Deep Learning

Today: recurrent neural networks, natural language processing

Neural networks

Recall from last time:

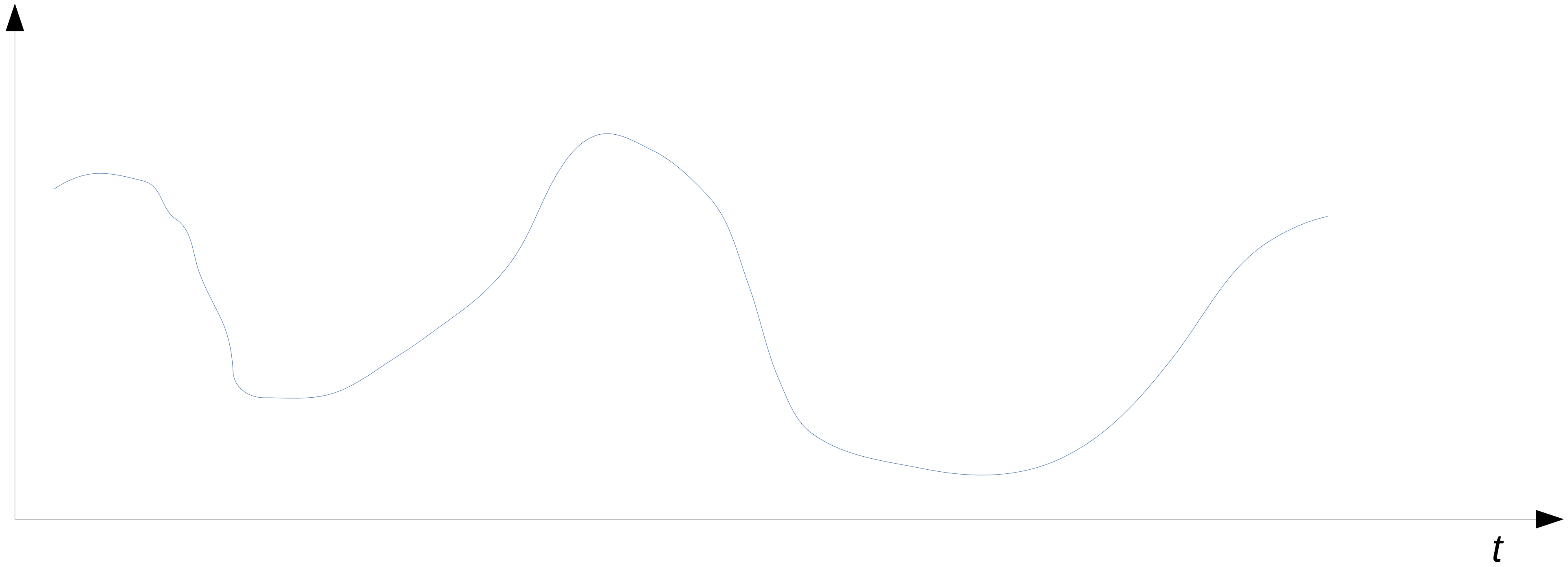
$$f(x; p)$$

Network architecture

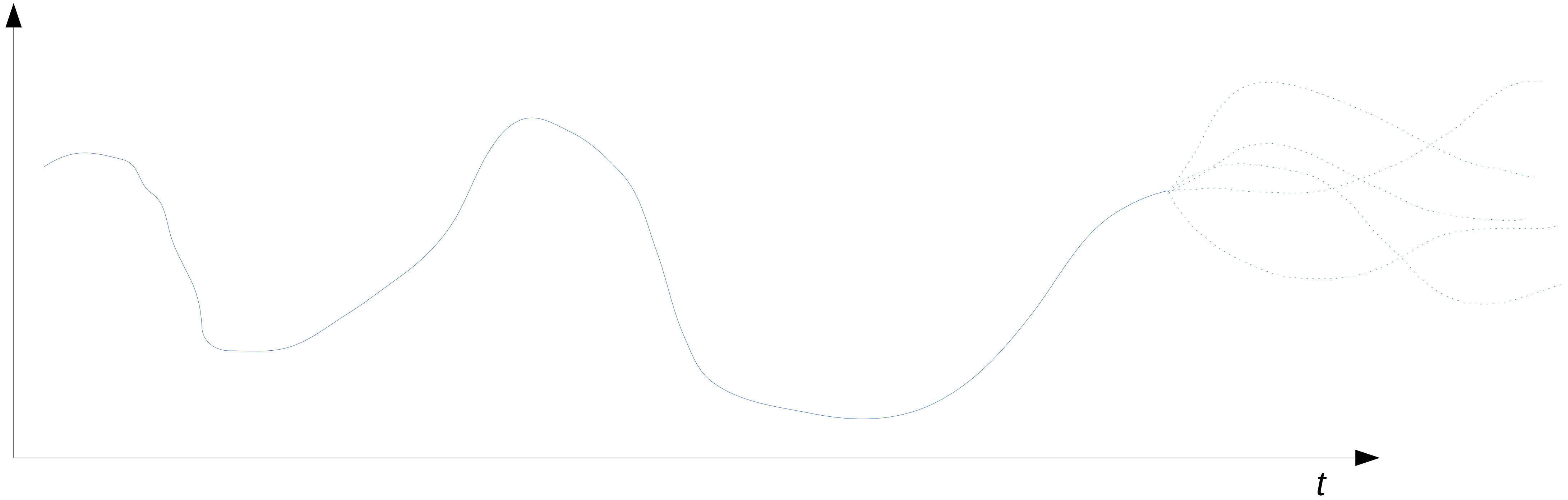
Parameters

Data

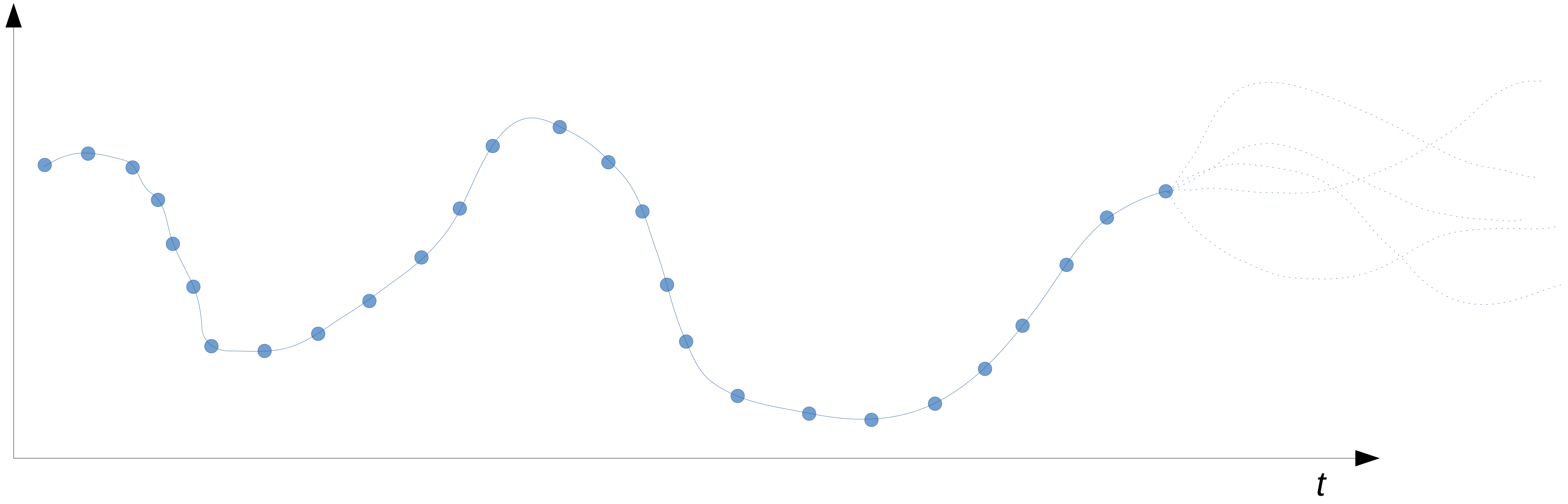
Time-signals



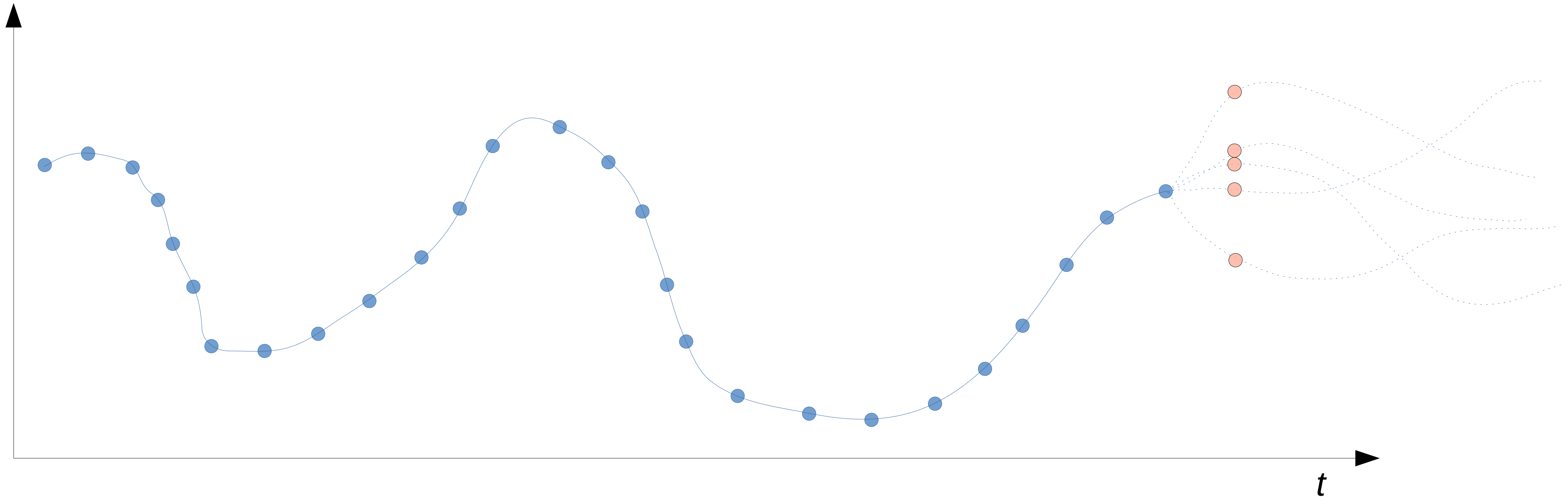
Time-signals



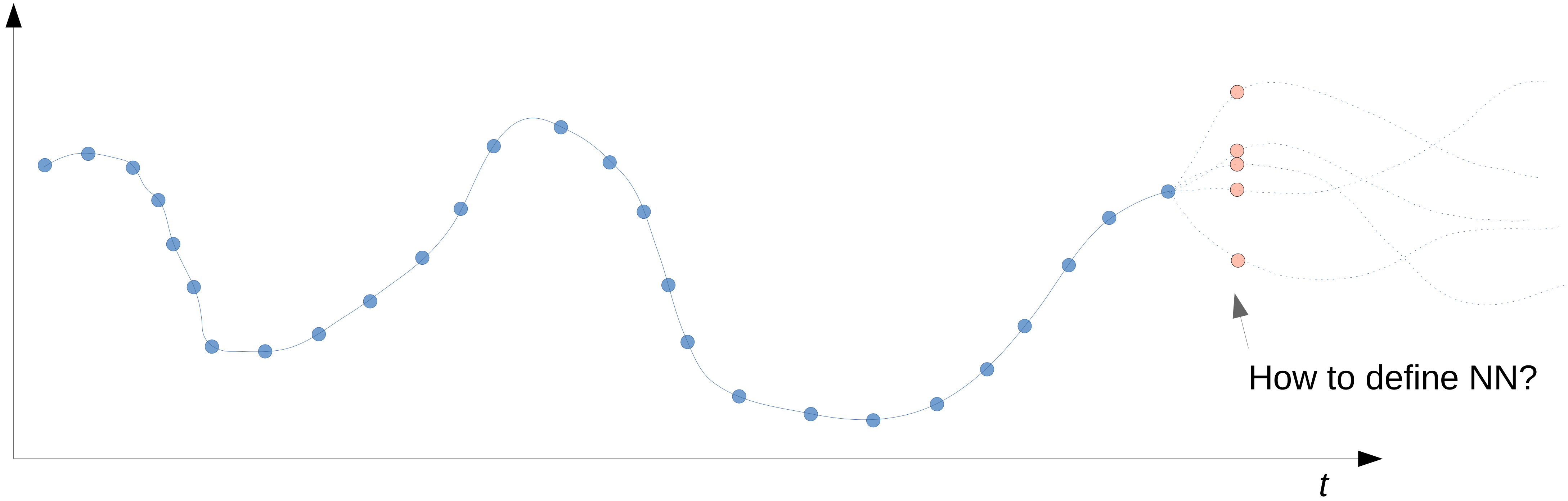
Time-signals



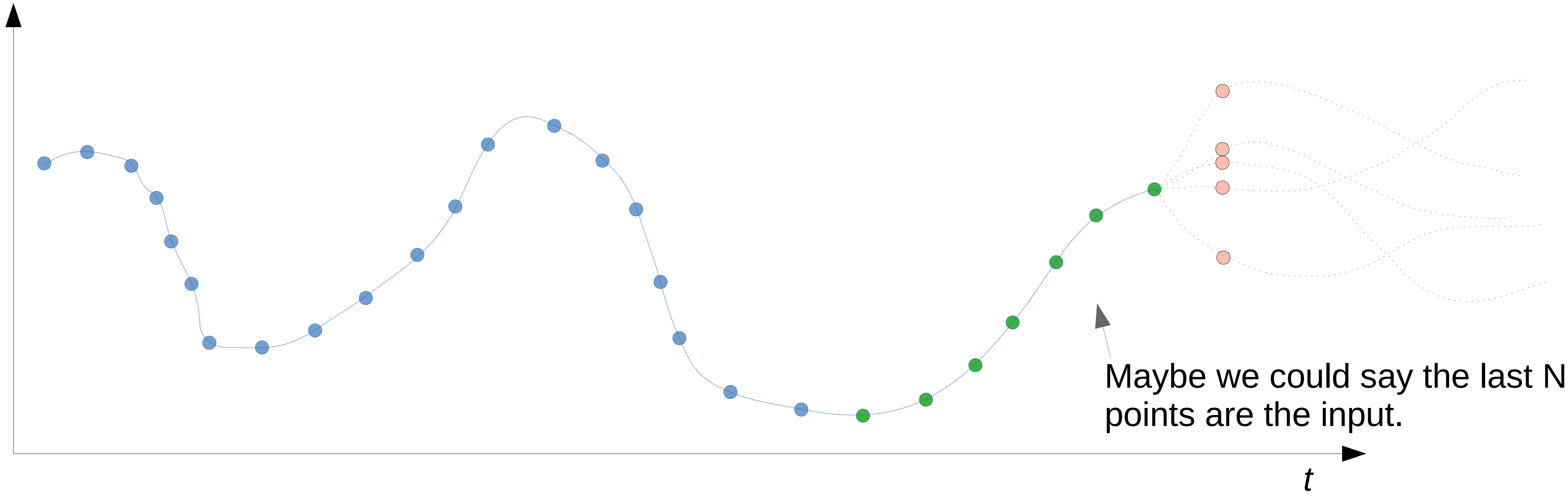
Time-signals



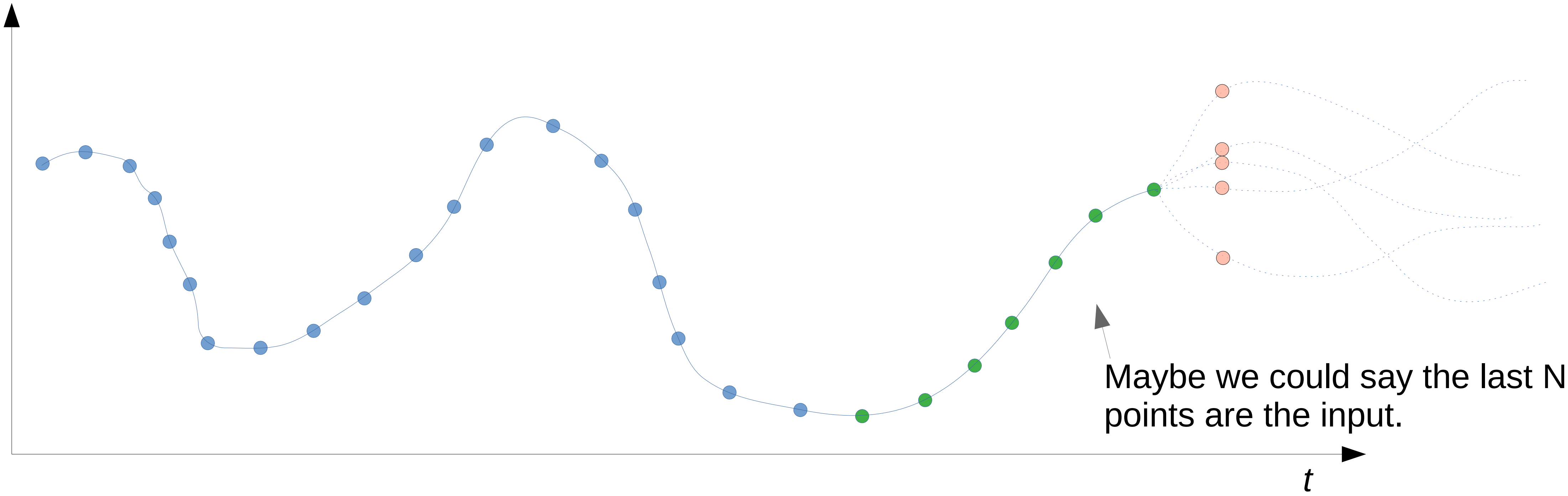
Time-signals



Time-signals

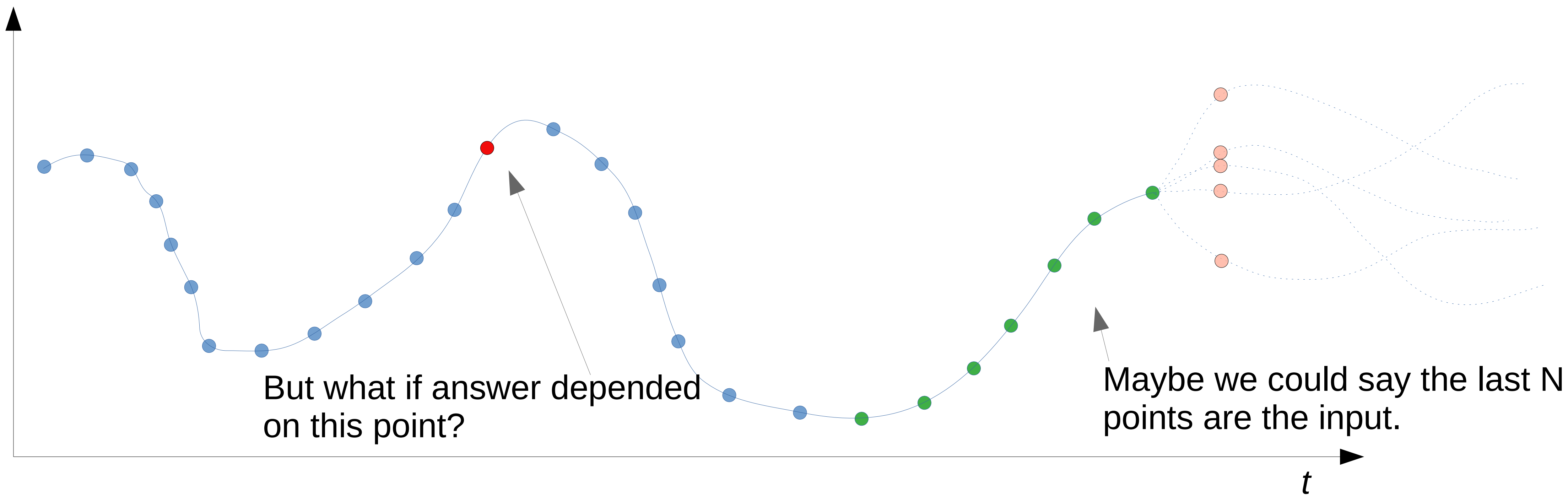


Time-signals



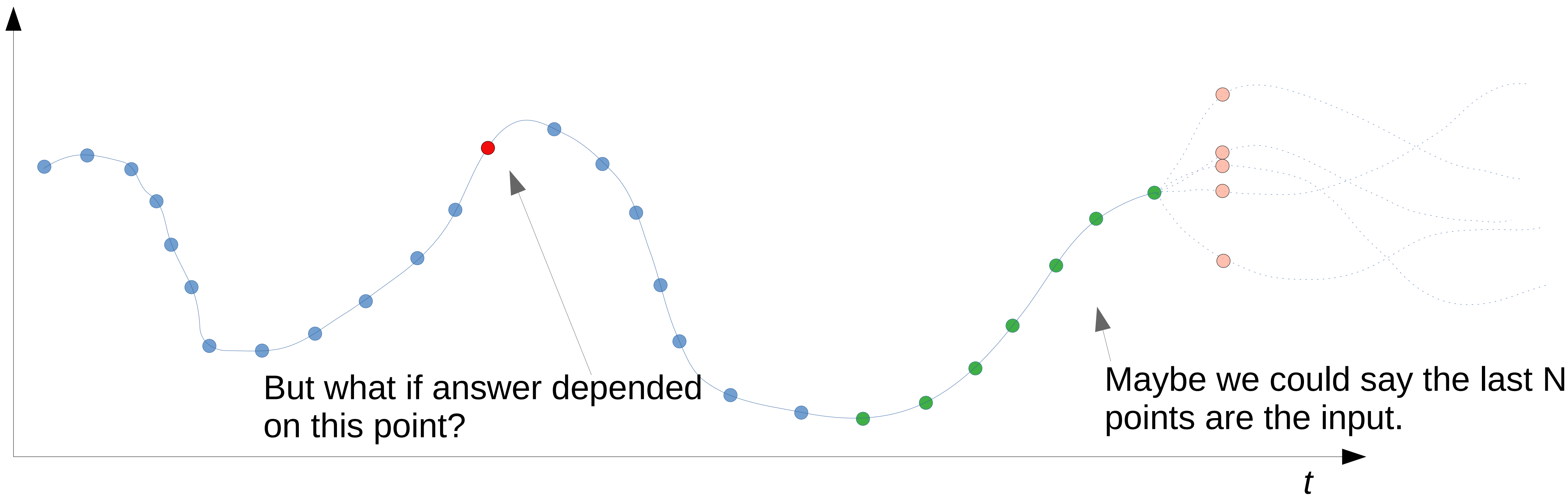
Then we can use a normal network! $f(x; p)$

Time-signals



Then we can use a normal network! $f(x;p)$

Time-signals



Then we can use a normal network! $f(x;p)$

We should not have to make the decision of “how much to memorize”! This should be *trainable*.

Time-signals: More examples

Input has variable length

Examples of output of NN:

- Sell or buy stock?
- Is this a safe heart rate?
- Predict next value
- Signal filtered from noise
- ...

Neural networks

Recall from last time:

$$\underbrace{f}_{\text{Network architecture}}(\underbrace{x}_{\text{Data}};\underbrace{p}_{\text{Parameters}})$$

Neural networks

Recursive neural networks:

