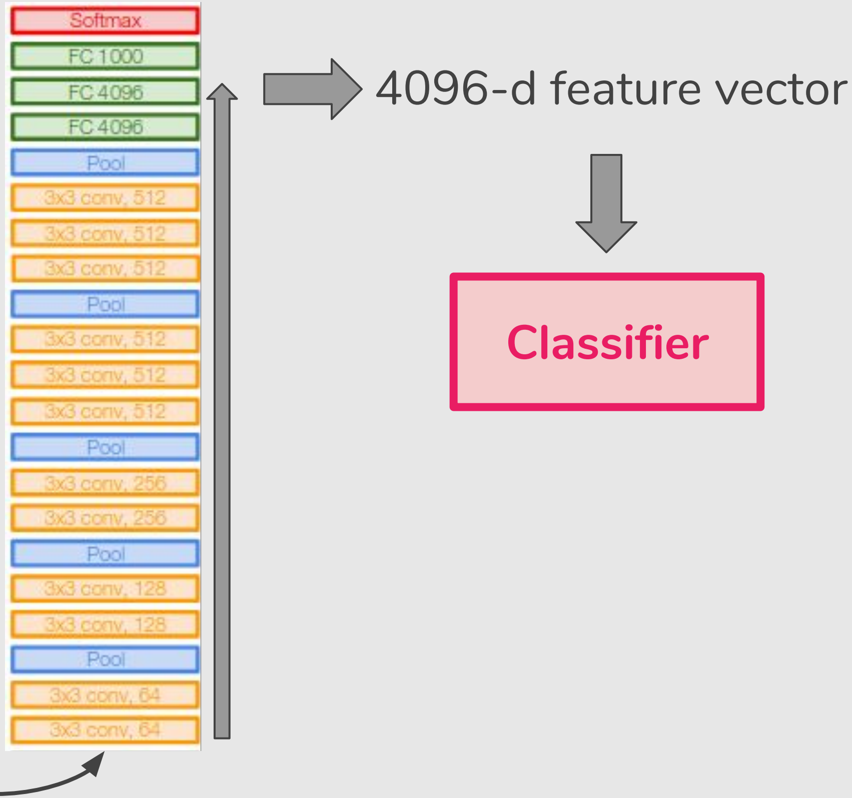

**Feature Extractor**

1. Train on ImageNet

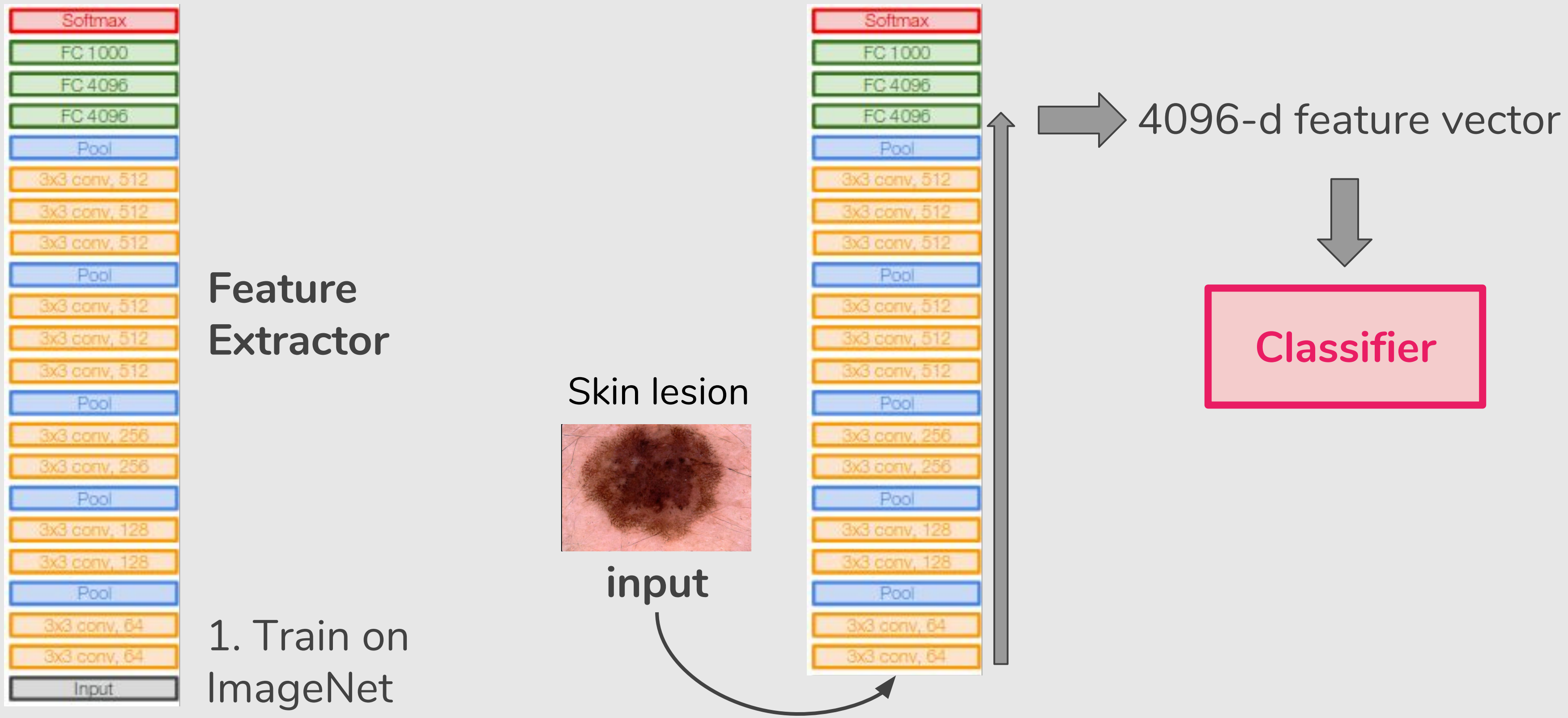
Skin lesion



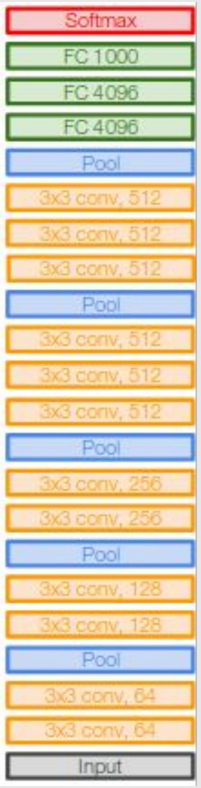
**input**



# Transfer Learning with CNNs



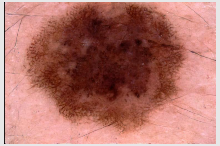
# Transfer Learning with CNNs



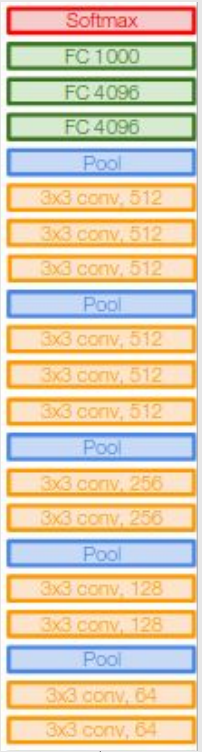

**Feature Extractor**

1. Train on ImageNet

Skin lesion



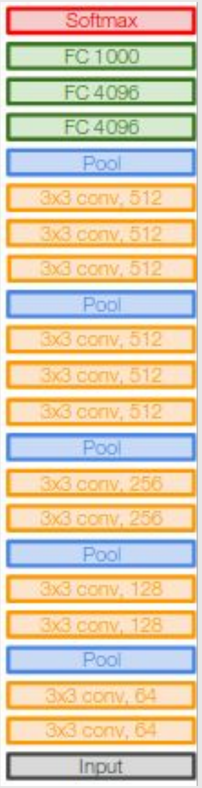
**input**




9216-d feature vector

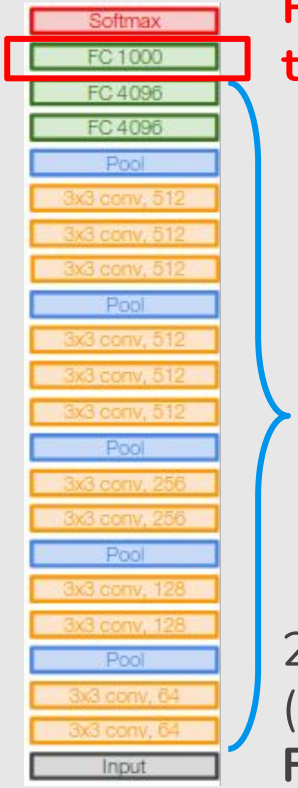
**Classifier**

# Transfer Learning with CNNs




**Feature Extractor**

1. Train on ImageNet



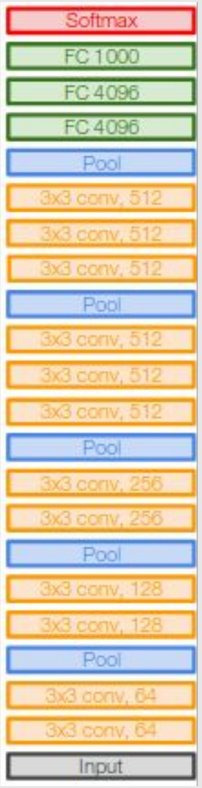

**Reinitialize this and train**

Freeze these

2. Small dataset (C classes)

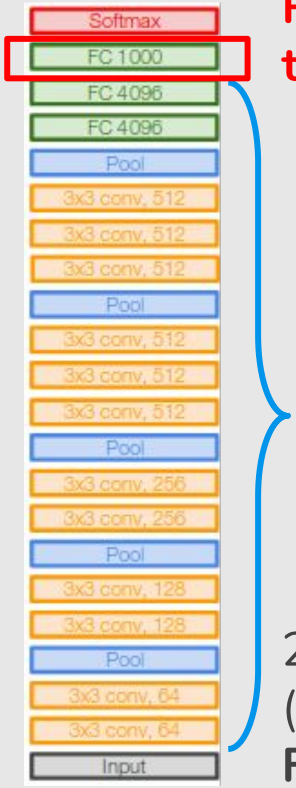
**Finetuning**

# Transfer Learning with CNNs




**Feature Extractor**

1. Train on ImageNet



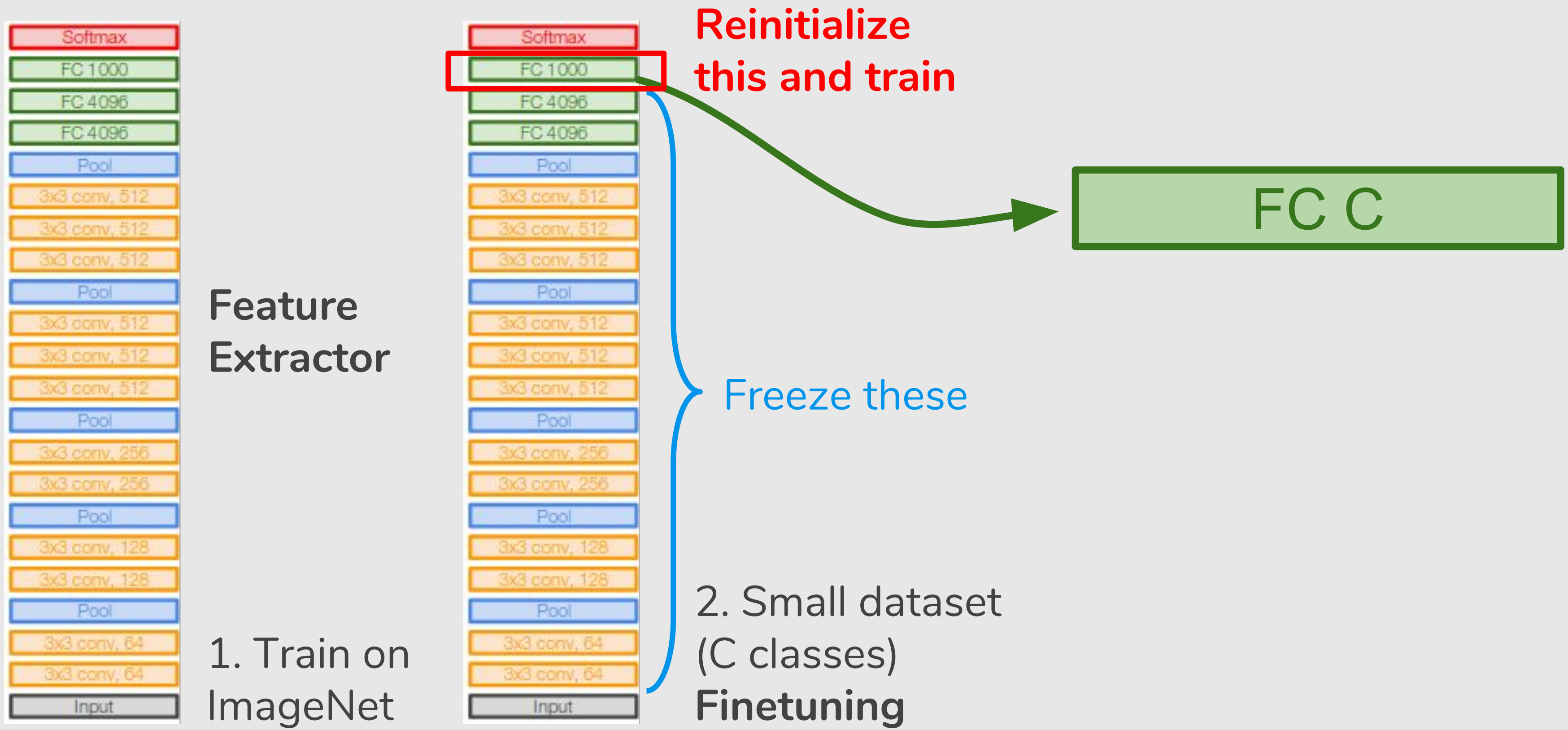

**Reinitialize this and train**

Freeze these

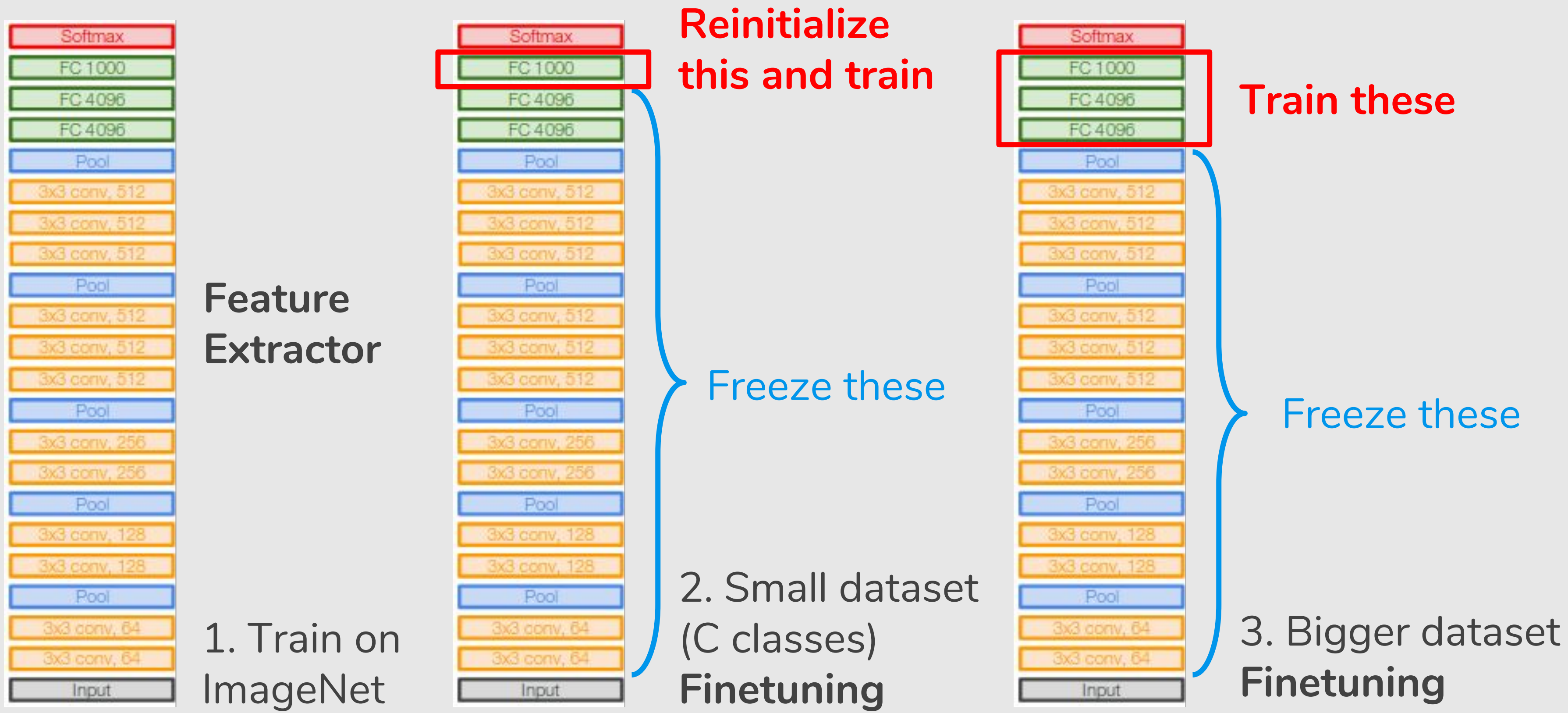
2. Small dataset (C classes)