$$f(x; h; p)$$

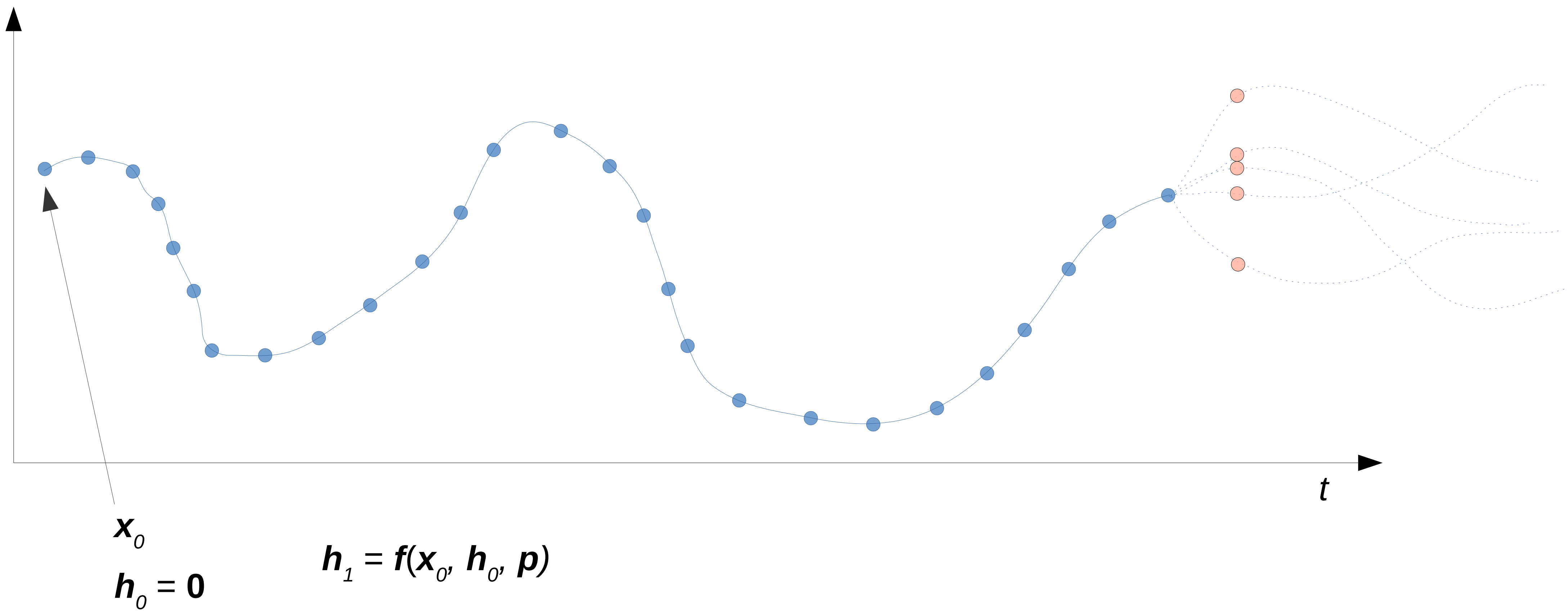
Network architecture

Data

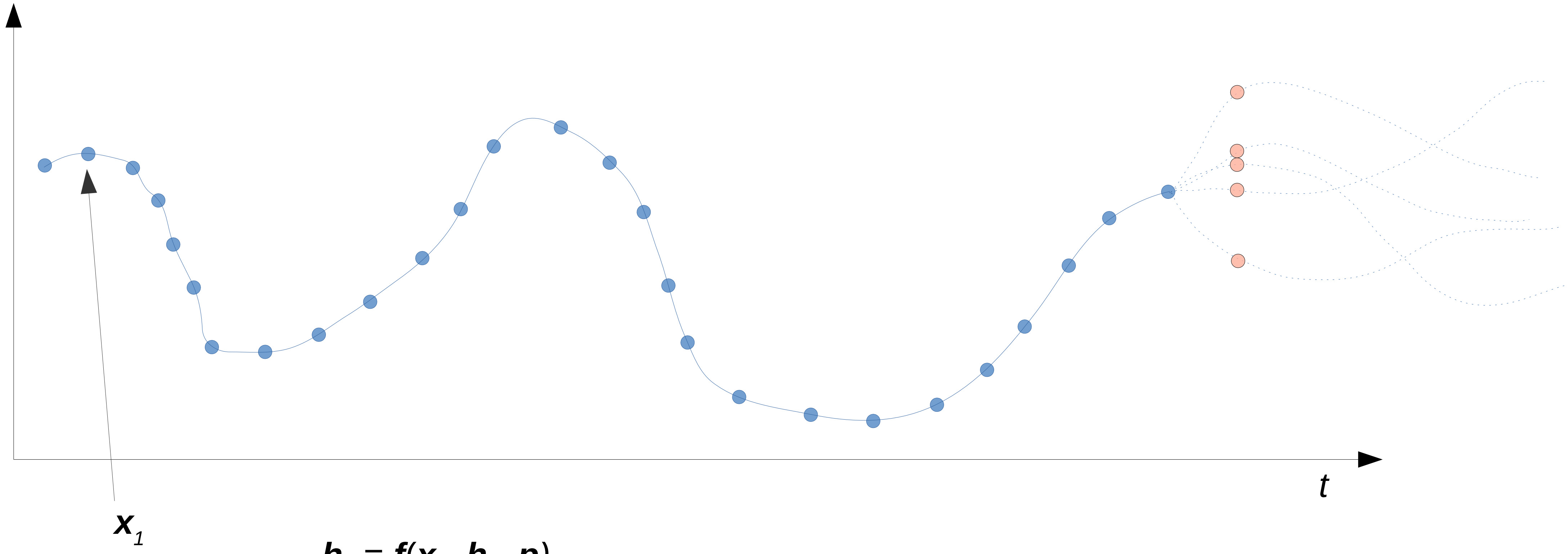
Hidden state

Parameters

Recursive Neural Networks: $f(x; h; p)$

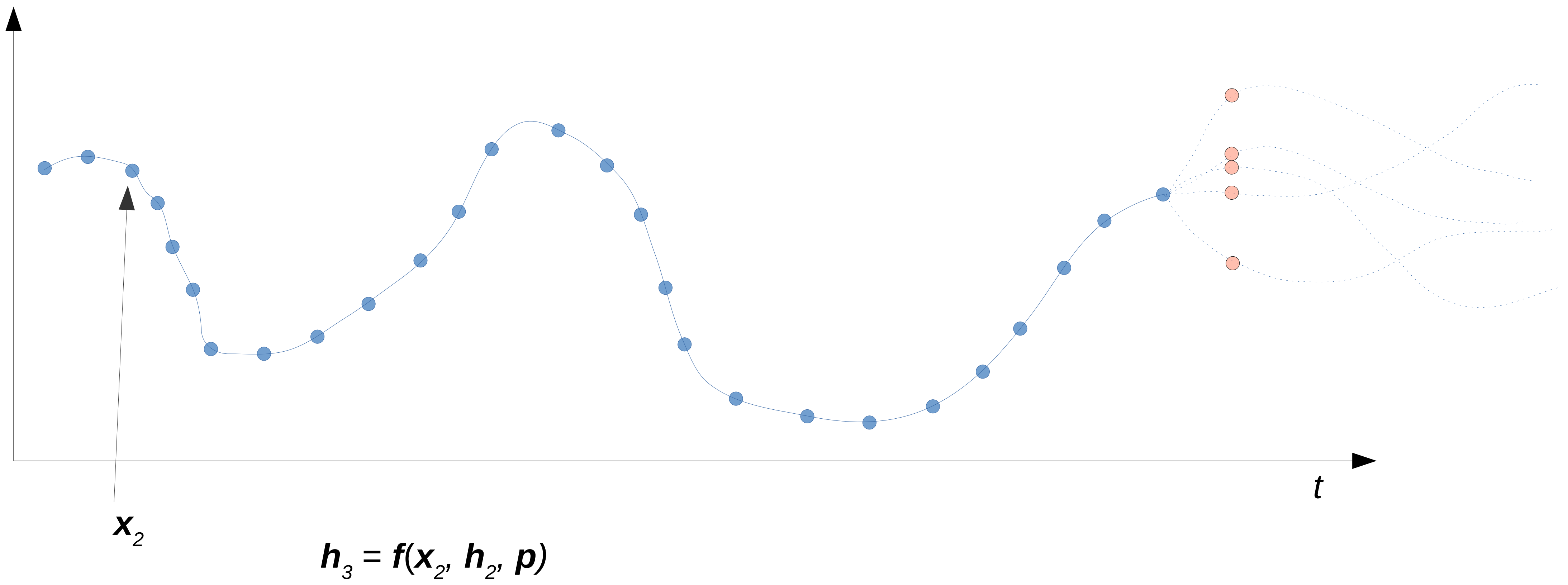


Recursive Neural Networks: $f(x; h; p)$

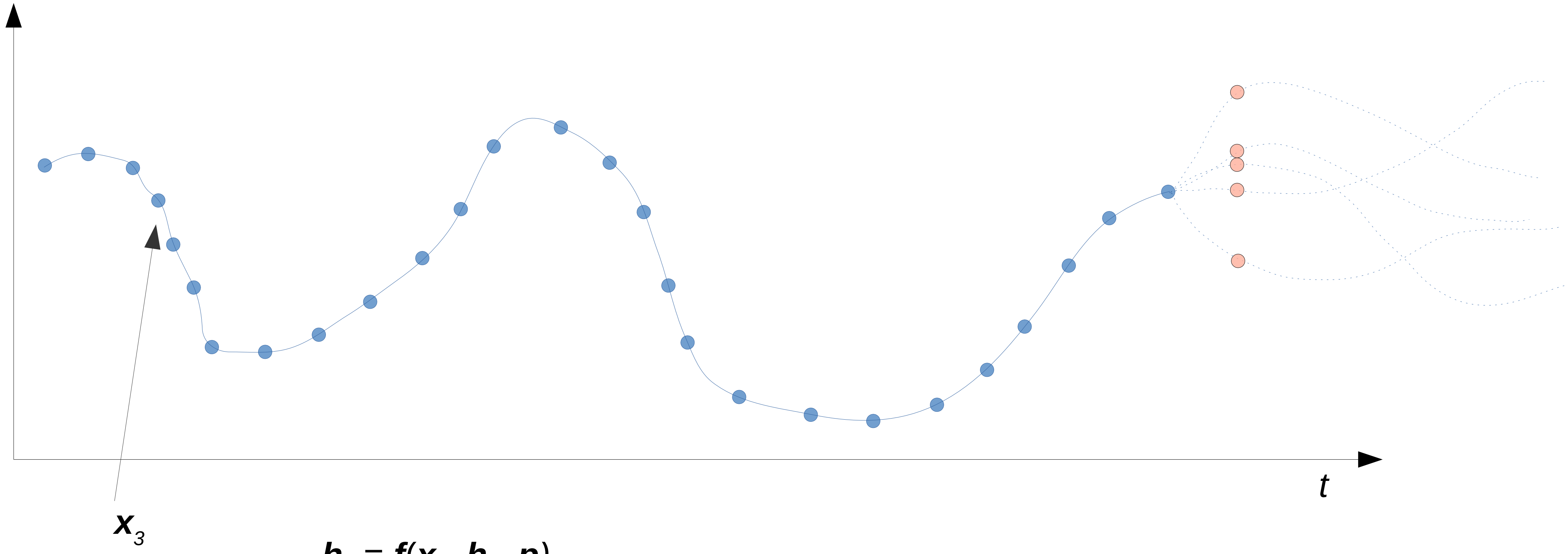


$h_2 = f(x_1, h_1, p)$

Recursive Neural Networks: $f(x; h; p)$

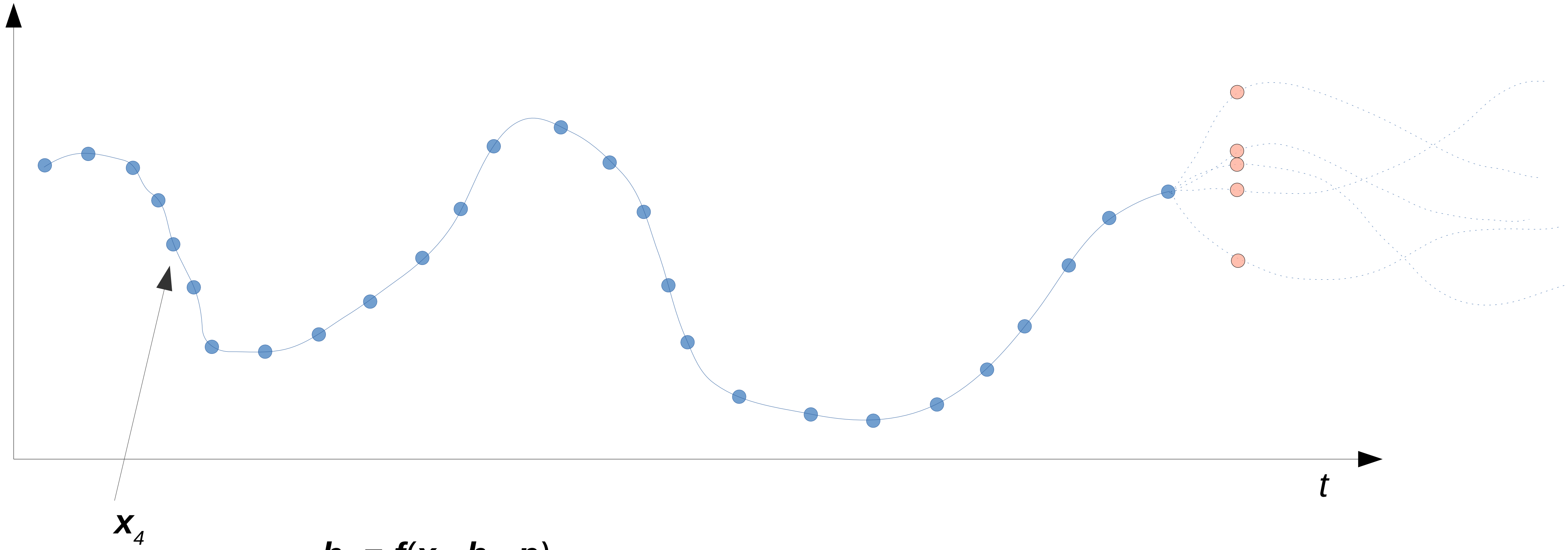


Recursive Neural Networks: $f(x; h; p)$



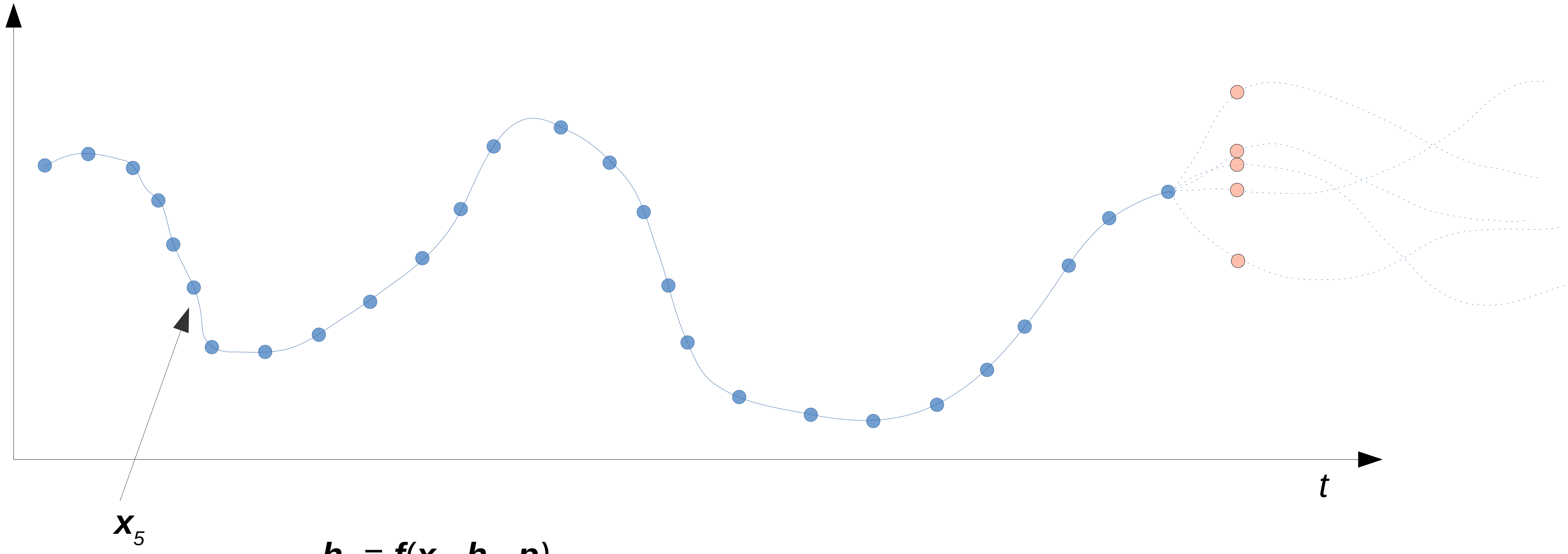
$h_4 = f(x_3, h_3, p)$

Recursive Neural Networks: $f(x; h; p)$



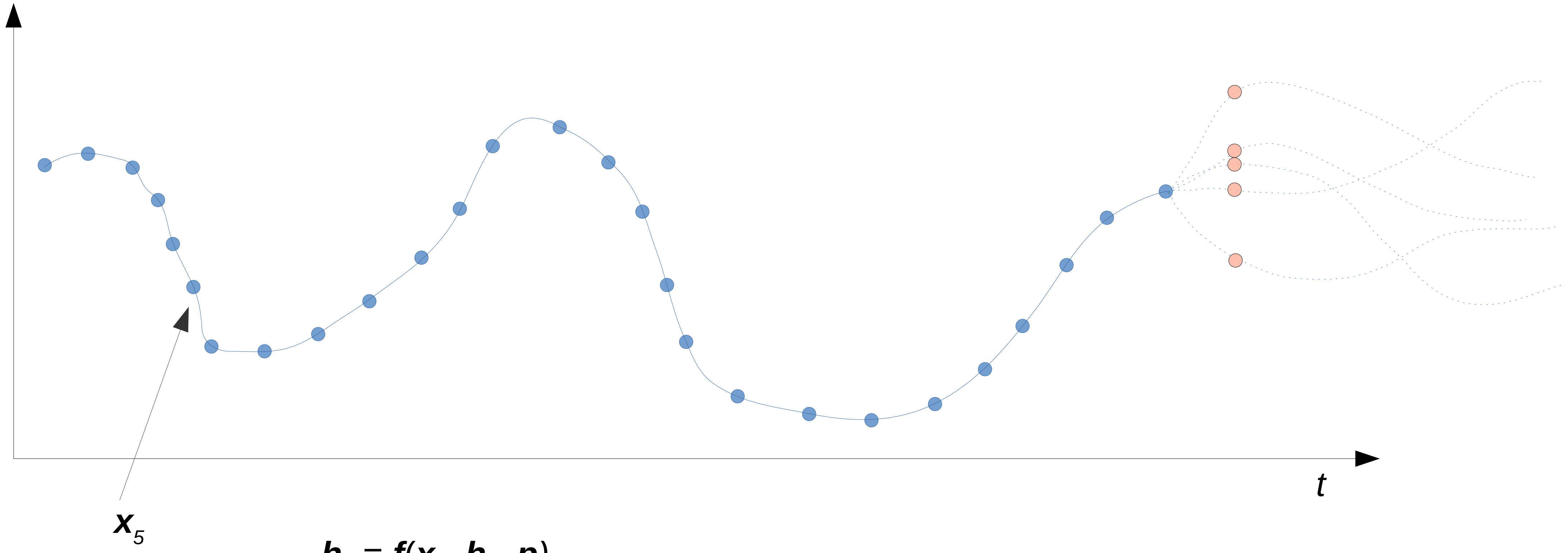
$$h_5 = f(x_4, h_4, p)$$

Recursive Neural Networks: $f(x; h; p)$



$\boldsymbol{h}_6 = \boldsymbol{f}(\boldsymbol{x}_5, \boldsymbol{h}_5, \boldsymbol{p})$

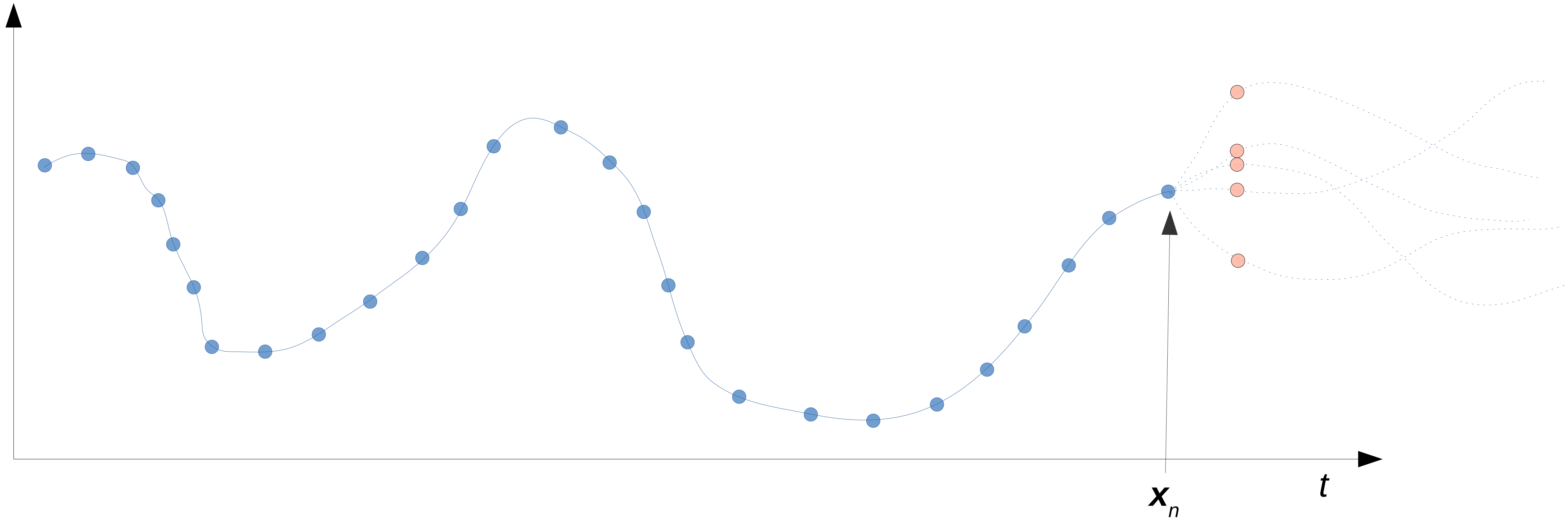