**Finetuning**

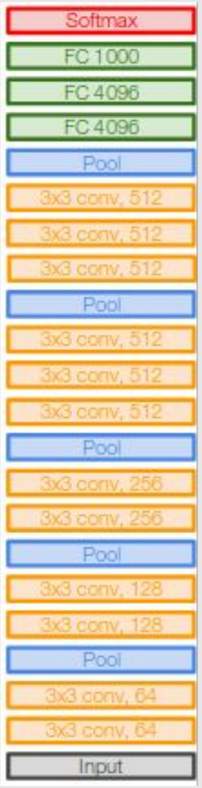
# Transfer Learning with CNNs



# Transfer Learning with CNNs

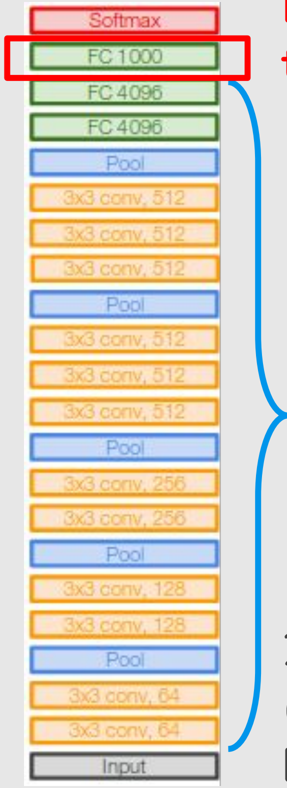


# Transfer Learning with CNNs




**Feature Extractor**

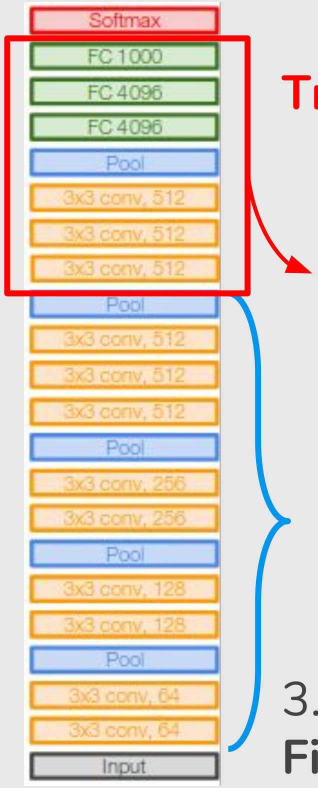
1. Train on ImageNet




**Reinitialize this and train**

Freeze these

2. Small dataset (C classes)
**Finetuning**



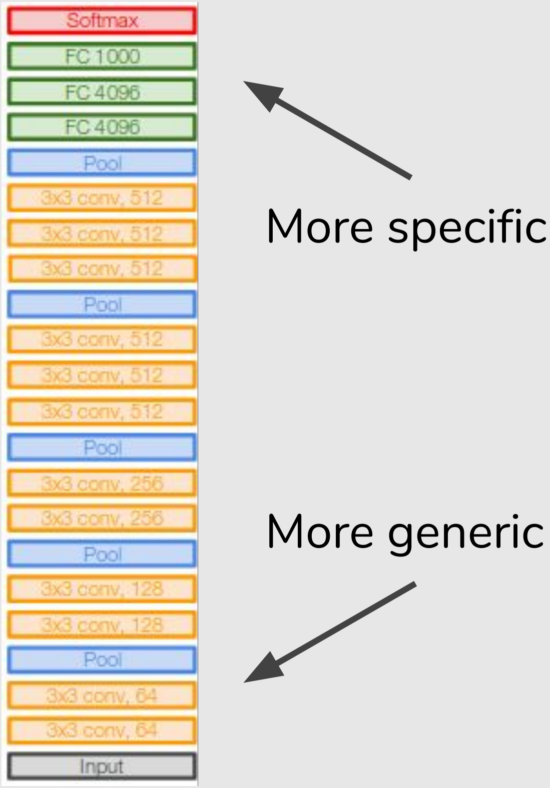

**Train these**

With bigger dataset, train more layers

Freeze these

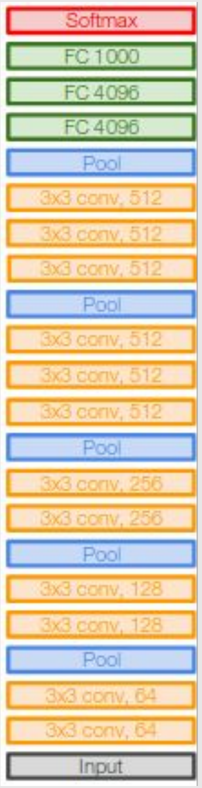
3. Bigger dataset
**Finetuning**

# Transfer Learning with CNNs




| | Very similar dataset | Very different dataset |
|---|---|---|
| Very little data | ? | ? |
| Quite a lot of data | ? | ? |

# Transfer Learning with CNNs

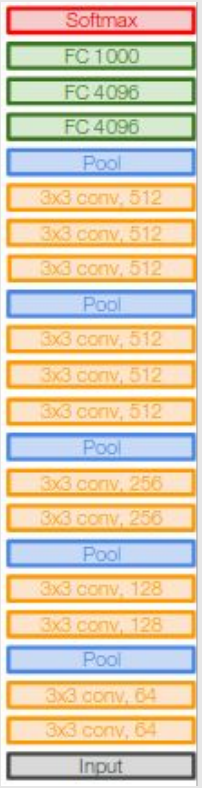



More specific

More generic

| | Very similar dataset | Very different dataset |
|---|---|---|
| Very little data | Use Linear Classifier on top layer | ? |
| Quite a lot of data | Finetune a few layers | ? |

# Transfer Learning with CNNs

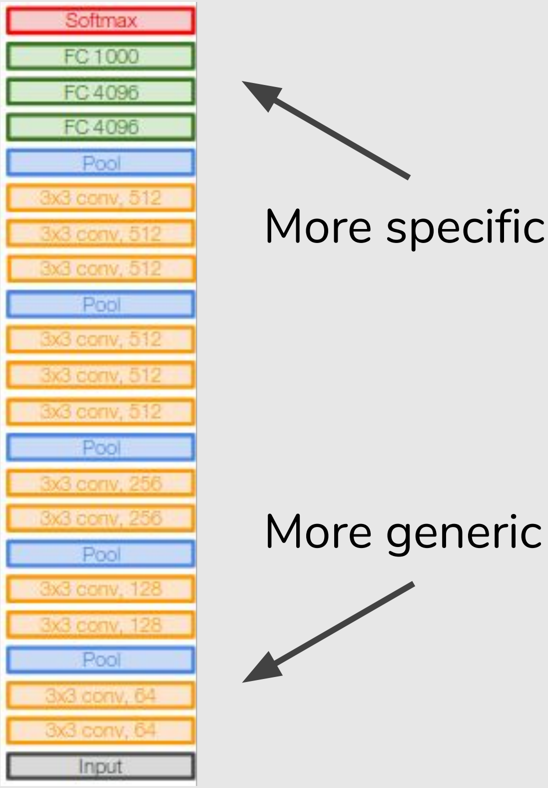



More specific

More generic

| | Very similar dataset | Very different dataset |
|---|---|---|
| Very little data | Use Linear Classifier on top layer | ? |
| Quite a lot of data | Finetune a few layers | Finetune a larger number of layers |

# Transfer Learning with CNNs




| | Very similar dataset | Very different dataset |
|---|---|---|
| Very little data | Use Linear Classifier on top layer | You're in trouble ... Try linear classifier from different stages |
| Quite a lot of data | Finetune a few layers | Finetune a larger number of layers |

# Take away for your projects and beyond:

Have some dataset of interest but it has < ~1M images?

1. Find a very large dataset that has similar data, train a big CNN there
2. Transfer learn to your dataset

# Take away for your projects and beyond:

Have some dataset of interest but it has < ~1M images?

1. Find a very large dataset that has similar data, train a big CNN there
2. Transfer learn to your dataset

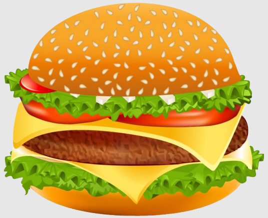
Deep learning frameworks provide a “Model Zoo” of pretrained models so you don’t need to train your own.

TensorFlow: https://github.com/tensorflow/models
PyTorch: https://github.com/pytorch/vision

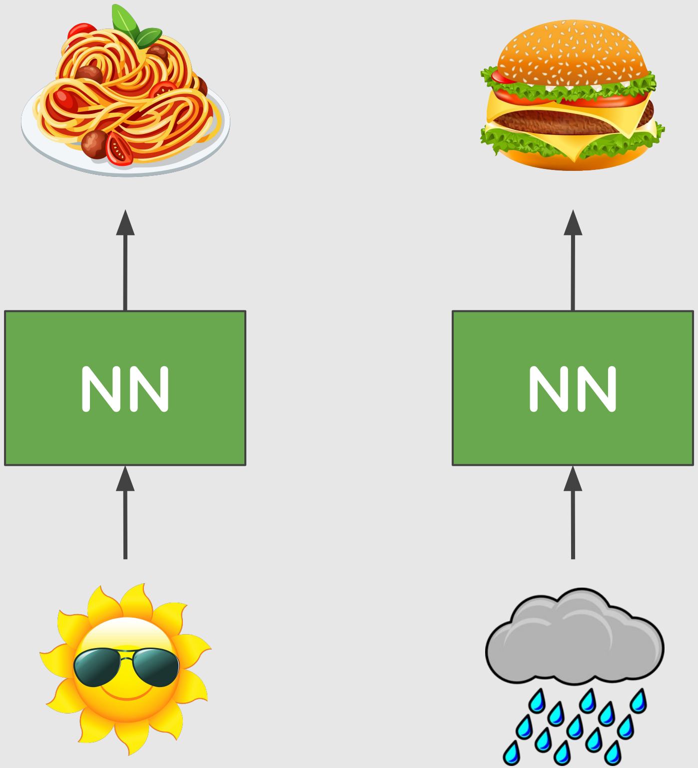
# Recurrent Neural Networks (RNNs)