Recursive Neural Networks: $f(x; h; p)$



$\boldsymbol{h}_6 = \boldsymbol{f}(\boldsymbol{x}_5, \boldsymbol{h}_5, \boldsymbol{p})$

The hidden state ***h*** can memorize important information about the time signal.

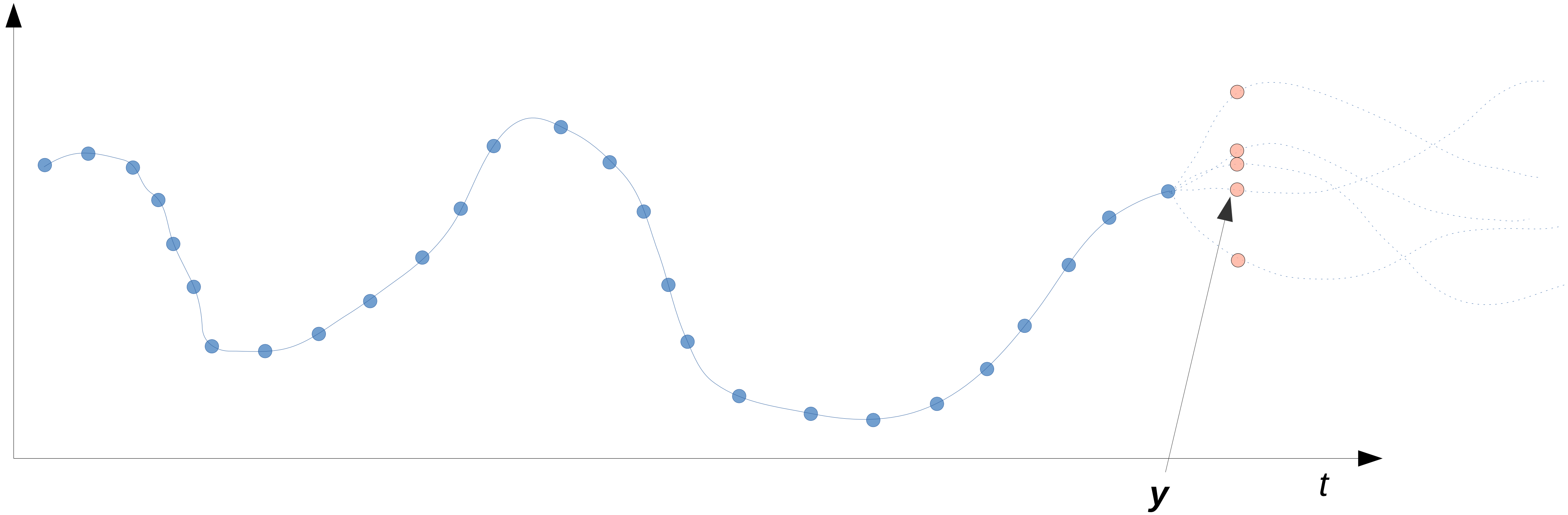
Recursive Neural Networks: $f(x; h; p)$



$\boldsymbol{h}_{final} = \boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{h}_n, \boldsymbol{p})$

The final hidden state should thus contain useful information to be used in a standard NN

Recursive Neural Networks: $f(x; h; p)$

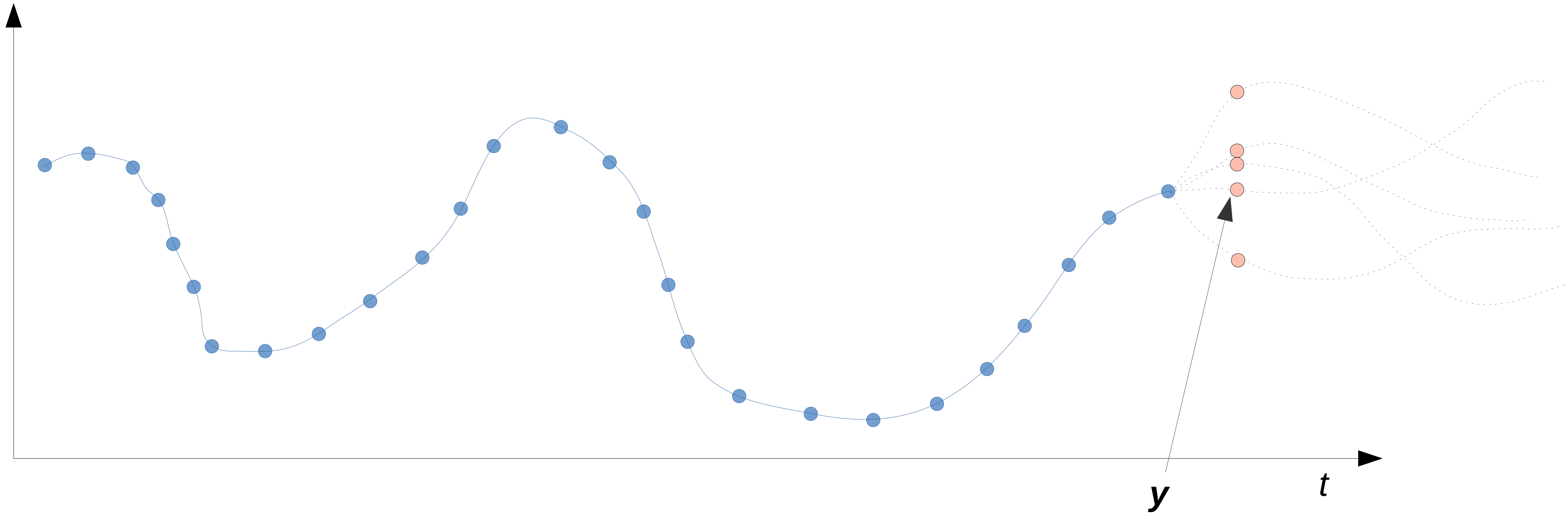


$\boldsymbol{h}_{final} = \boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{h}_n, \boldsymbol{p})$

$\boldsymbol{y} = \mathrm{NN}(\boldsymbol{h}_{final})$

The final hidden state should thus contain useful information to be used in a standard NN

Recursive Neural Networks: $f(x;h;p)$

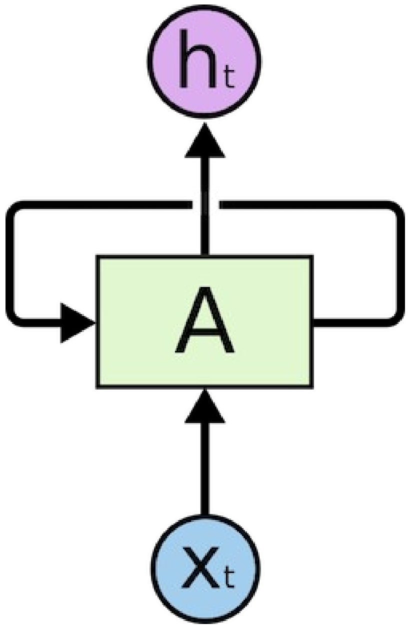


$\boldsymbol{h}_{final} = \boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{h}_n, \boldsymbol{p})$

$\boldsymbol{y} = \mathrm{NN}(\boldsymbol{h}_{final})$

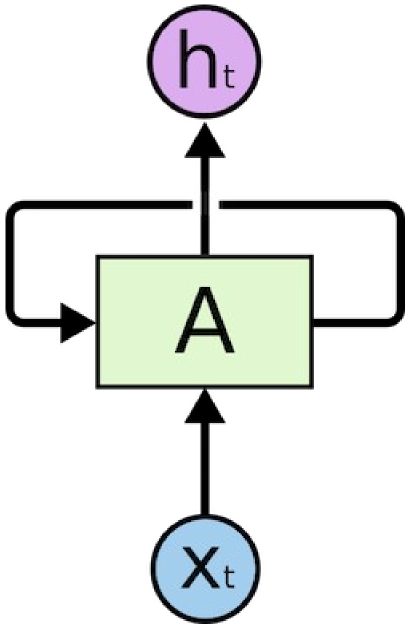
Note that the recursive nature makes it natural to handle input of variable length!

Recursive Neural Networks: $f(x; h; p)$



$$h_1 = f(x_0; h_0; p) \quad h_2 = f(x_2; h_1; p) \quad h_3 = f(x_2; h_2; p)$$

Recursive Neural Networks: $f(x; h; p)$



$$h_1 = f(x_0; h_0; p) \quad h_2 = f(x_2; h_1; p) \quad h_3 = f(x_2; h_2; p)$$

Example: (“classic” RNN):

$$h_1 = \tanh(W_{ih} x_0 + W_{hh} h_0 + b)$$

Language Modelling

$$f(x; h; p)$$

Hi mom, I'll be late for …

$f(\text{‘}late\text{‘}; h_3; p) = h_4$ $f(\text{‘}for\text{‘}; h_4; p) = h_5$ $f(\text{‘}dinner\text{‘}; h_5; p) = h_6$

$h_i = \text{hidden state}$

h_N can be used to predict next word

Language Modelling

$$f(x; h; p)$$

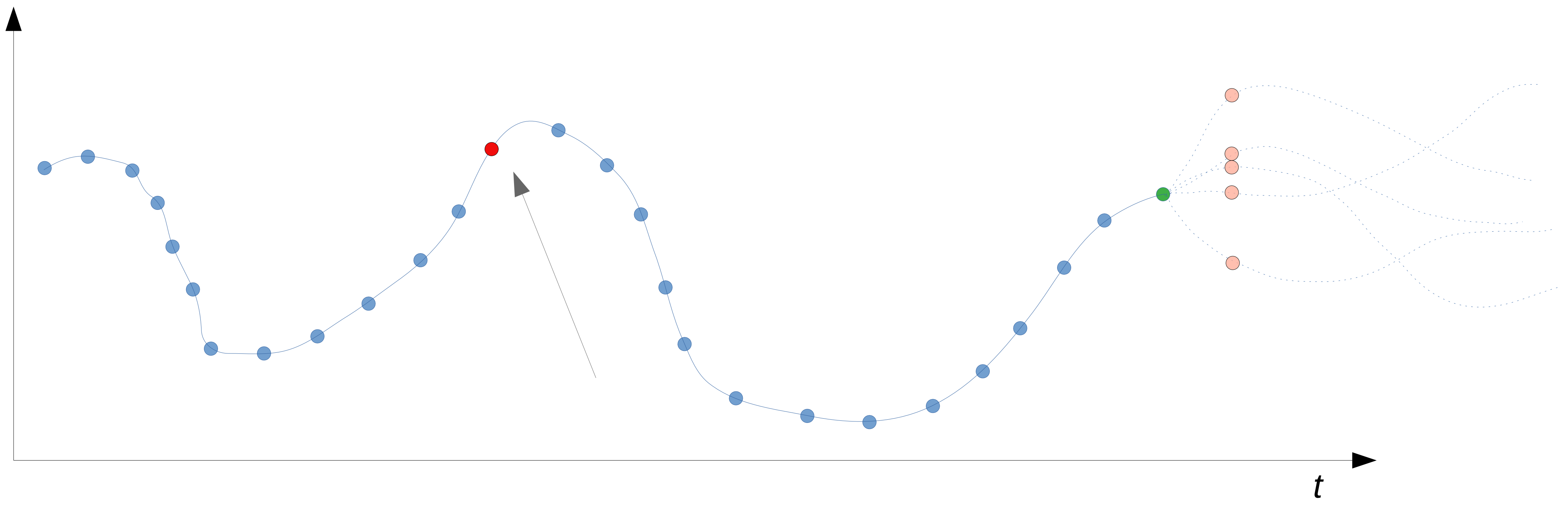
Hi mom, I’ll be late for …

$f(\text{‘}late\text{‘}; h_3; p) = h_4$ $f(\text{‘}for\text{‘}; h_4; p) = h_5$ $f(\text{‘}dinner\text{‘}; h_5; p) = h_6$

$h_i = \text{hidden state}$

h_N can be used to predict next word

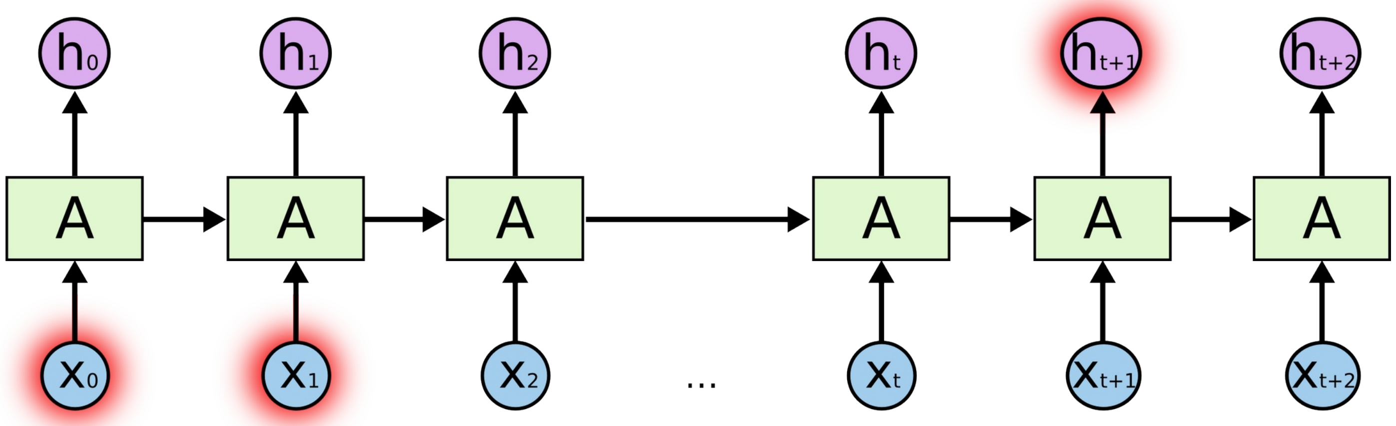
Time-signals



We hope that the state can memorize information
that appeared early in the signal!