Weather

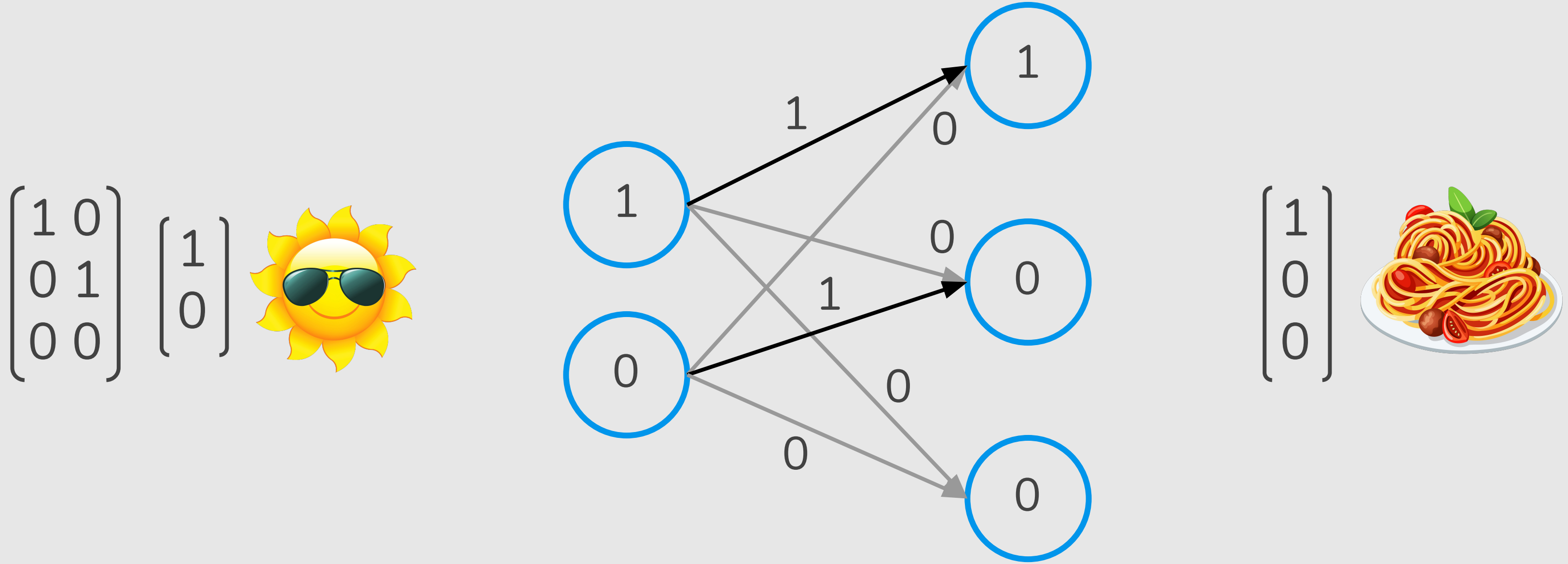
Perfect Roommate



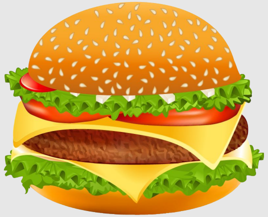
# Neural Network



# Neural Network




Pasta

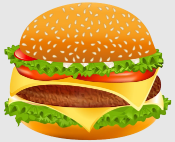


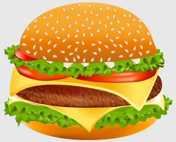
Burger

Salad

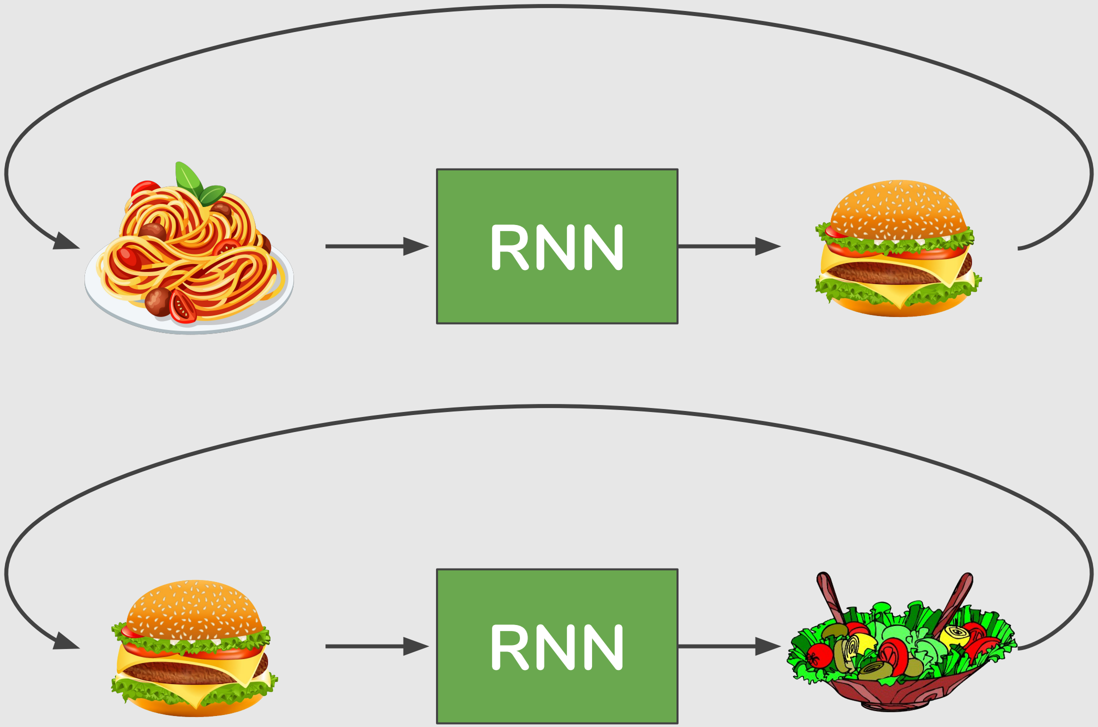

# Cooking Schedule

Monday Tuesday Wednesday Thursday Friday Saturday

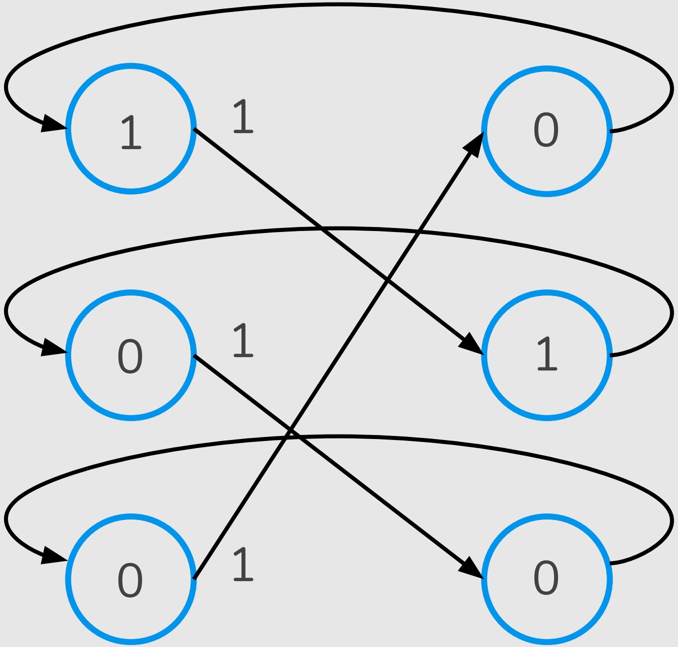




# Recurrent Neural Network

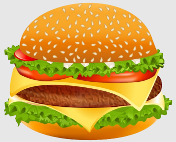


# Simple (Recurrent) Neural Network

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$




$$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

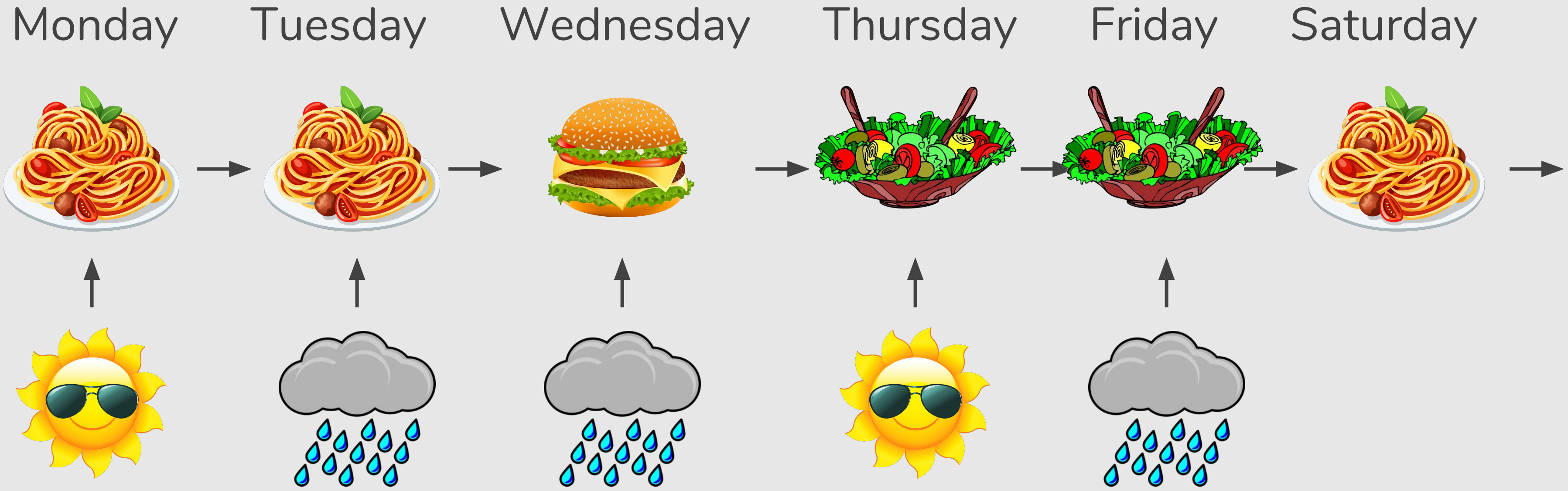


# Weather

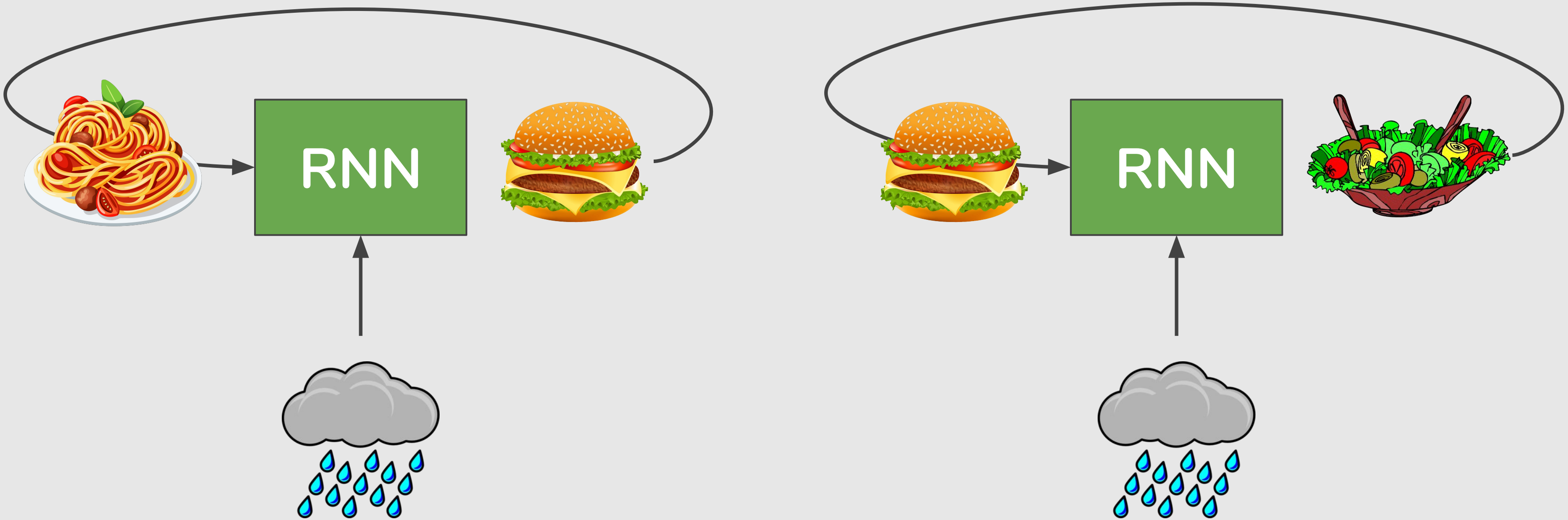
Sunny
Same as yesterday

Rain
Next dish

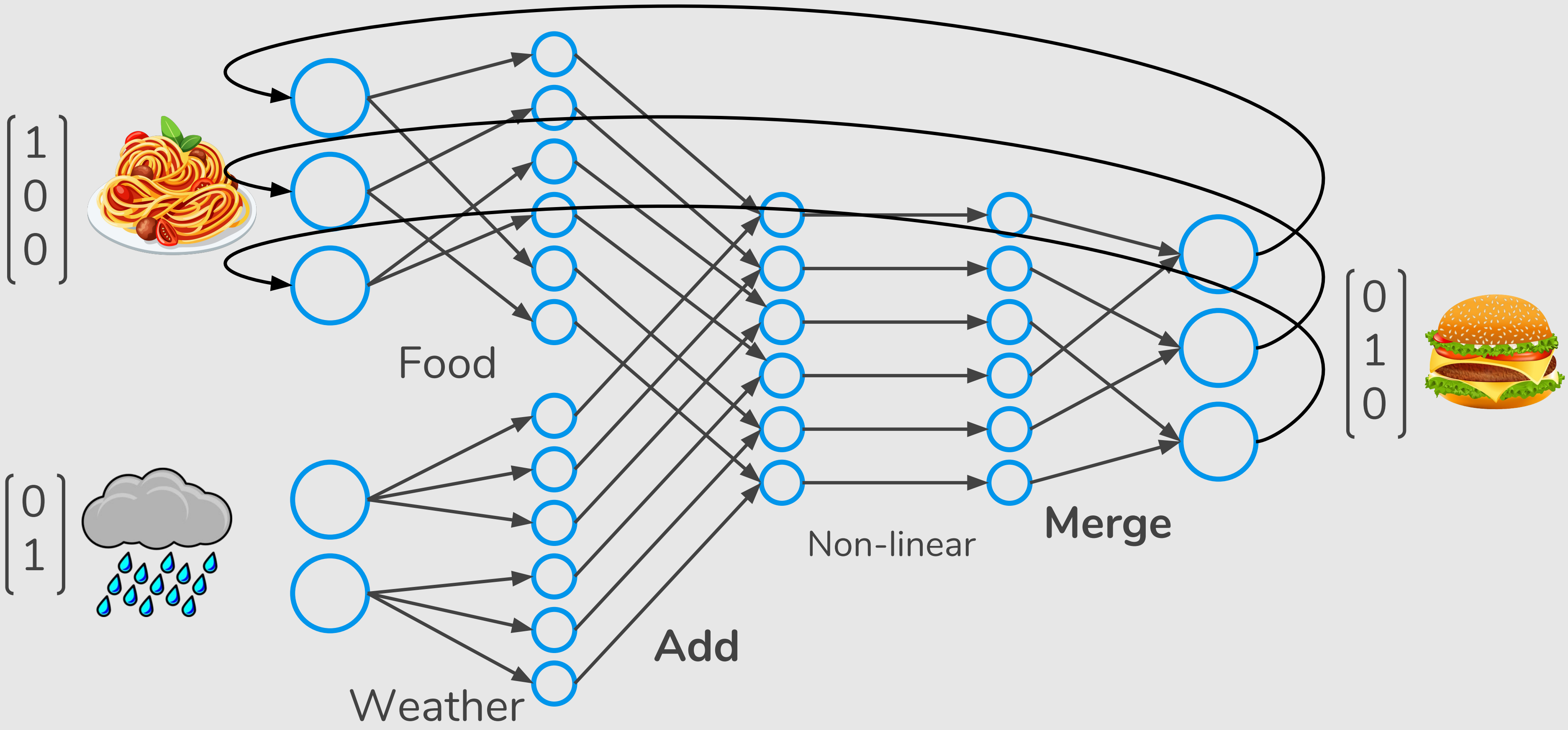

# Cooking Schedule



# Recurrent Neural Network

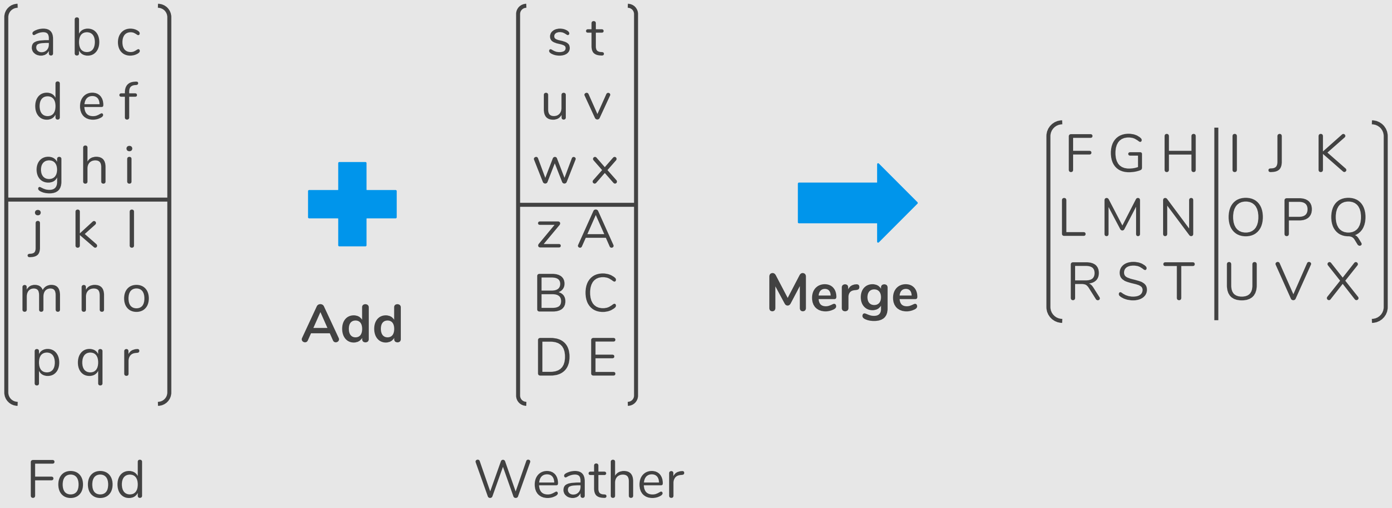


# Recurrent Neural Network



# How to train the RNN?

## Start with Random Weights



# Deep Neural Networks

## Machine Learning and Pattern Recognition

(Largely based on slides from Luis Serrano & Fei-Fei Li & Andrej Karpathy & Justin Johnson & Serena Yeung)

**Prof. Sandra Avila**

Institute of Computing (IC/Unicamp)

MC886/MO444, October 25, 2018

# Recurrent Neural Networks: Process Sequences

one to one

Vanilla Neural Networks

# Recurrent Neural Networks: Process Sequences

one to one

Vanilla Neural Networks

one to many

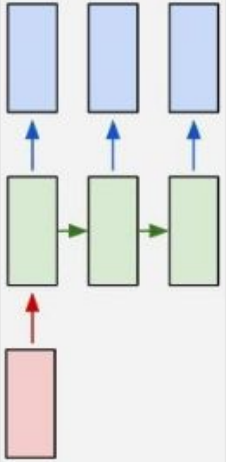


**Image Captioning**
image ⇒ seq. words

# Recurrent Neural Networks: Process Sequences

one to one

Vanilla Neural
Networks

one to many

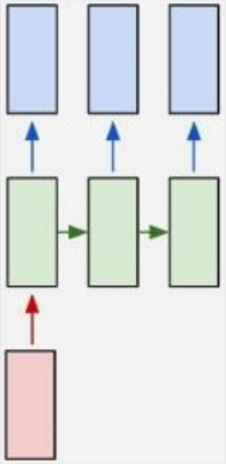
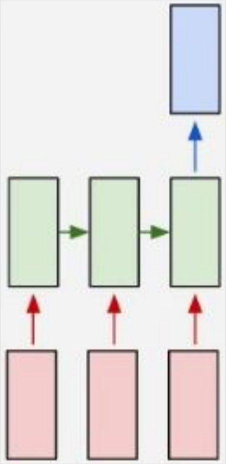


**Image Captioning**
image ⇒ seq. words

many to one



**Sentiment Classification**
seq. words ⇒ sentiment

# Recurrent Neural Networks: Process Sequences

one to one

Vanilla Neural Networks

one to many

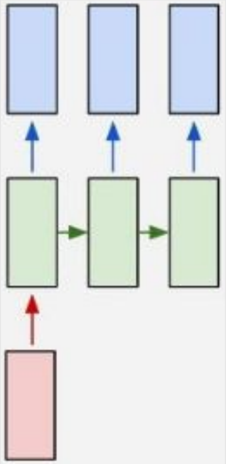
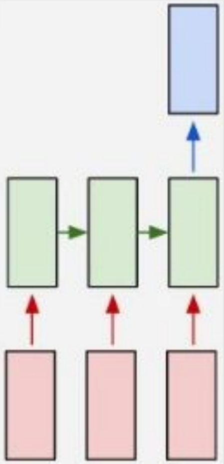


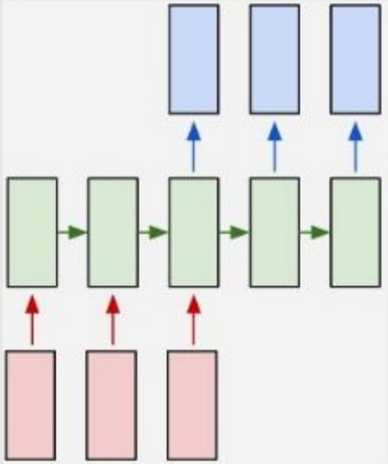
**Image Captioning**
image ⇒ seq. words

many to one



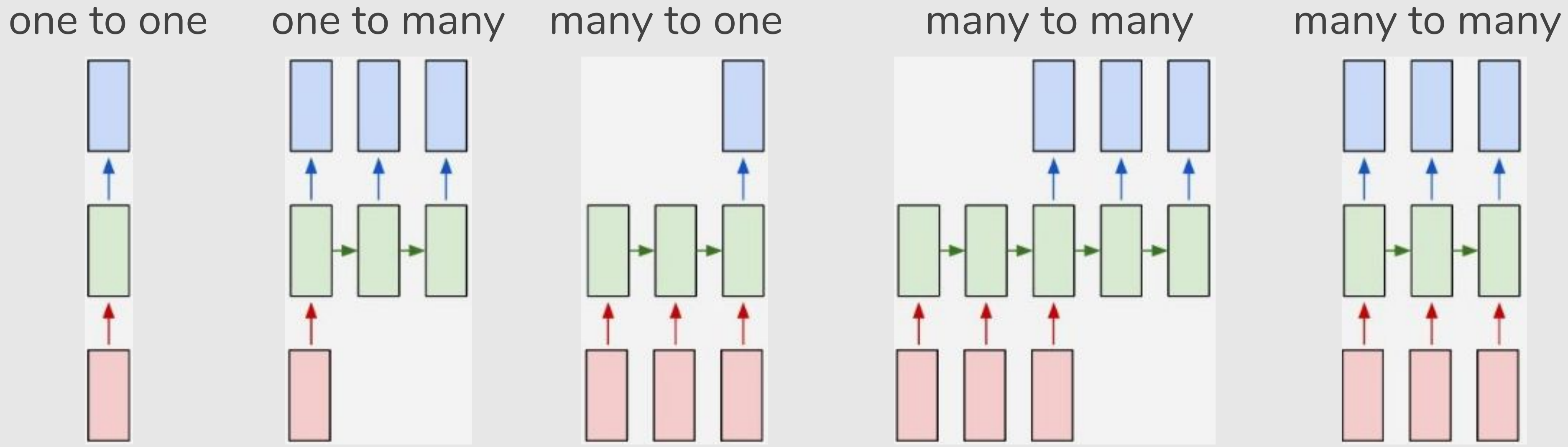
**Sentiment Classification**
seq. words ⇒ sentiment

many to many



**Machine Translation**
seq. words ⇒ seq. of words

# Recurrent Neural Networks: Process Sequences




Vanilla Neural Networks

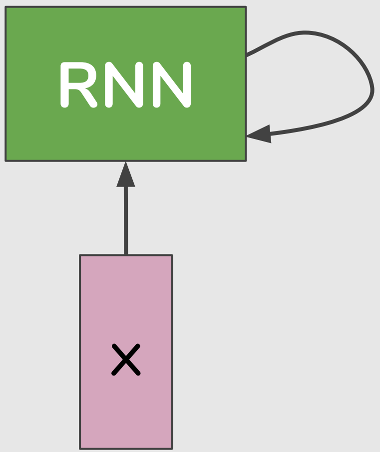
**Image Captioning**
image ⇒ seq. words

**Sentiment Classification**
seq. words ⇒ sentiment

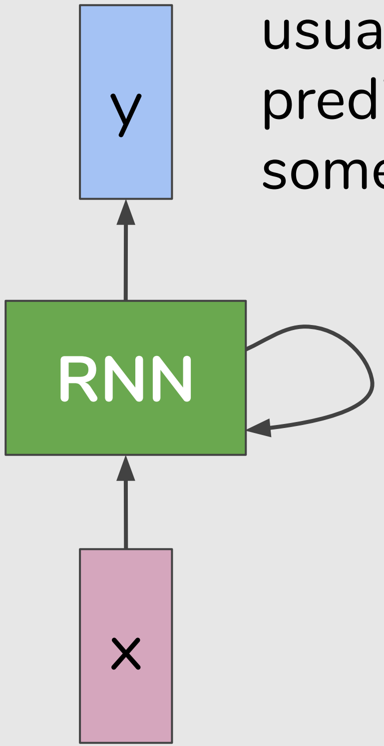
**Machine Translation**
seq. words ⇒ seq. of words

**Video classification on frame level**

# Recurrent Neural Network



# Recurrent Neural Network




usually want to
predict a vector at
some time steps

# Recurrent Neural Network

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

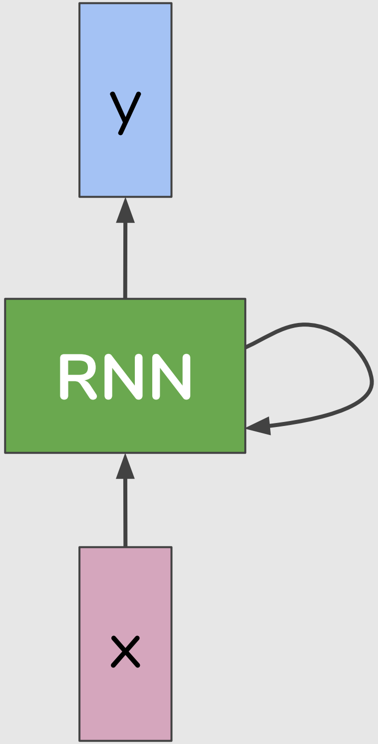
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

input vector at some time step

some function with parameters W

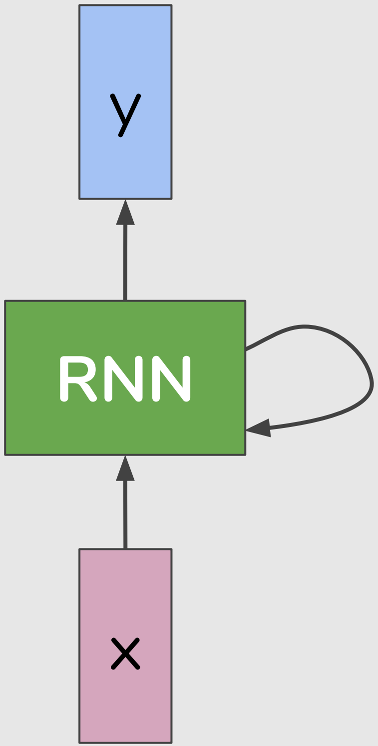


# Recurrent Neural Network

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

**Notice: the same function and the same set of parameters are used at every time step.**



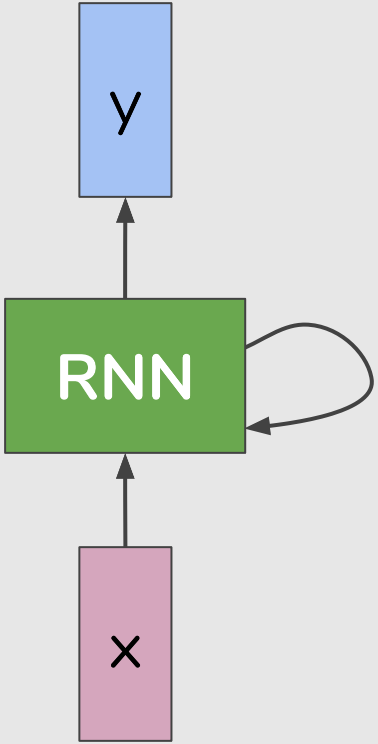
# Recurrent Neural Network

The state consists of a single "hidden" vector **h**:

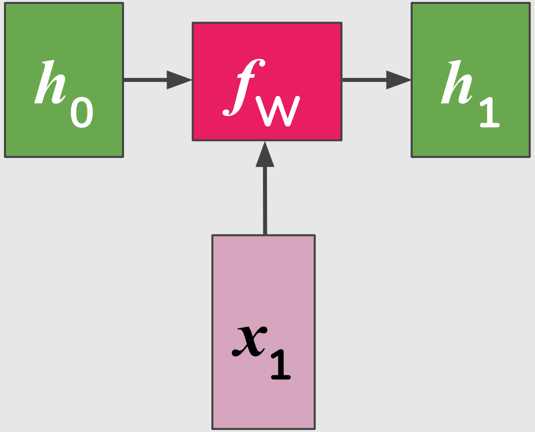
$$h_t = f_W(h_{t-1}, x_t)$$

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

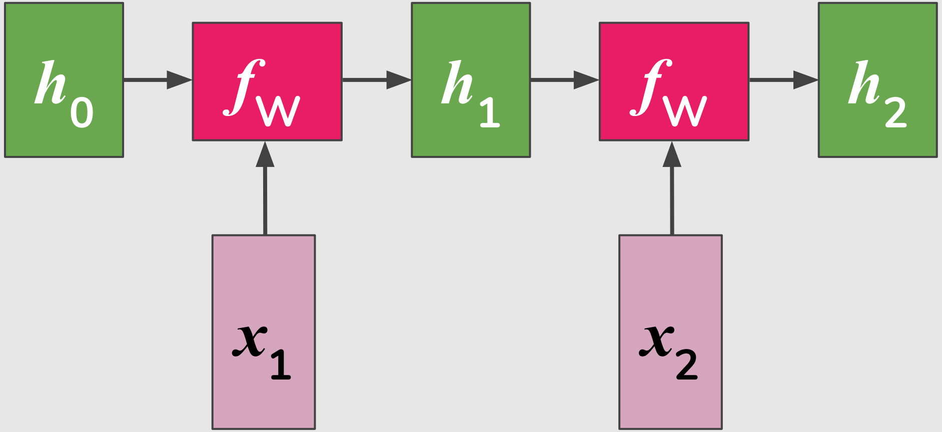
$$y_t = W_{hy}h_t$$



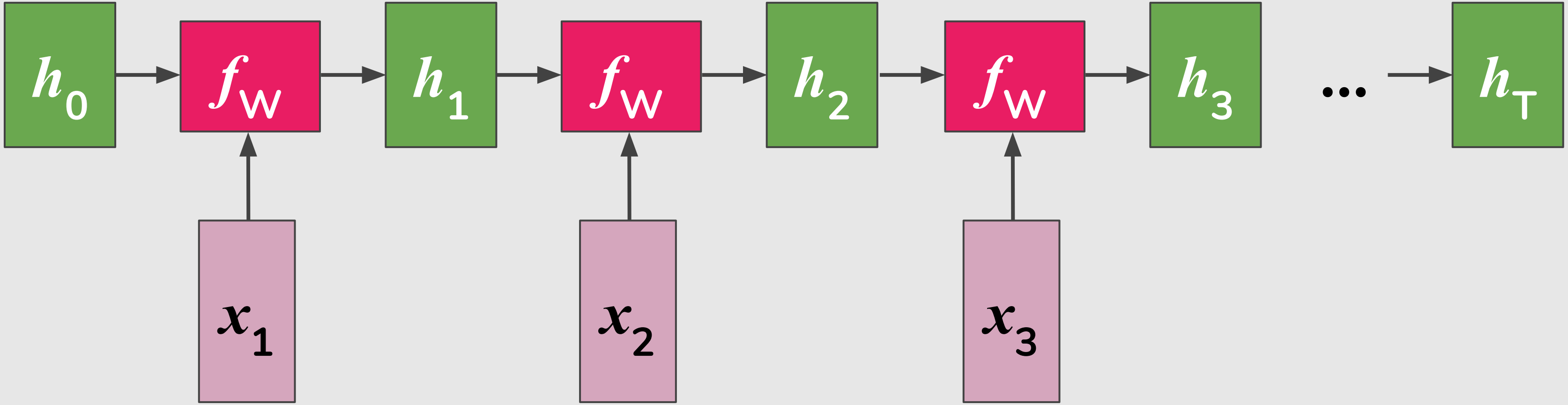
# RNN: Computational Graph



# RNN: Computational Graph

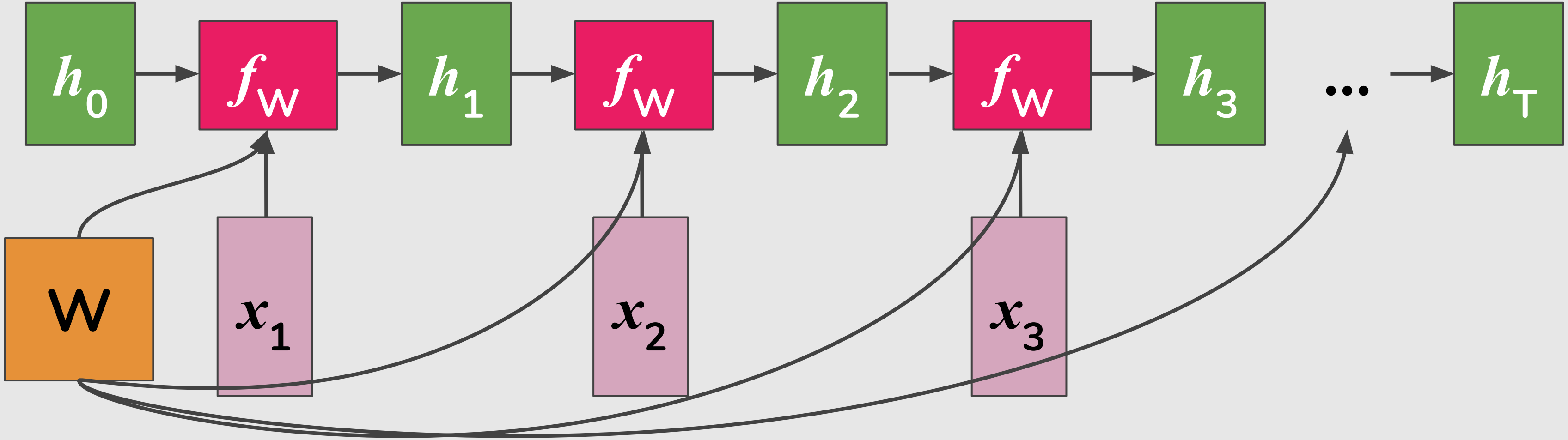


# RNN: Computational Graph

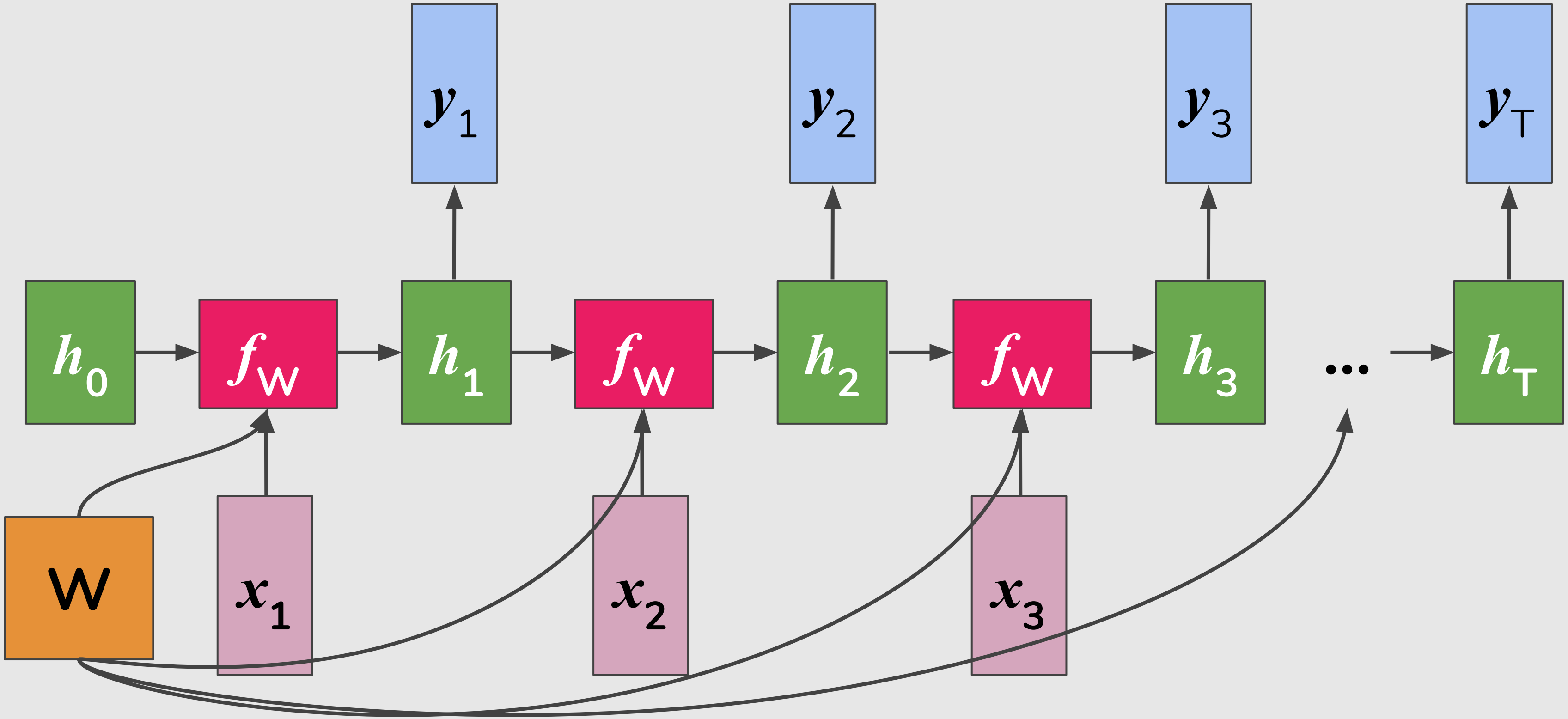


# RNN: Computational Graph

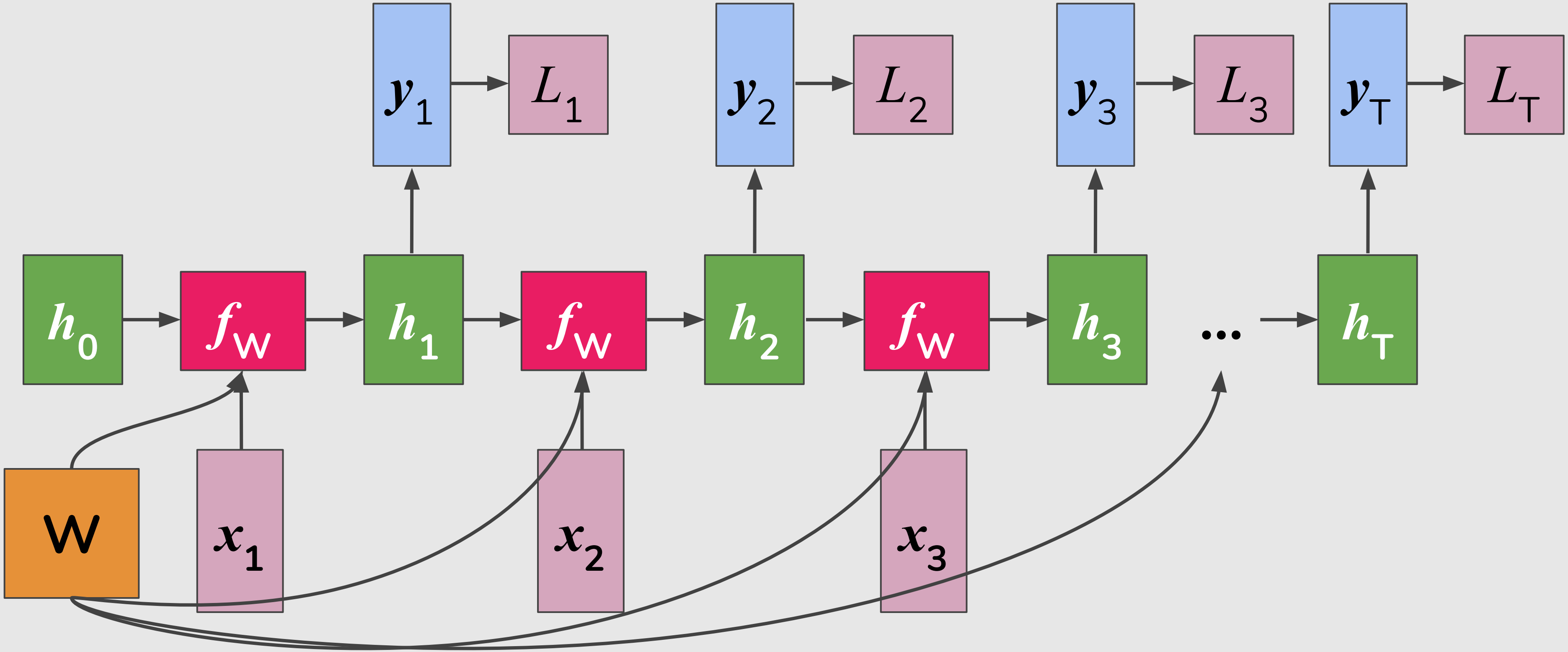
Re-use the same weight matrix at every time-step