Language Modelling

$$f(x; h; p)$$

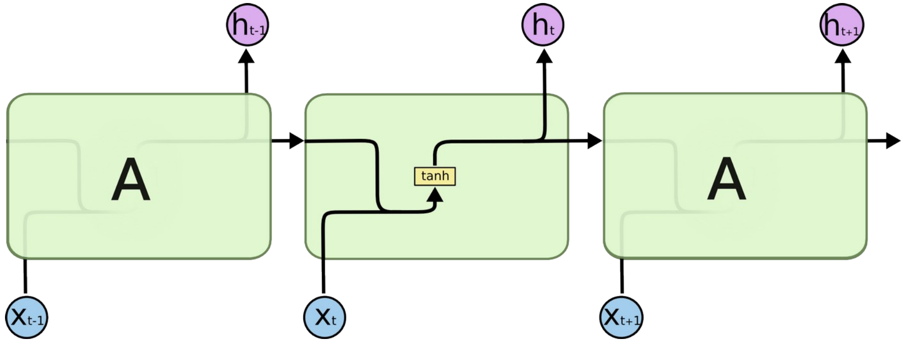


"I grew up in France"

"Since my mother tongue is ____"

Recursive neural networks $f(x; h; p)$

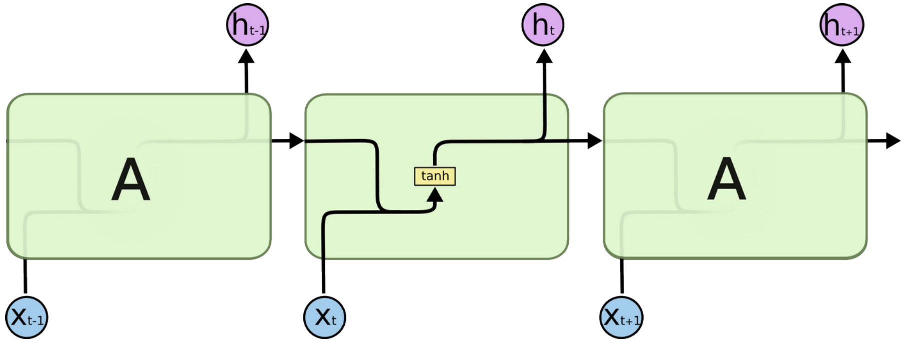
So can the standard RNN remember far back?



$$h' = \tanh(W_{ih}x + b_{ih} + W_{hh}h + b_{hh})$$

Recursive neural networks $f(x; h; p)$

So can the standard RNN remember far back?



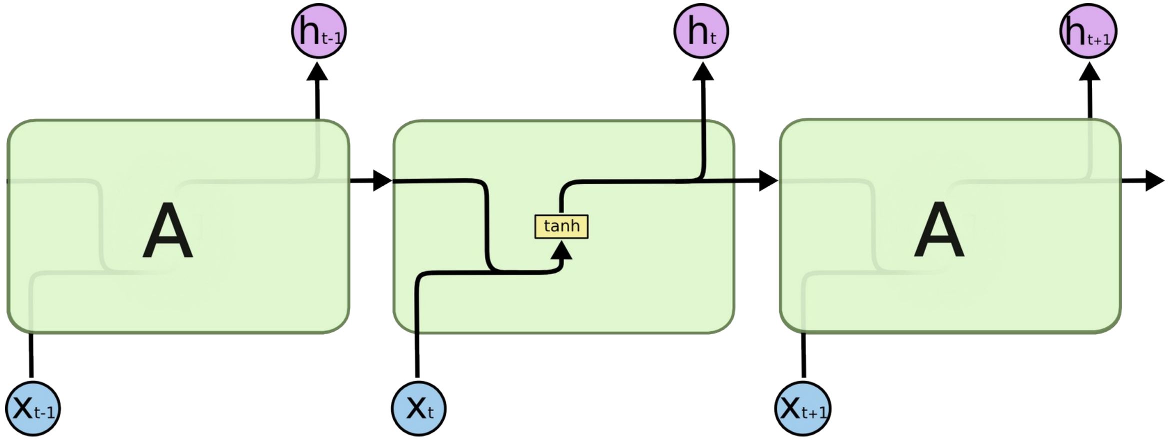
$$h' = \tanh(W_{ih}x + b_{ih} + W_{hh}h + b_{hh})$$

No! ☹

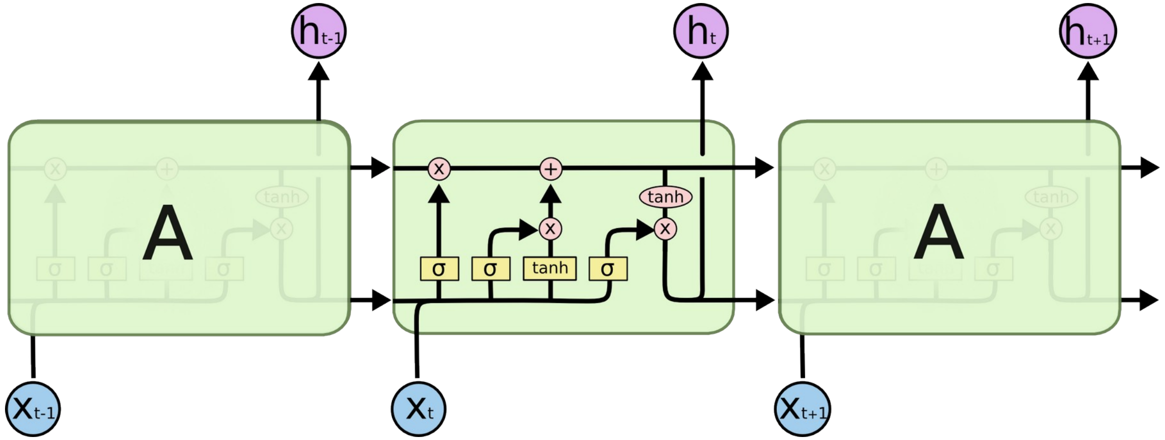
(basically it forgets exponentially fast!)

LSTM: Long Short Term Memory

Standard RNN:



LSTM:

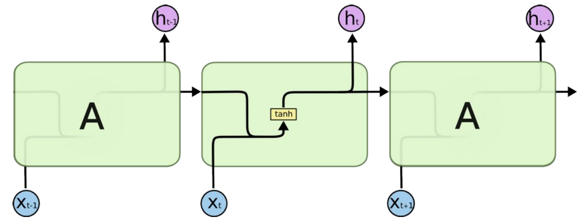


See https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM: Long Short Term Memory

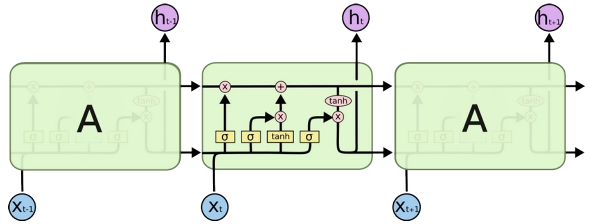
Standard RNN:

$$h' = \tanh(W_{ih}x + b_{ih} + W_{hh}h + b_{hh})$$



LSTM:

$$i = \sigma(W_{ii}x + b_{ii} + W_{hi}h + b_{hi})$$
$$f = \sigma(W_{if}x + b_{if} + W_{hf}h + b_{hf})$$
$$g = \tanh(W_{ig}x + b_{ig} + W_{hg}h + b_{hg})$$
$$o = \sigma(W_{io}x + b_{io} + W_{ho}h + b_{ho})$$
$$c' = f * c + i * g$$
$$h' = o * \tanh(c')$$

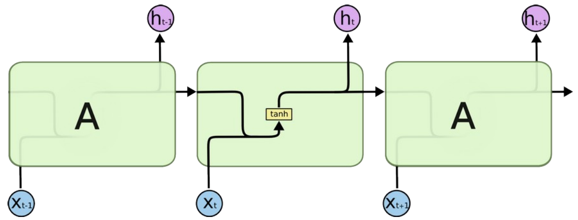


LSTM: Long Short Term Memory

Standard RNN:

$$h' = \tanh(W_{ih}x + b_{ih} + W_{hh}h + b_{hh})$$

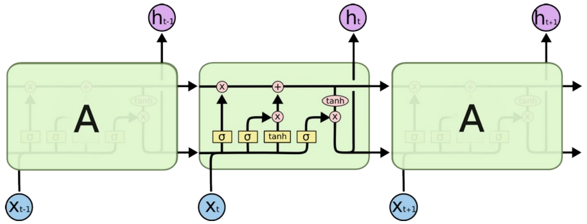
```
nn.RNN(input_size, hidden_size)
```



LSTM:

$$
\begin{aligned}
i &= \sigma(W_{ii}x + b_{ii} + W_{hi}h + b_{hi}) \\
f &= \sigma(W_{if}x + b_{if} + W_{hf}h + b_{hf}) \\
g &= \tanh(W_{ig}x + b_{ig} + W_{hg}h + b_{hg}) \\
o &= \sigma(W_{io}x + b_{io} + W_{ho}h + b_{ho}) \\
c' &= f * c + i * g \\
h' &= o * \tanh(c')
\end{aligned}
$$

```
nn.LSTM(input_size, hidden_size)
```



Time signals

Applying LSTM’s (or other RNN’s) to time signal is fairly straight-forward.