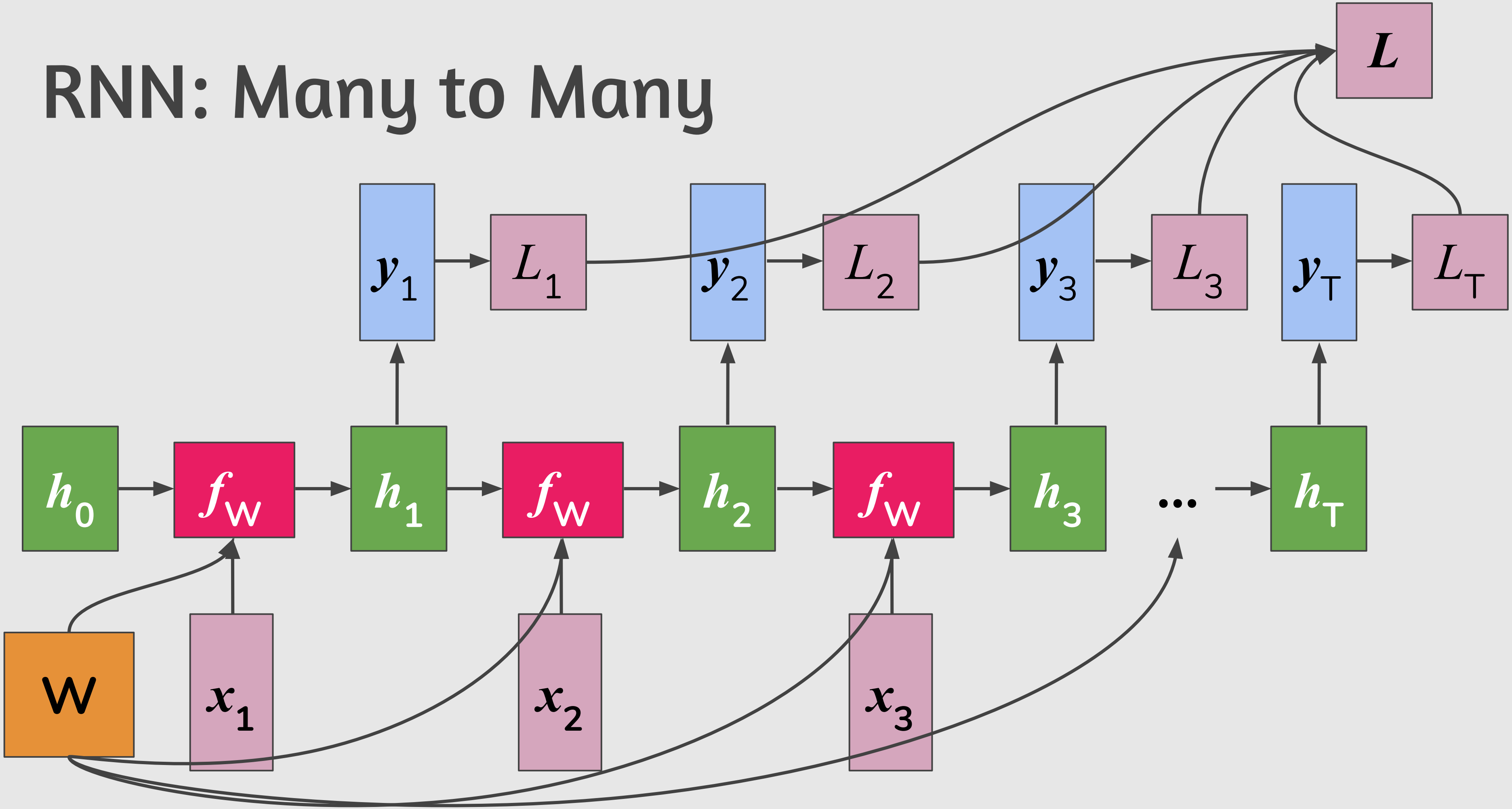
# RNN: Computational Graph



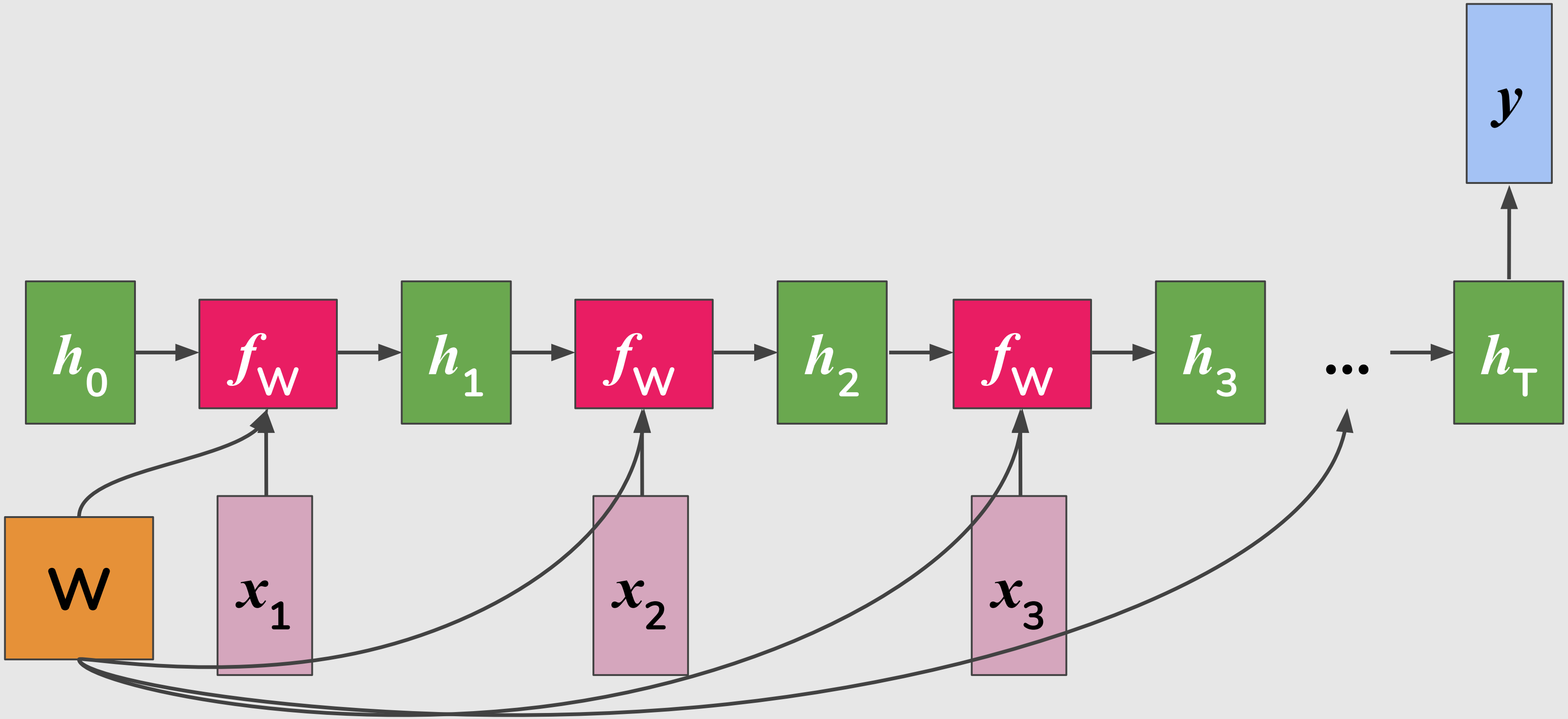
# RNN: Many to Many



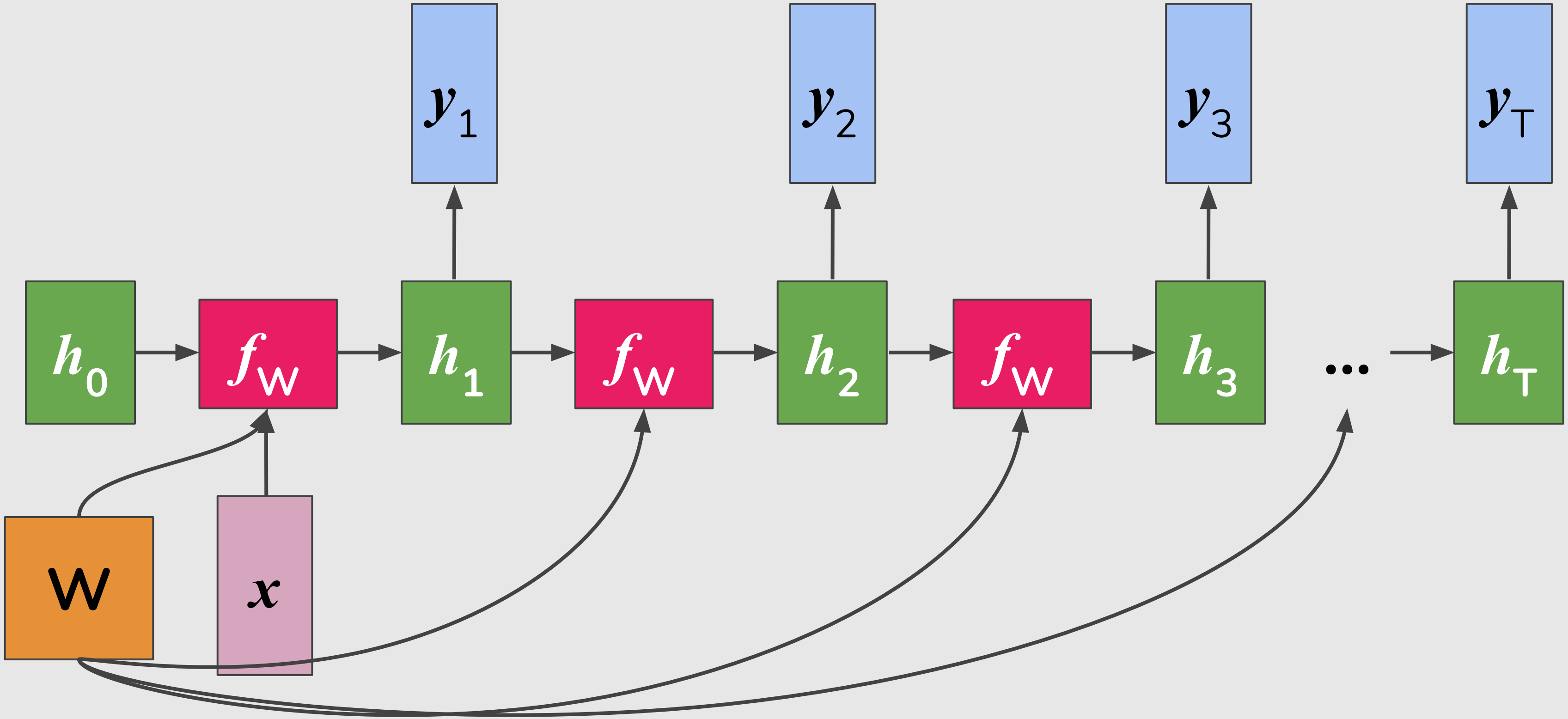
# RNN: Many to Many



# RNN: Many to One



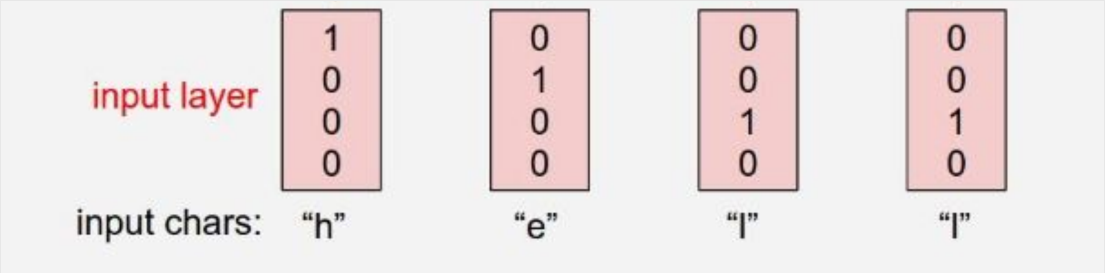
# RNN: One to Many



# Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

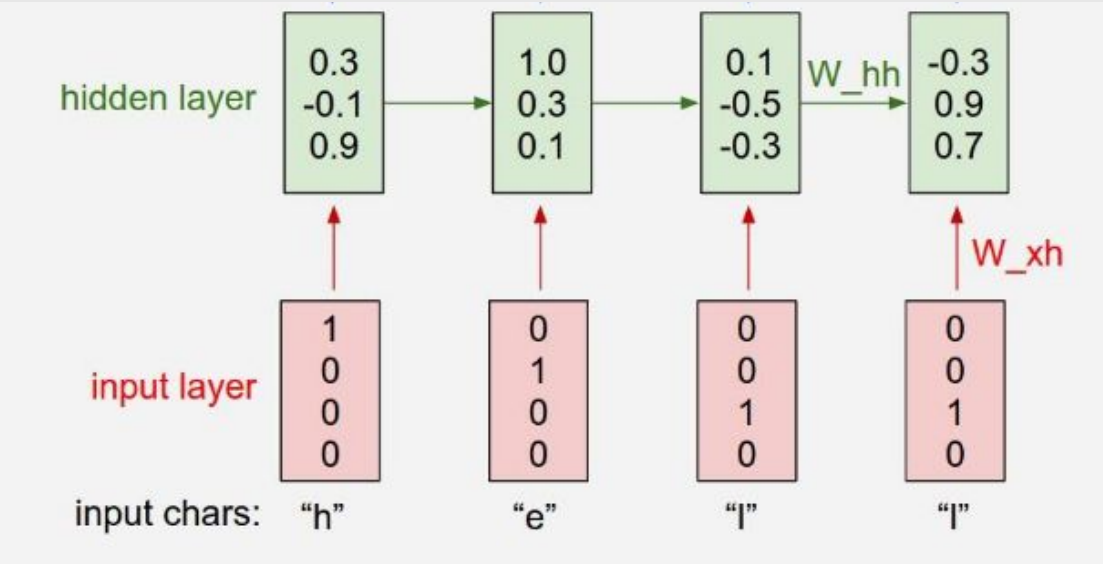


# Character-level Language Model

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

Vocabulary:
[h,e,l,o]

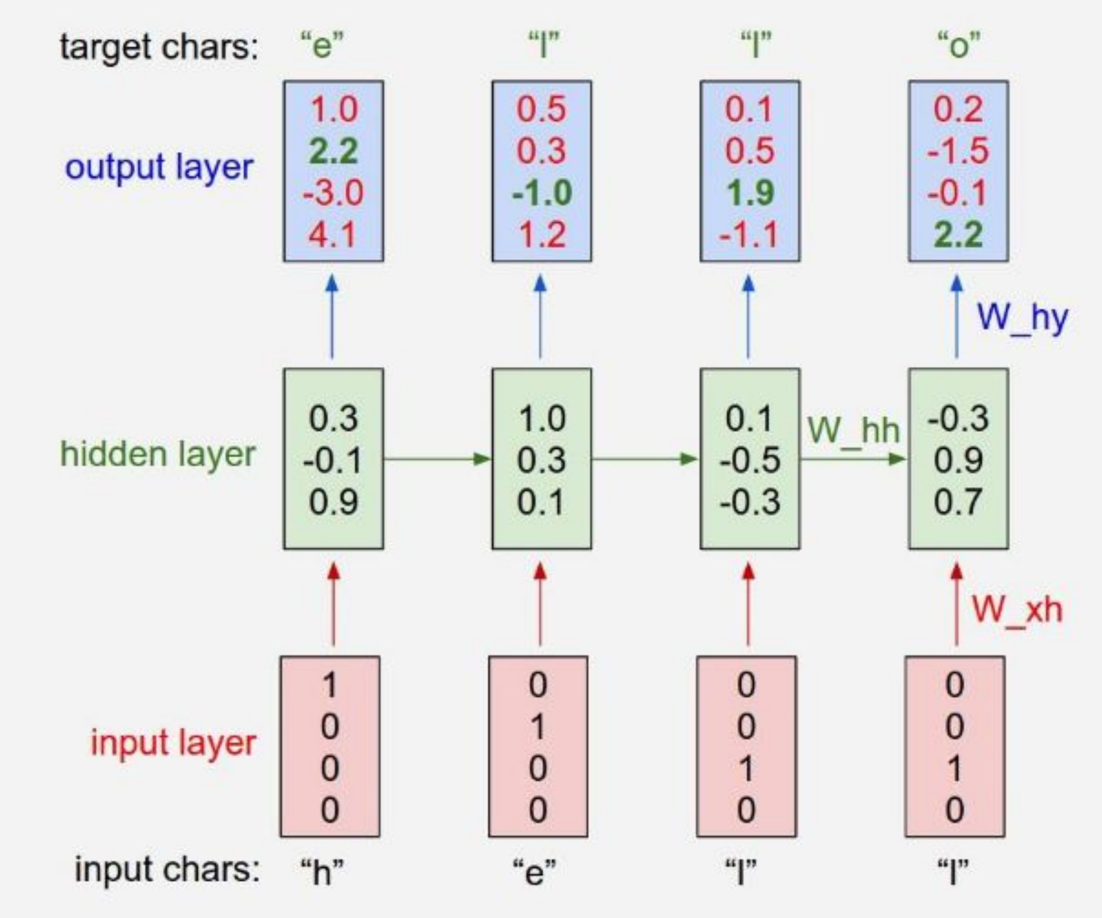
Example training sequence:
“hello”



# Character-level Language Model

Vocabulary:
[h,e,l,o]

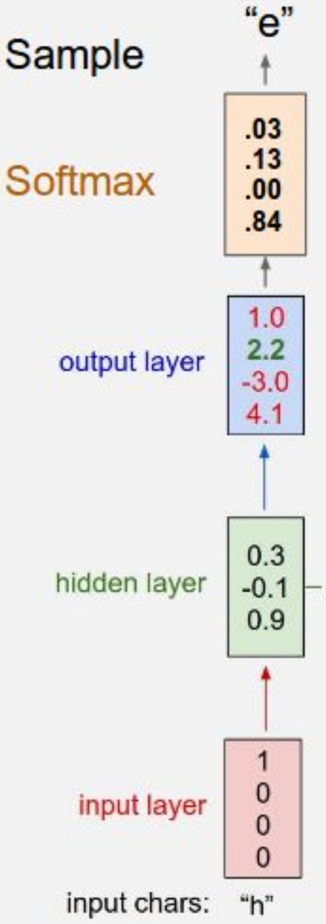
Example training sequence:
“hello”



# Character-level Language Model

Vocabulary:
[h,e,l,o]

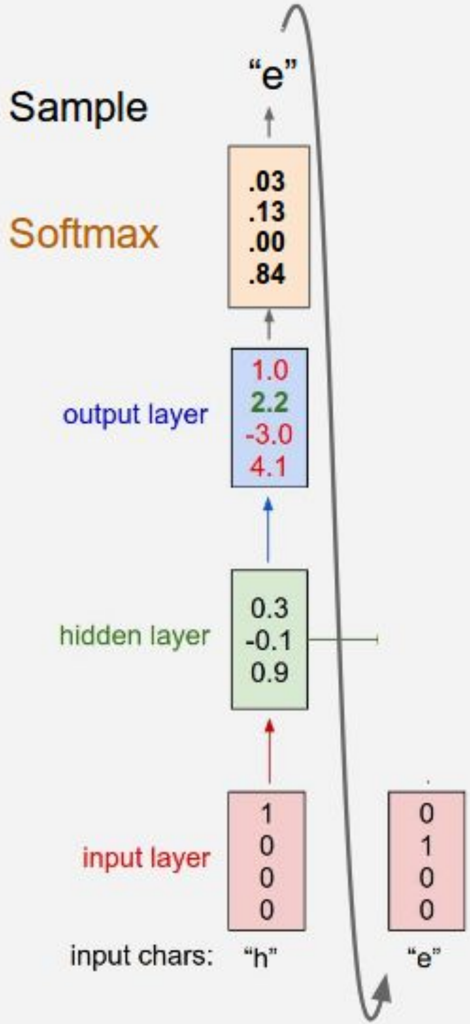
At test-time sample characters one at a time, feed back to model



# Character-level Language Model

Vocabulary:
[h,e,l,o]

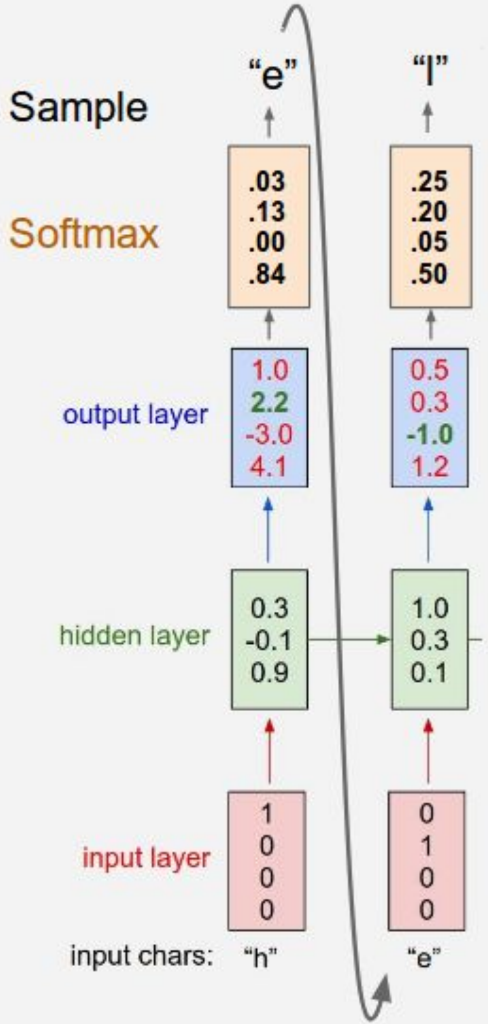
At test-time sample characters one at a time, feed back to model



# Character-level Language Model

Vocabulary:
[h,e,l,o]

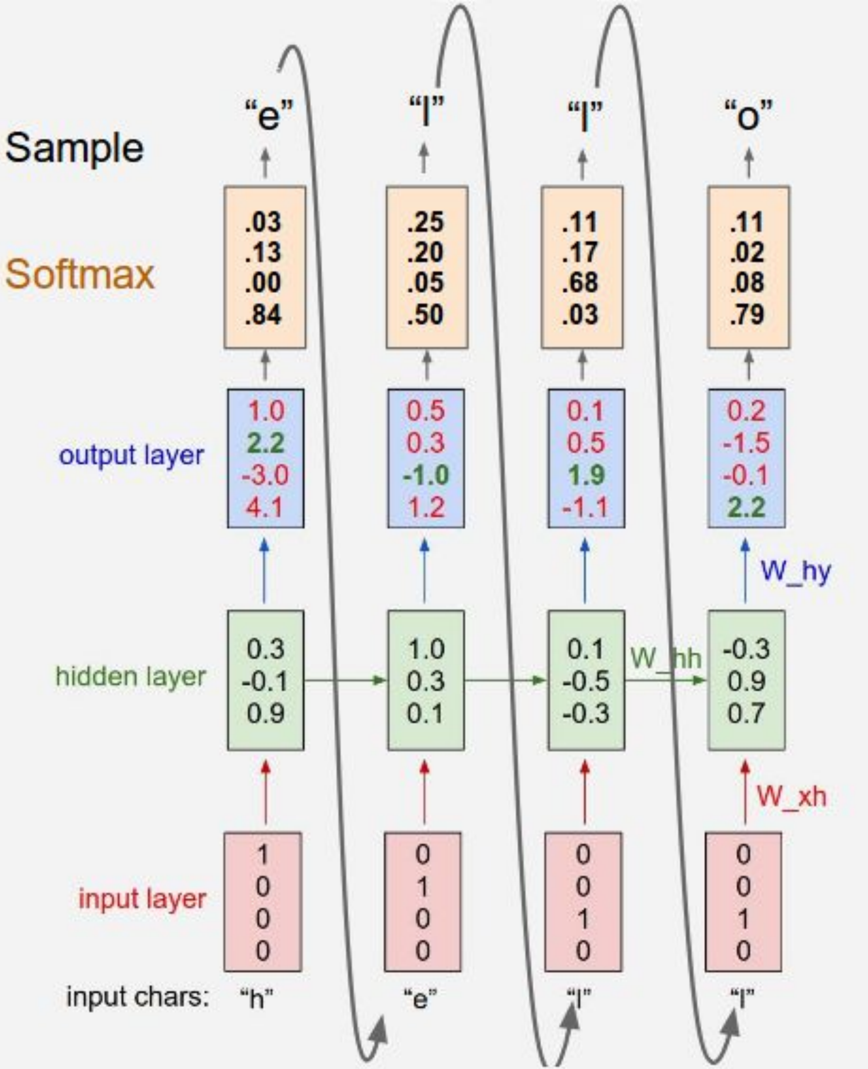
At test-time sample characters one at a time, feed back to model

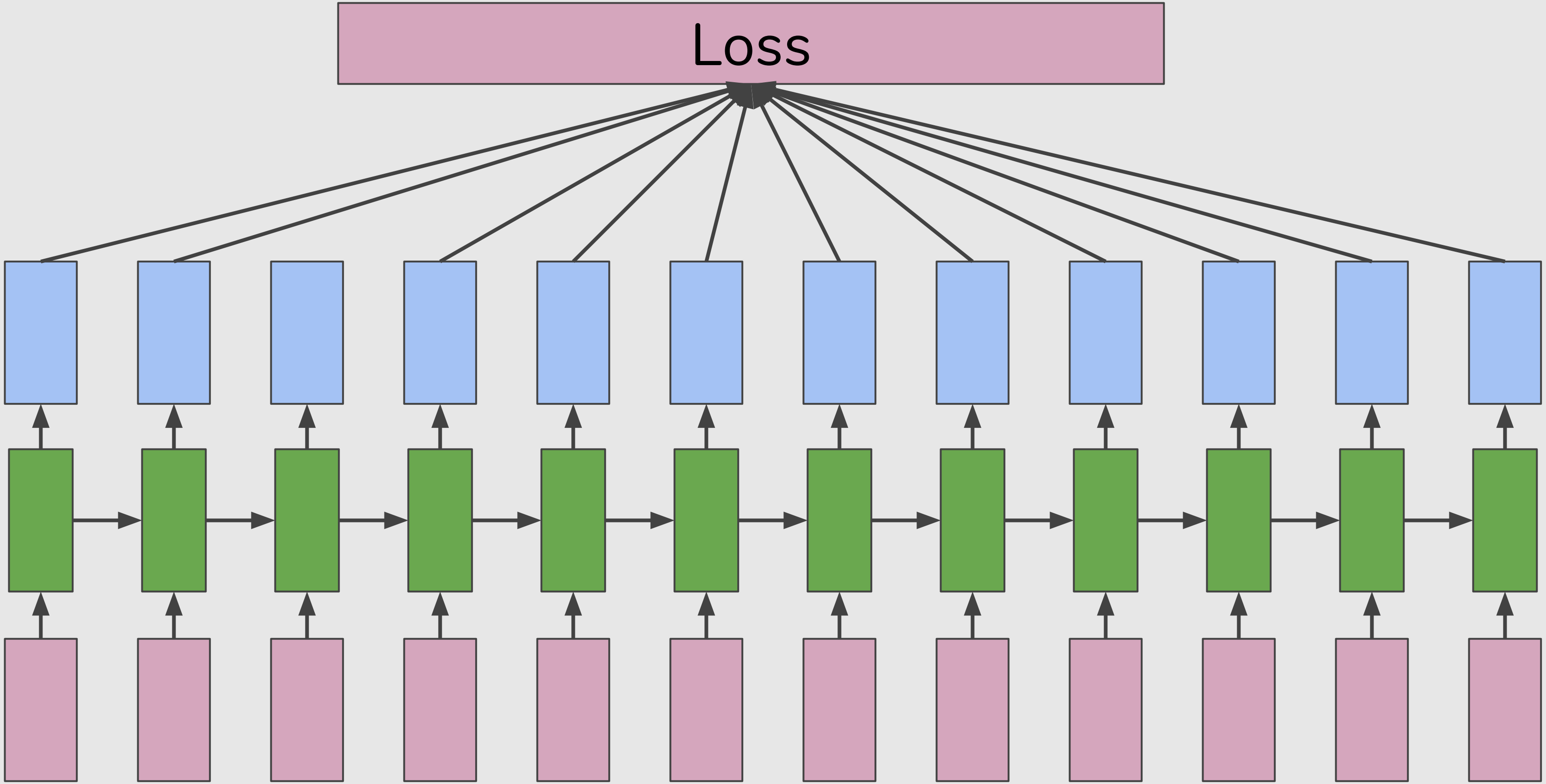


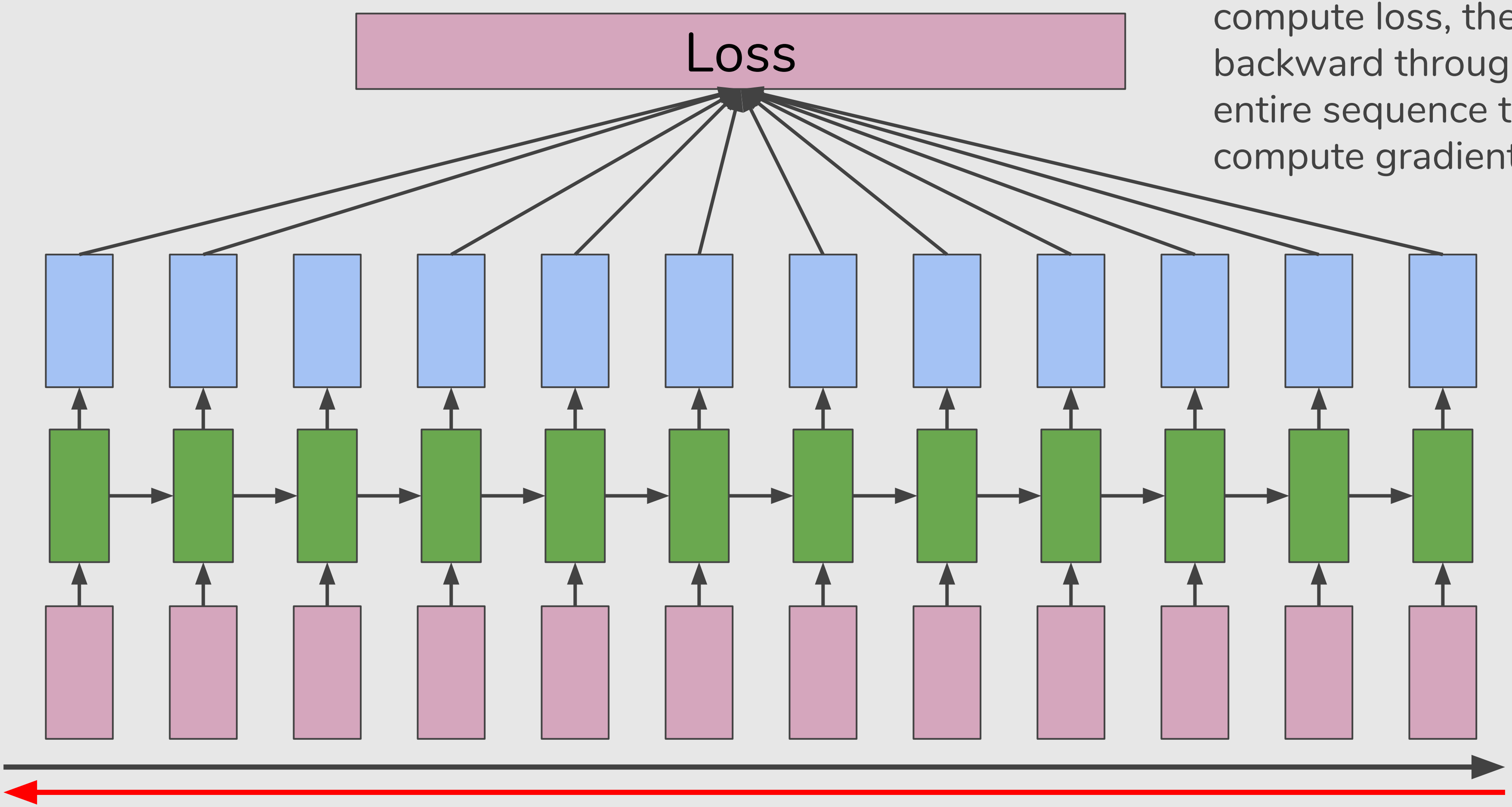
# Character-level Language Model

Vocabulary:
[h,e,l,o]

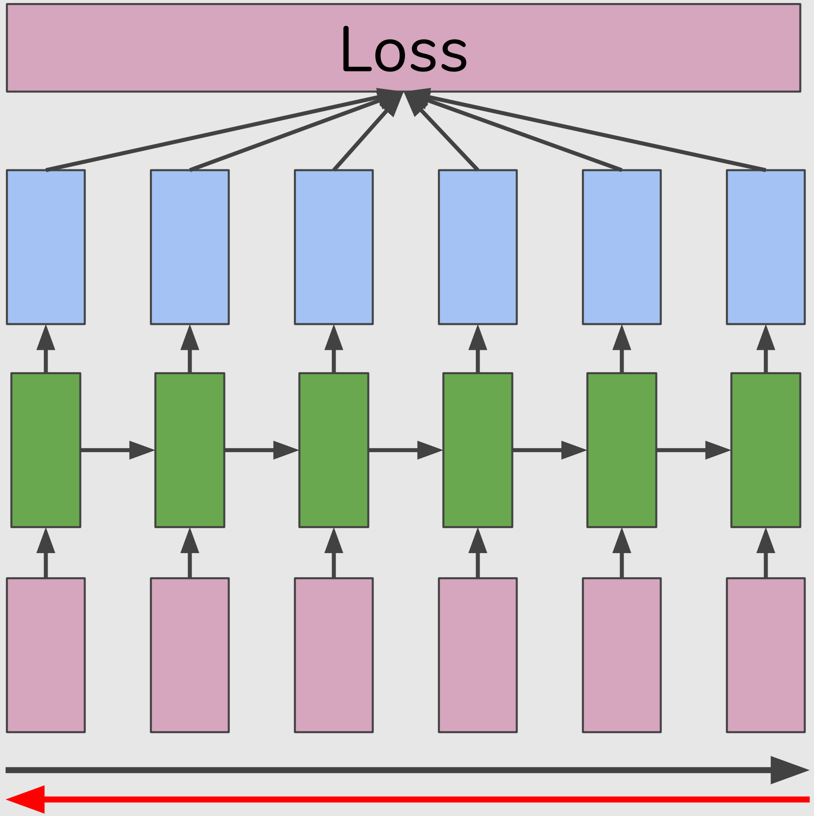
At test-time sample characters one at a time, feed back to model




Loss



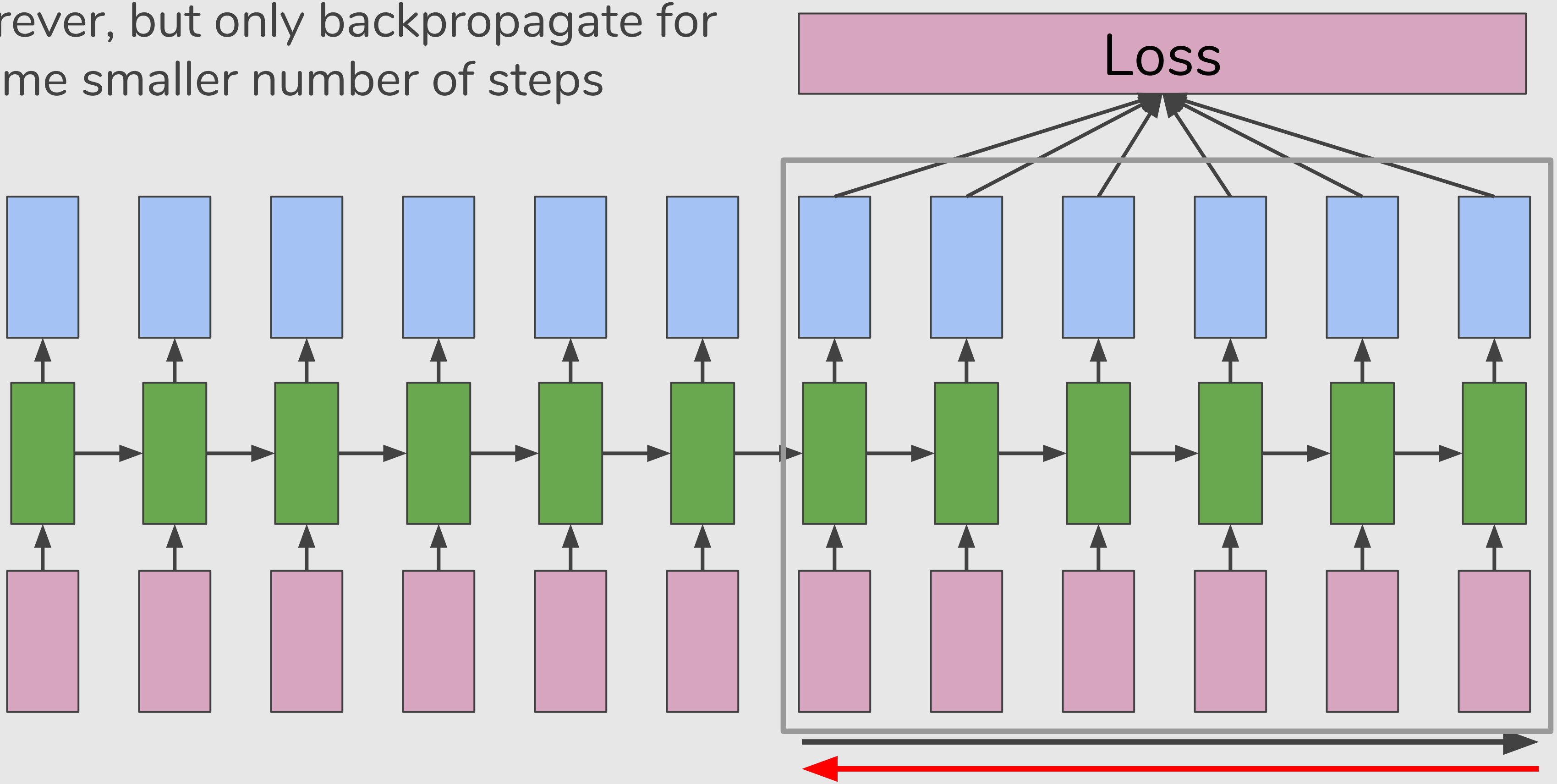

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient.



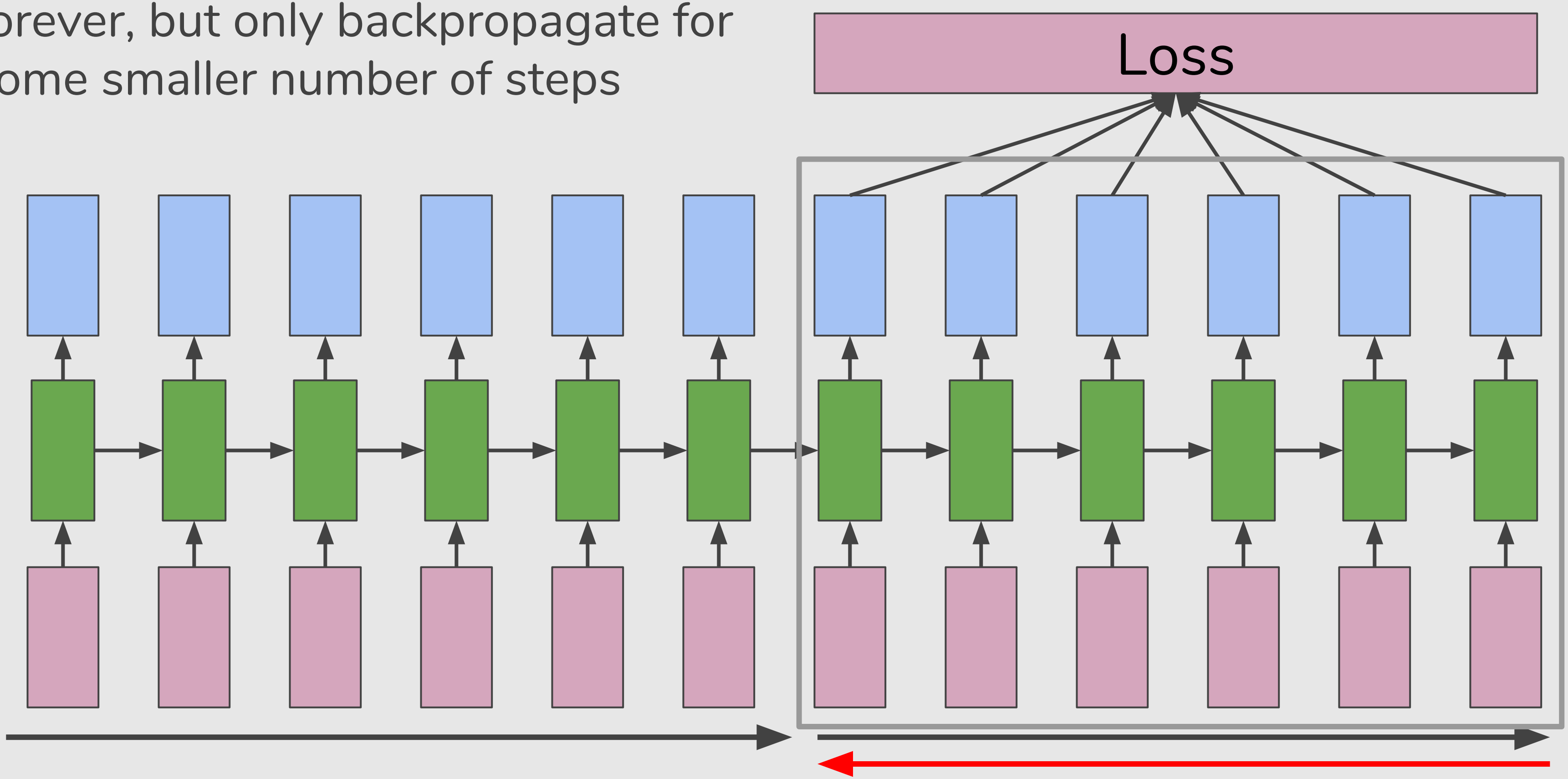

**Truncated** backpropagation through time

Run forward and backward through chunks of the sequence instead of whole sequence

Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps



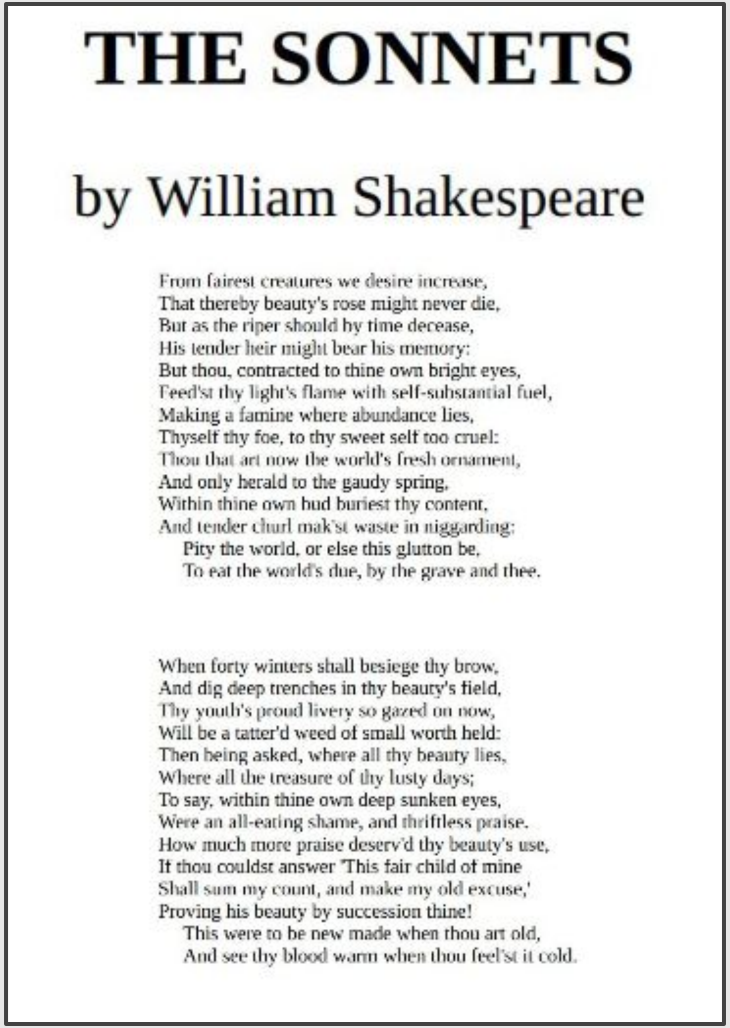
Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps



https://gist.github.com/karpathy/d4dee566867f8291f086

## min-char-rnn.py 112 lines of Python

```python
"""
Minimal character-level Vanilla RNN model. Written by Andrej Karpathy (@karpathy)
BSD License
"""
import numpy as np

# data I/O
data = open('input.txt', 'r').read() # should be simple plain text file
chars = list(set(data))
data_size, vocab_size = len(data), len(chars)
print 'data has %d characters, %d unique.' % (data_size, vocab_size)
char_to_ix = { ch:i for i,ch in enumerate(chars) }
ix_to_char = { i:ch for i,ch in enumerate(chars) }

# hyperparameters
hidden_size = 100 # size of hidden layer of neurons
seq_length = 25 # number of steps to unroll the RNN for
learning_rate = 1e-1

# model parameters
Wxh = np.random.randn(hidden_size, vocab_size)*0.01 # input to hidden
Whh = np.random.randn(hidden_size, hidden_size)*0.01 # hidden to hidden
Why = np.random.randn(vocab_size, hidden_size)*0.01 # hidden to output
bh = np.zeros((hidden_size, 1)) # hidden bias
by = np.zeros((vocab_size, 1)) # output bias

def lossFun(inputs, targets, hprev):
  """
  inputs,targets are both list of integers.
  hprev is Hx1 array of initial hidden state
  returns the loss, gradients on model parameters, and last hidden state
  """
  xs, hs, ys, ps = {}, {}, {}, {}
  hs[-1] = np.copy(hprev)
  loss = 0
  # forward pass
  for t in xrange(len(inputs)):
    xs[t] = np.zeros((vocab_size,1)) # encode in 1-of-k representation
    xs[t][inputs[t]] = 1
    hs[t] = np.tanh(np.dot(Wxh, xs[t]) + np.dot(Whh, hs[t-1]) + bh) # hidden state
    ys[t] = np.dot(Why, hs[t]) + by # unnormalized log probabilities for next chars
    ps[t] = np.exp(ys[t]) / np.sum(np.exp(ys[t])) # probabilities for next chars
    loss += -np.log(ps[t][targets[t],0]) # softmax (cross-entropy loss)
  # backward pass: compute gradients going backwards
  dWxh, dWhh, dWhy = np.zeros_like(Wxh), np.zeros_like(Whh), np.zeros_like(Why)
  dbh, dby = np.zeros_like(bh), np.zeros_like(by)
  dhnext = np.zeros_like(hs[0])
  for t in reversed(xrange(len(inputs))):
    dy = np.copy(ps[t])
    dy[targets[t]] -= 1 # backprop into y
    dWhy += np.dot(dy, hs[t].T)
    dby += dy
    dh = np.dot(Why.T, dy) + dhnext # backprop into h
    dhraw = (1 - hs[t] * hs[t]) * dh # backprop through tanh nonlinearity
    dbh += dhraw
    dWxh += np.dot(dhraw, xs[t].T)
    dWhh += np.dot(dhraw, hs[t-1].T)
    dhnext = np.dot(Whh.T, dhraw)
  for dparam in [dWxh, dWhh, dWhy, dbh, dby]:
    np.clip(dparam, -5, 5, out=dparam) # clip to mitigate exploding gradients
  return loss, dWxh, dWhh, dWhy, dbh, dby, hs[len(inputs)-1]

def sample(h, seed_ix, n):
  """
  sample a sequence of integers from the model
  h is memory state, seed_ix is seed letter for first time step
  """
  x = np.zeros((vocab_size, 1))
  x[seed_ix] = 1
  ixes = []
  for t in xrange(n):
    h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h) + bh)
    y = np.dot(Why, h) + by
    p = np.exp(y) / np.sum(np.exp(y))
    ix = np.random.choice(range(vocab_size), p=p.ravel())
    x = np.zeros((vocab_size, 1))
    x[ix] = 1
    ixes.append(ix)
  return ixes

n, p = 0, 0
mWxh, mWhh, mWhy = np.zeros_like(Wxh), np.zeros_like(Whh), np.zeros_like(Why)
mbh, mby = np.zeros_like(bh), np.zeros_like(by) # memory variables for Adagrad
smooth_loss = -np.log(1.0/vocab_size)*seq_length # loss at iteration 0
while True:
  # prepare inputs (we're sweeping from left to right in steps seq_length long)
  if p+seq_length+1 >= len(data) or n == 0:
    hprev = np.zeros((hidden_size,1)) # reset RNN memory
    p = 0 # go from start of data
  inputs = [char_to_ix[ch] for ch in data[p:p+seq_length]]
  targets = [char_to_ix[ch] for ch in data[p+1:p+seq_length+1]]

  # sample from the model now and then
  if n % 100 == 0:
    sample_ix = sample(hprev, inputs[0], 200)
    txt = ''.join(ix_to_char[ix] for ix in sample_ix)
    print '----\n %s \n----' % (txt, )

  # forward seq_length characters through the net and fetch gradient
  loss, dWxh, dWhh, dWhy, dbh, dby, hprev = lossFun(inputs, targets, hprev)
  smooth_loss = smooth_loss * 0.999 + loss * 0.001
  if n % 100 == 0: print 'iter %d, loss: %f' % (n, smooth_loss) # print progress

  # perform parameter update with Adagrad
  for param, dparam, mem in zip([Wxh, Whh, Why, bh, by],
                                [dWxh, dWhh, dWhy, dbh, dby],
                                [mWxh, mWhh, mWhy, mbh, mby]):
    mem += dparam * dparam
    param += -learning_rate * dparam / np.sqrt(mem + 1e-8) # adagrad update

  p += seq_length # move data pointer
  n += 1 # iteration counter
```

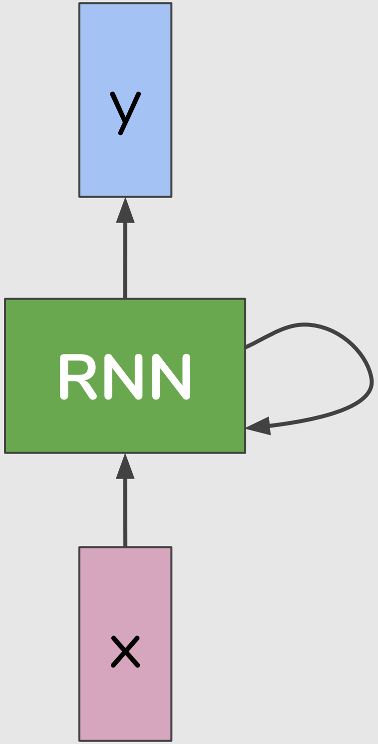


# THE SONNETS

## by William Shakespeare

From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the riper should by time decease,
His tender heir might bear his memory:
But thou, contracted to thine own bright eyes,
Feed'st thy light's flame with self-substantial fuel,
Making a famine where abundance lies,
Thyself thy foe, to thy sweet self too cruel:
Thou that art now the world's fresh ornament,
And only herald to the gaudy spring,
Within thine own bud buriest thy content,
And tender churl mak'st waste in niggarding:
  Pity the world, or else this glutton be,
  To eat the world's due, by the grave and thee.

When forty winters shall besiege thy brow,
And dig deep trenches in thy beauty's field,
Thy youth's proud livery so gazed on now,
Will be a tatter'd weed of small worth held:
Then being asked, where all thy beauty lies,
Where all the treasure of thy lusty days;
To say, within thine own deep sunken eyes,
Were an all-eating shame, and thriftless praise.
How much more praise deserv'd thy beauty's use,
If thou couldst answer 'This fair child of mine
Shall sum my count, and make my old excuse,'
Proving his beauty by succession thine!
  This were to be new made when thou art old,
  And see thy blood warm when thou feel'st it cold.



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.
```

PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

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.

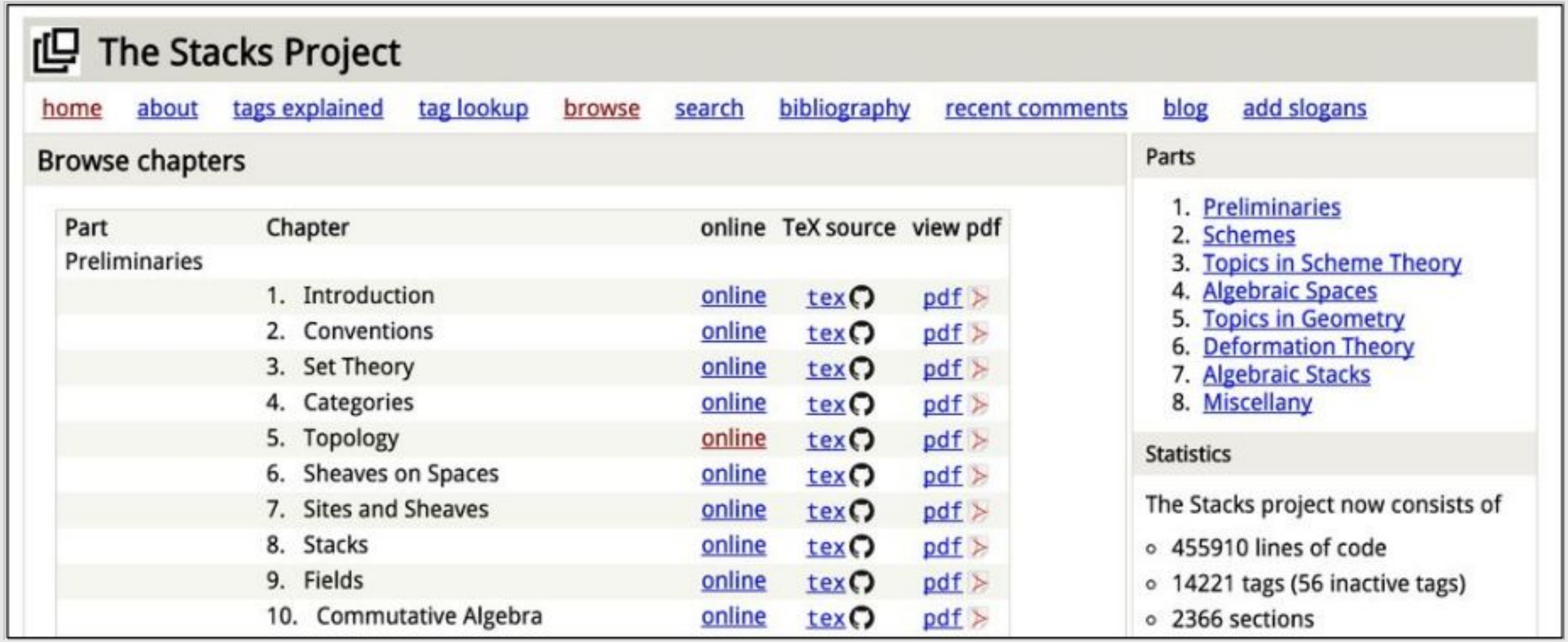
Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

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

VIOLA:
I'll drink it.

VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.


# The Stacks Project

home about tags explained tag lookup browse search bibliography recent comments blog add slogans

## Browse chapters

| Part | Chapter | online | TeX source | view pdf |
|---|---|---|---|---|
| Preliminaries | | | | |
| | 1. Introduction | online | tex | pdf |
| | 2. Conventions | online | tex | pdf |
| | 3. Set Theory | online | tex | pdf |
| | 4. Categories | online | tex | pdf |
| | 5. Topology | online | tex | pdf |
| | 6. Sheaves on Spaces | online | tex | pdf |
| | 7. Sites and Sheaves | online | tex | pdf |
| | 8. Stacks | online | tex | pdf |
| | 9. Fields | online | tex | pdf |
| | 10. Commutative Algebra | online | tex | pdf |

### Parts

1. Preliminaries
2. Schemes
3. Topics in Scheme Theory
4. Algebraic Spaces
5. Topics in Geometry
6. Deformation Theory
7. Algebraic Stacks
8. Miscellany

### Statistics

The Stacks project now consists of

- 455910 lines of code
- 14221 tags (56 inactive tags)
- 2366 sections

Latex source

For $\bigoplus_{n=1,\dots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on $X$, $U$ is a closed immersion of $S$, then $U \to T$ is a separated algebraic space.

*Proof.* Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps $M$ along the set of points $Sch_{fppf}$ and $U \to U$ is the fibre category of $S$ in $U$ in Section, ?? and the fact that any $U$ affine, see Morphisms, Lemma ??. Hence we obtain a scheme $S$ and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that $f_i$ is of finite presentation over $S$. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of $X$. Thus $U$ is affine. This is a continuous map of $X$ is the inverse, the groupoid scheme $S$.

*Proof.* See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace $S$ by $X_{spaces,étale}$ which gives an open subspace of $X$ and $T$ equal to $S_{Zar}$, see Descent, Lemma ??. Namely, by Lemma ?? we see that $R$ is geometrically regular over $S$.

**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 $U \cap U = \coprod_{i=1,\dots,n} U_i$ 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 $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{X,\dots,0}$.

**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 $\delta_{n+1}$ is a scheme over $S$. □

*Proof.* Omitted. □

**Lemma 0.1.** *Let $\mathcal{C}$ be a set of the construction.*

*Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that*

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

*Proof.* This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

**Lemma 0.2.** *This is an integer $\mathcal{Z}$ is injective.*

*Proof.* See Spaces, Lemma ??. □

**Lemma 0.3.** *Let $S$ be a scheme. Let $X$ be a scheme and $X$ is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let $X$ be a scheme. Let $X$ be a scheme which is equal to the formal complex.*

*The following to the construction of the lemma follows.*

*Let $X$ be a scheme. Let $X$ be a scheme covering. Let*

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

*be a morphism of algebraic spaces over $S$ and $Y$.*

*Proof.* Let $X$ be a nonzero scheme of $X$. Let $X$ be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over $S$.
(2) If $X$ is an affine open covering.

Consider a common structure on $X$ and $X$ the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram

$$\begin{tikzcd}
\mathcal{S} \arrow[r] \arrow[d] & {} & & & \\
\xi \arrow[r] & \mathcal{O}_{X'} \arrow[dr] & & & \\
\mathrm{gor}_* & {} \arrow[u] & {} & & \\
& =\alpha' \arrow[r] \arrow[d, leftrightarrow] & {} & & \\
& =\alpha' \arrow[r] & \alpha & & X \arrow[d] \\
& \mathrm{Spec}(K_\psi) & & \mathrm{Mor}_{Sets} & \mathrm{d}(\mathcal{O}_{X_{X/k}}, \mathcal{G})
\end{tikzcd}$$

is a limit. Then $\mathcal{G}$ is a finite type and assume $S$ is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type $f_*$. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

*Proof.* We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of $X$ is an open neighbourhood of $U$. □

*Proof.* This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that $U$ is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad -1(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that $X$ is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over $S$.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

*Proof.* Omitted. □

**Lemma 0.1.** *Let $\mathcal{C}$ be a set of the construction.*

*Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that*

.

*Proof.* Omitted.

*Proof.* This is an algebra have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

**Lemma 0.2.** *This is an integer $\mathcal{Z}$ is injective.*

*Proof.* See Spaces, Lemma ??. □

**Lemma 0.3.** *Let $S$ be a scheme. Let $X$ be a scheme and $X$ is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let $X$ be a scheme. Let $X$ be a scheme which is equal to the formal complex.*

*The following to the construction of the lemma follows.*

*Let $X$ be a scheme. Let $X$ be a scheme covering. Let*

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

*be a morphism of algebraic spaces over $S$ and $Y$.*

*Proof.* Let $X$ be a nonzero scheme of $X$. Let $X$ be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over $S$.
(2) If $X$ is an affine open covering.

Consider a common structure on $X$ and $X$ the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram

$$\begin{tikzcd} S \arrow[r] \arrow[d] & {} & \\ \xi \arrow[r] & \mathcal{O}_{X'} \arrow[dr] & \\ \mathrm{gor}_* & {} \arrow[u] & {} \\ & =\alpha' \arrow[r] \arrow[d, leftrightarrow] & {} \\ & =\alpha' \arrow[r] & \alpha & X \arrow[d] \\ & \mathrm{Spec}(K_\psi) & \mathrm{Mor}_{Sets} & d(\mathcal{O}_{X_{x/k}}, \mathcal{G}) \end{tikzcd}$$

is a limit. Then $\mathcal{G}$ is a finite type and assume $S$ is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type $f_*$. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

*Proof.* We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of $X$ is an open neighbourhood of $U$. □

*Proof.* This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that $U$ is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad -1(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that $X$ is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over $S$.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

```
static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```

Generated code

# Training: “Maior dúvida da aula” 27/october/2017

## https://github.com/llSourcell/recurrent_neural_network

##### GoogLeNet, Inception Module

Não entendi muito bem sobre as inception layers na GoogLeNet. Entendi a ideia de fazer a mesma coisa de um filtro grande com vários filtros menores. Com vários filtros menores temos menos parâmetros que um filtro grande?

Quando fazemos inception e concatenados os resultados, podemos comparar isso à criação de vetor de características? Porque estamos retirando tipos diferentes de informações de uma mesma camada de input e juntando elas pra formar um output.

Acho que não consegui entender muito bem o inception module da arquitetura GoogLeNet. Para que ele serve exatamente? Obrigada.

no modelo de inception v4,  usa a paralelizacao para obter menos parametros, entao esso quer dizer que enquanto menos parametros e mais profundo da melhores resultados?

Não entendi exatamente que fator possibilitou a remoção das camadas fully connected na GoogleLeNet. Pelo que eu entendi, as redes mais modernas voltaram com a camada fully connected. Então quando usá-la ou não usá-la?

## Números de parâmetros

Em relação a arquiterua proposta na rede GoogLeNet, não ficou muito claro para mim as camadas internas, principalmente na parte em que aplicar vários filtros menores, equilave a aplicar um filtro maior (embora o resultado não seja o mesmo).

Não ficou claro para mim qual a vantagem de se utilizar, por exemplo, 3 pequenos filtros 3x3 ao invés de um 7x7. Na aula você comentou que é para evitar diminuir drasticamente a imagem, mas qual a desvantagem disso?

Eu nao entendi aquelas contas dos filtros que reduziam o numero de parametros

##### ResNet Filtro 1x1

Achei um pouco confuso as dimensões do filtro 1x1. Achei confuso a parte da convolução de tal filtro.

# Training: “Maior dúvida da aula” 27/october/2017

```
iter 0, loss: 107.601633
----
 'ōqIE:ō:3(é
O Q.L"cÉhíL'uàfMO)êoâz.àãâéláç-)D(iéêdàF(lLFLrRcFA0nC(Pô(á#HM5éI?#ázHrtGTRF)5wlGaúa2éj?pd7,u
xp5LQ"r24F7é1efL"Cabvêúhy Ldã 7àã2àObmxv?qnAodí'P)mTg4(u4F7ú13ómrQnmeFNbãoúvâ3i?sxsuRãjáécó.-
záy-
----
```

# Training: “Maior dúvida da aula” 27/october/2017

```
iter 0, loss: 107.601633
----
 'ōqIE:ō:3(é
O Q.L"cÉhíL'uàfMO)êoâz.àãâéláç-)D(iéêdàF(lLFLrRcFA0nC(Pô(á#HM5éI?#ázHrtGTRF)5wlGaúa2éj?pd7,u
xp5LQ"r24F7é1efL"Cabvêúhy Ldã 7àã2àObmxv?qnAodí'P)mTg4(u4F7ú13ómrQnmeFNbãoúvâ3i?sxsuRãjáécó.-
záy
---
```

```
iter 46000, loss: 23.238596
----
 és GoogLeNet. E a rede aprede?

O Daras dúvrvilg. ( ende no pré-tro "rar outlara destidas? Com uttres dessar algo us filtros
parte novados aplicar au mula.

e narepteno Retênne camada entros lemos e m
```

# Training: “Maior dúvida da aula” 27/october/2017

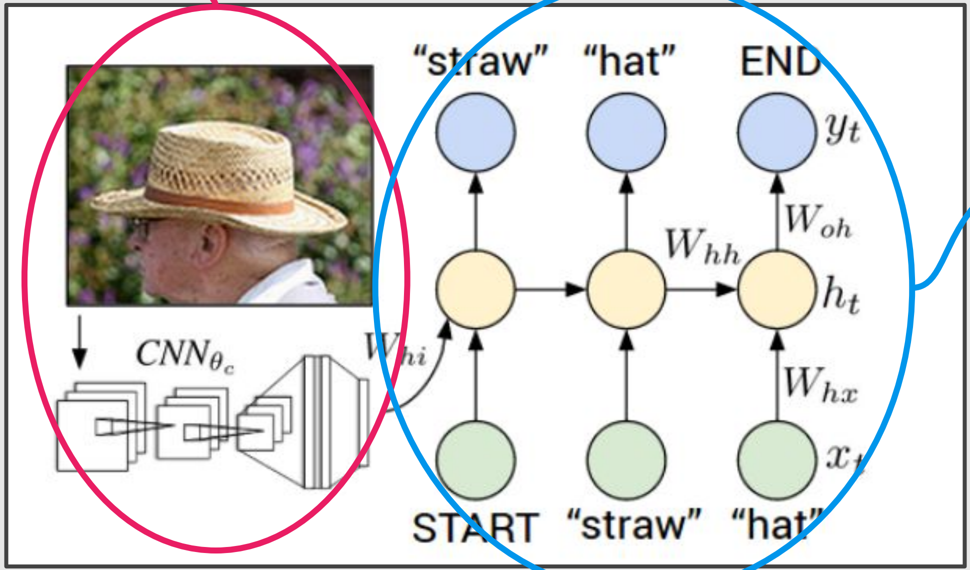
```
iter 0, loss: 107.601633
----
 'ōqIE:ō:3(é
O Q.L"cÉhíL'uàfMO)êoâz.àãâéláç-)D(iéêdàF(lLFLrRcFA0nC(Pô(á#HM5éI?#ázHrtGTRF)5wlGaúa2éj?pd7,u
xp5LQ"r24F7é1efL"Cabvêúhydã 7àã2àObmxv?qnAodí'P)mTg4(u4F7ú13ómrQnmeFNbãoúvâ3i?sxsuRãjáécó.-
záy
---
```

```
iter 46000, loss: 23.238596
----
 és GoogLeNet. E a rede aprede?

O Daras dúvrvilg. ( ende no pré-tro "rar outlara destidas? Com uttres dessar algo us filtros
parte novados aplicar au mula.

e nar
```

```
iter 204000, loss: 10.733449
----
 to, ina utir alpal asvelum motrio tarada mexexenterna mai reviso de enter meiss grandas

##### ResNet Filtro 1x1? Alheing?

Não entendi exatamente que fia, confenhalo deset desecta..

##### Como as
```

# Image Captioning

Convolutional Neural Network




Recurrent Neural Network

Explain Images with Multimodal Recurrent Neural Networks, Mao et al.
Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy and Fei-Fei, CVPR 2015
Show and Tell: A Neural Image Caption Generator, Vinyals et al., CVPR 2015
Long-term Recurrent Convolutional Networks for Visual Recognition and Description, Donahue et al.
Learning a Recurrent Visual Representation for Image Caption Generation, Chen and Zitnick

test image


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax

test image


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096

test image

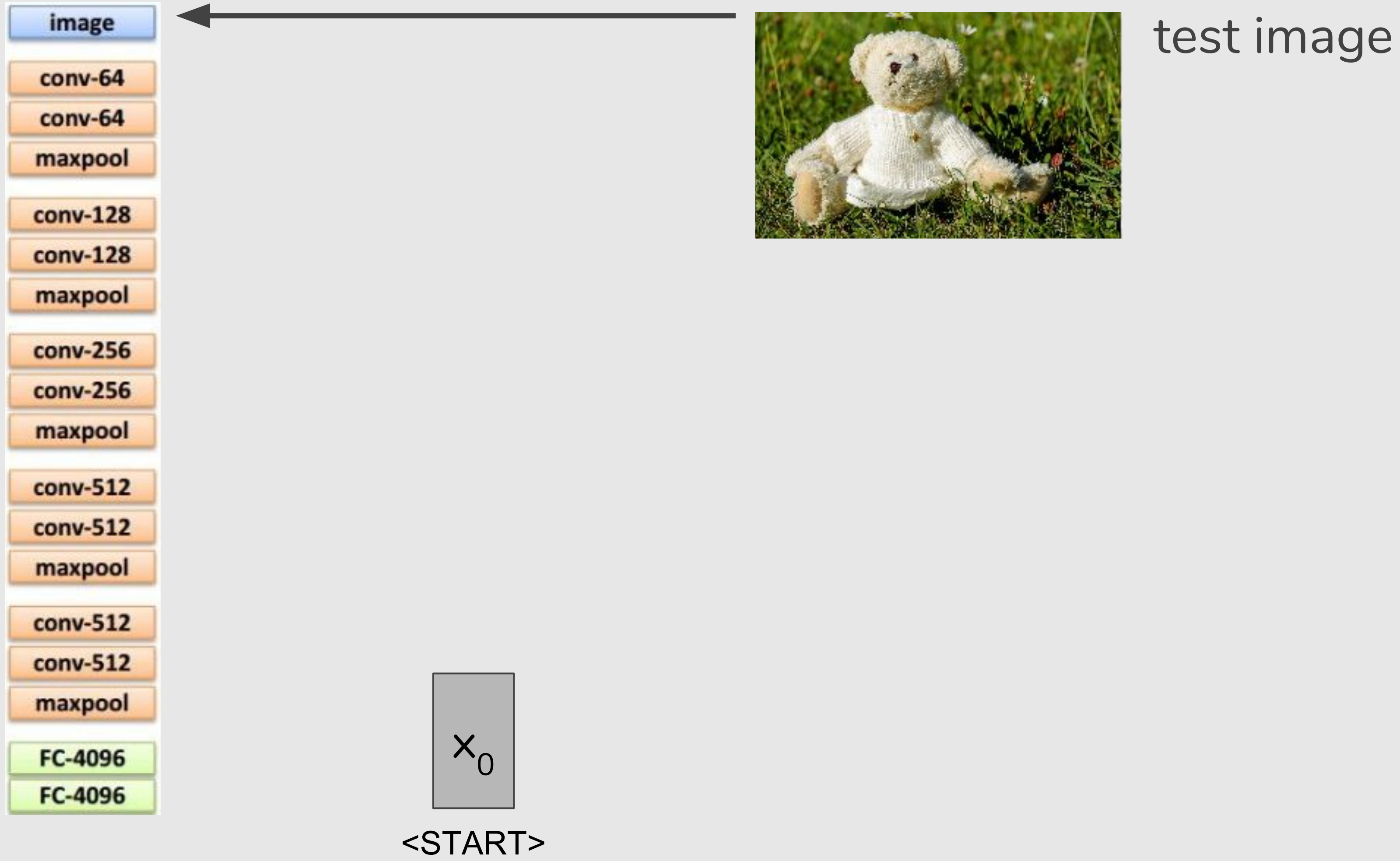
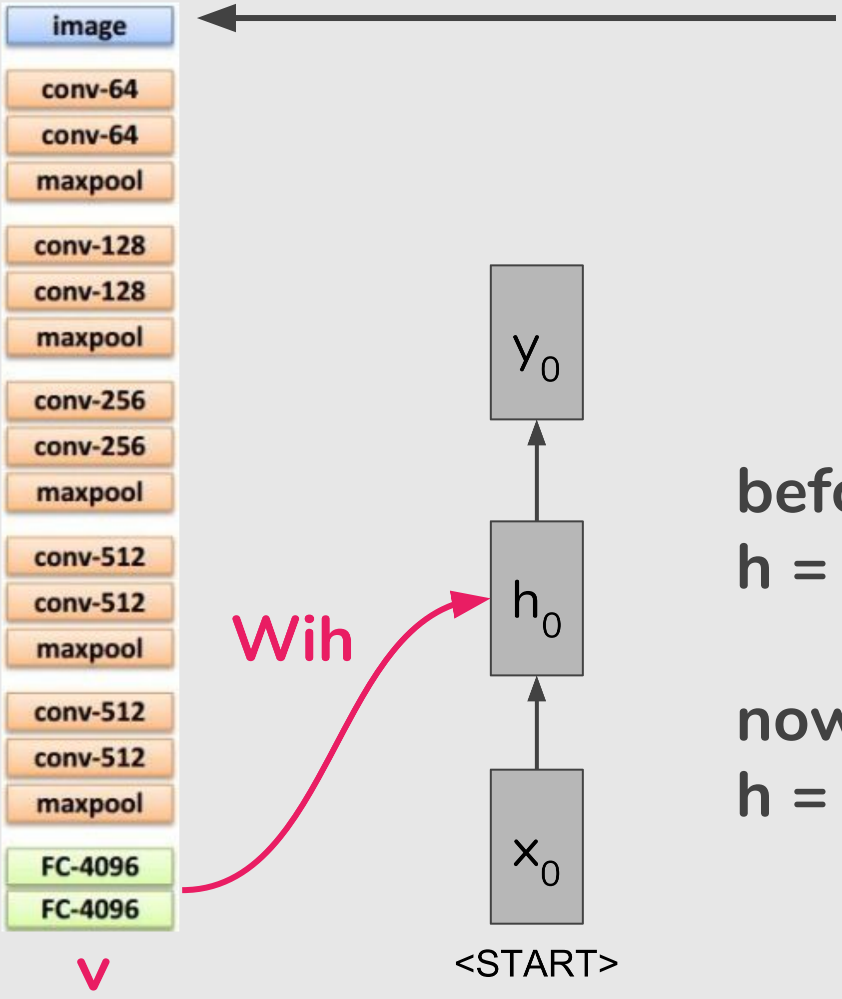


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
$x_0$
<START>




test image

before:

$$h = \tanh(Wxh*x + Whh*h)$$

now:

$$h = \tanh(Wxh*x + Whh*h + Wih*v)$$

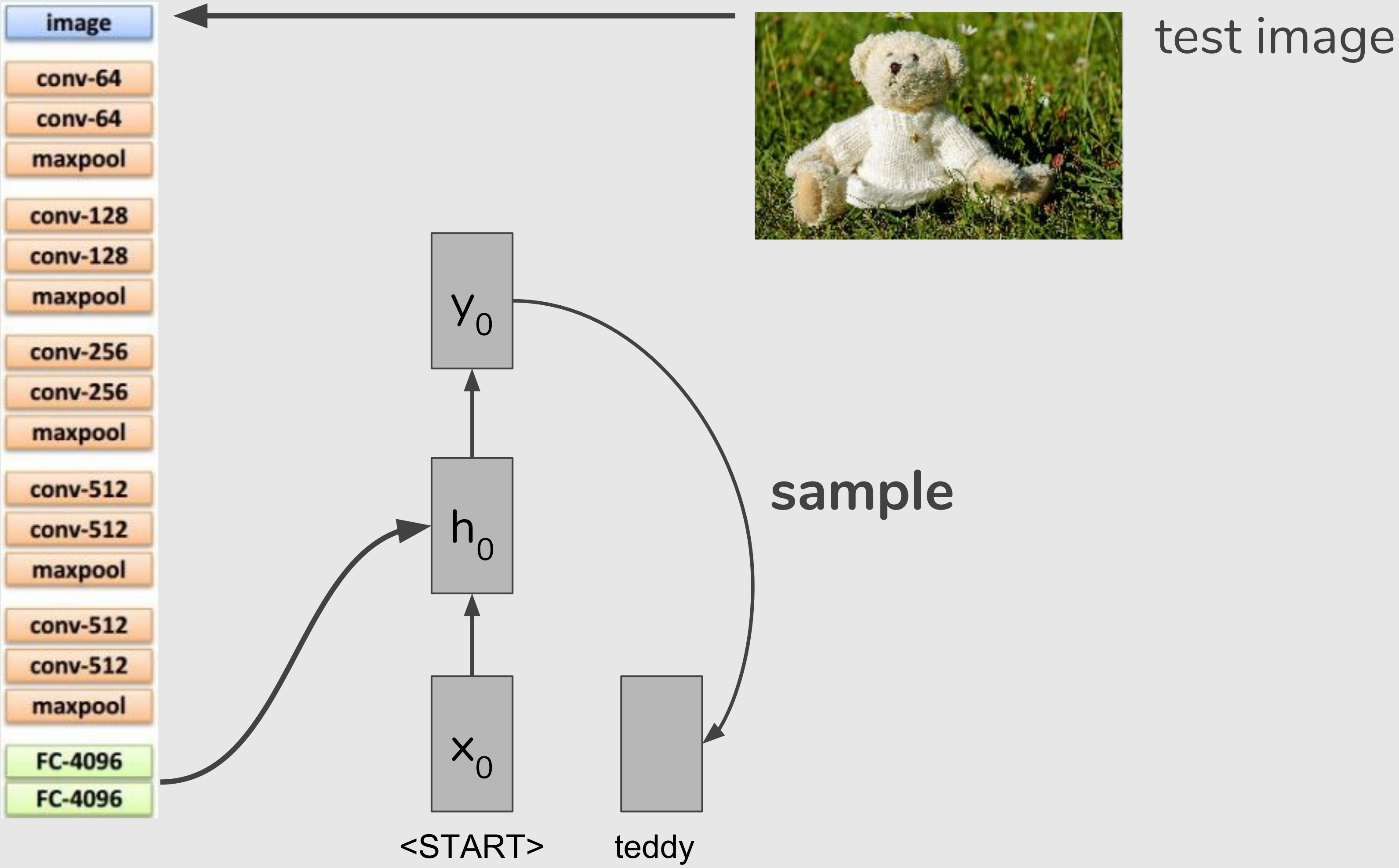


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
$y_0$
$h_0$
$x_0$
sample
<START>
teddy

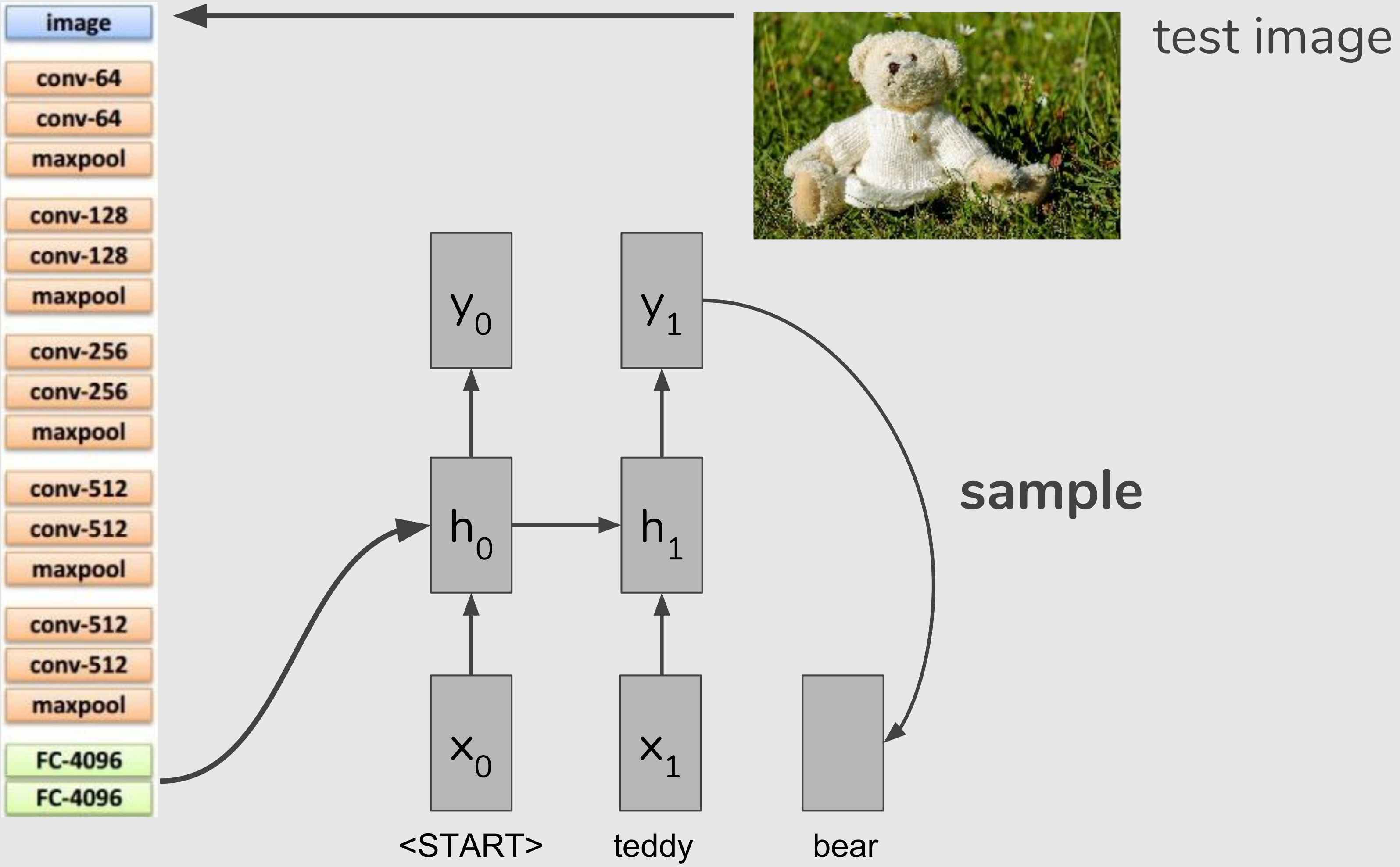


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
$y_0$
$y_1$
$h_0$
$h_1$
$x_0$
$x_1$
sample
<START>
teddy
bear

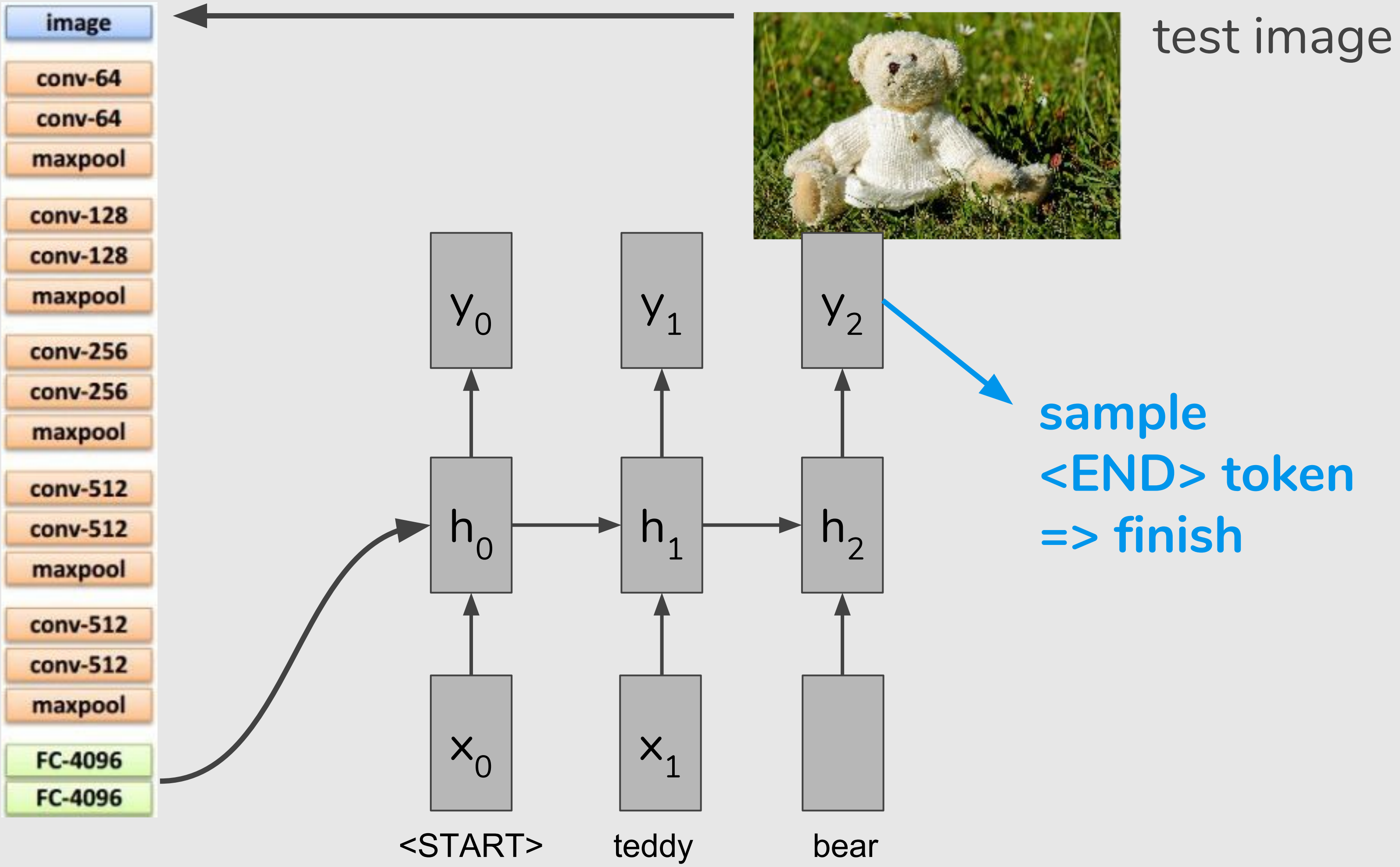


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
$y_0$
$y_1$
$y_2$
$h_0$
$h_1$
$h_2$
$x_0$
$x_1$
<START>
teddy
bear
sample
<END> token
=> finish

# Image Captioning

## No errors

A white teddy bear sitting in the grass

A man riding a wave on top of a surfboard

## Minor errors

A man in baseball uniform throwing a ball

A cat sitting on a suitcase on the floor

## Somewhat related

A woman is holding a cat in her hand

A woman standing on a beach holding a surfboard

# References


## Machine Learning Books

- Deep Learning, http://www.deeplearningbook.org/contents/rnn.html

## Machine Learning Courses

- http://cs231n.stanford.edu/2017/syllabus
- “The 3 popular courses on Deep Learning”: https://medium.com/towards-data-science/the-3-popular-courses-for-deeplearning-ai-ac37d4433bd