A couple of pointers:

- Standard convention for RNN’s is to have batch-dimension as *second* axis.

- Can be hard to manage variable-sized input (does not fit into a standard tensor)

- Reference implementation often pad input (and for this course, we **recommend** this, too!)

Time signals

Applying LSTM’s (or other RNN’s) to time signal is fairly straight-forward.

A couple of pointers:

- Standard convention for RNN’s is to have batch-dimension as *second* axis.

- Can be hard to manage variable-sized input (does not fit into a standard tensor)

- Reference implementation often pad input (and for this course, we **recommend** this, too!)

```
import torch
from torch import nn
from torch.nn.utils.rnn import pack_sequence, pad_sequence
```

```
net = nn.LSTM(input_size=2, hidden_size=7, batch_first=True)
```

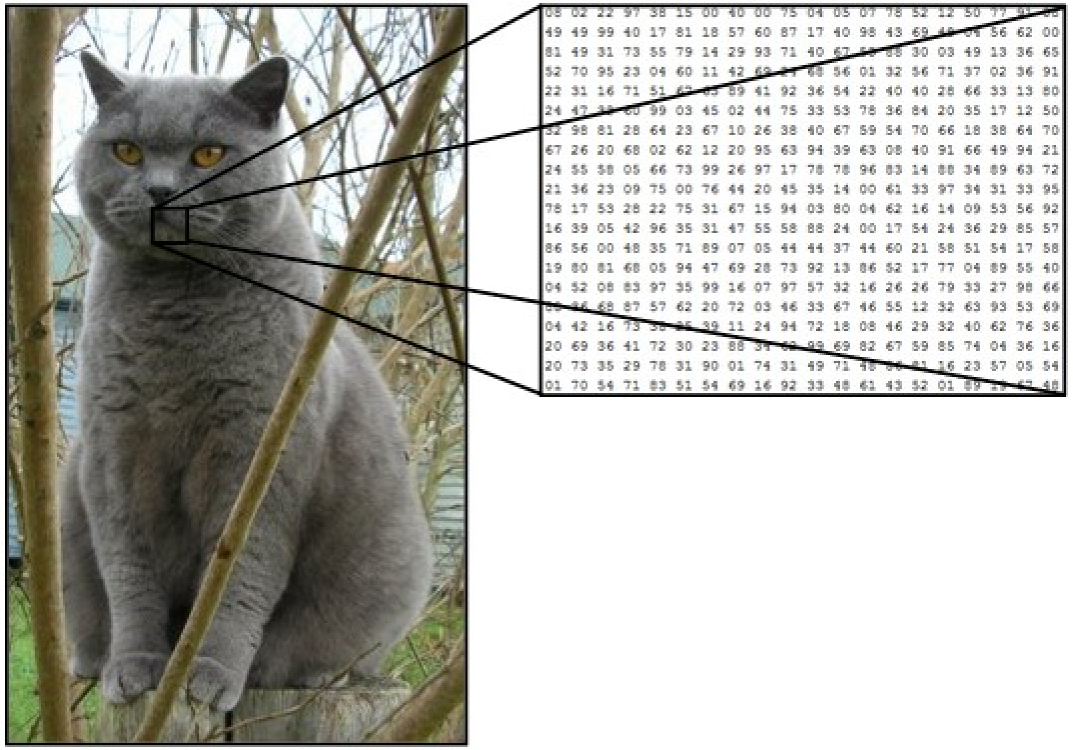
```
x1 = torch.tensor([[1., 2.], [2., 3.], [3., 4.], [4., 5.], [5., 6.]])
x2 = torch.tensor([[5., 5.], [6., 6.], [7., 7.]])
data = pack_sequence([x1, x2])
```

```
x1 = torch.tensor([[1., 2.], [2., 3.], [3., 4.], [4., 5.], [5., 6.]])
x2 = torch.tensor([[5., 5.], [6., 6.], [7., 7.]])
data = pad_sequence([x1, x2])
```

Natural Language Processing

HOW TO REPRESENT WORDS / SENTENCES?

Images



The Trouble

“How are you?” →

Bag of Words

“How are you” $\longrightarrow$ $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$

“How how” $\longrightarrow$ $\begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix}$

Bag of Words, poor behaviour #1

“I had my car cleaned.” $\longrightarrow \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$

“I had cleaned my car.” $\longrightarrow \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$ (order ignored)

Bag of Words, poor behaviour #2

“Good day mate!” $\longrightarrow \begin{pmatrix}1\\1\\1\\0\\0\\0\end{pmatrix}$

“Gooood day mate” $\longrightarrow \begin{pmatrix}0\\1\\1\\1\\0\\0\end{pmatrix}$ (semantically similar)

“God dayy mate” $\longrightarrow \begin{pmatrix}0\\0\\1\\0\\1\\1\end{pmatrix}$ (typos)

Bag of Words, poor behaviour #3

$$\text{“Good day mate”} \longrightarrow \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

The idea for a solution

Idea: Represent each word as a vector.

The idea for a solution

Idea: Represent each word as a vector.

Vectors of similar words should be “close” somehow.

The idea for a solution

Idea: Represent each word as a vector.

Vectors of similar words should be “close” somehow.

What dimensions should these vectors be?
And how should one calculate such vectors?

The idea for a solution

”Context defines meaning”:

The country was ruled by a _____

The bishop anointed the ____ with aromatic oils

The crown was put on the ____

The idea for a solution

”Context defines meaning”:

The country was ruled by a _____

The bishop anointed the ____ with aromatic oils

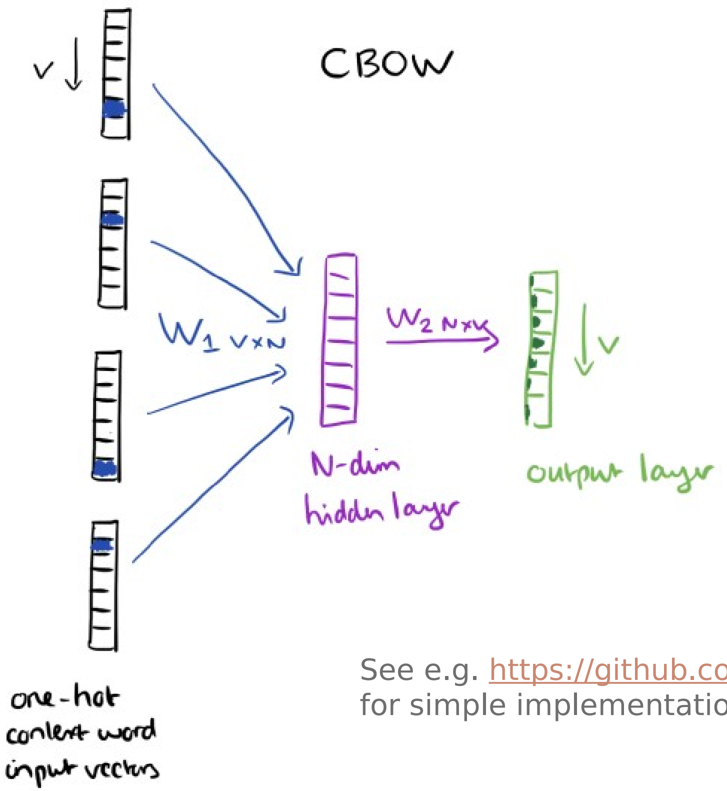
The crown was put on the ____

King/Queen

Continous Bag of Words

The bishop anointed the **queen** with aromatic oils

Context — Focus word — Context



- Input is a ”one-hot” vector
- We force network to make each word into a ~200 length vector
- From these vectors we predict ”focus word”
- When done, keep ”embeddings”

See e.g. https://github.com/FraLotito/pytorch-continuous-bag-of-words/blob/master/cbow.py for simple implementation

Continous Bag of Words

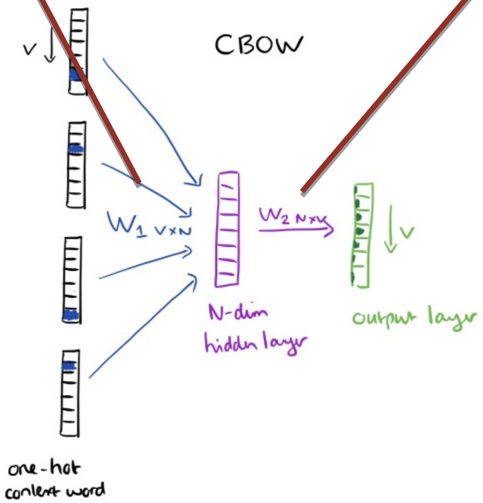
I think **therefore** I am

Context — Focus word — Context

Dictionary: [“I”, “think”, “therefore”, “am”]

Context size = 2

$$\begin{pmatrix}1\\2\\1\\4\end{pmatrix} \rightarrow \begin{pmatrix}1.2 & -0.5 & \cdots\\ -0.1 & 2.3 & \cdots\\ 1.2 & -0.5 & \cdots\\ -1.1 & 0.1 & \cdots\end{pmatrix} \rightarrow \begin{pmatrix}0.05\\0.05\\0.88\\0.02\end{pmatrix} \qquad y = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}$$



Continous Bag of Words

Very simple version:

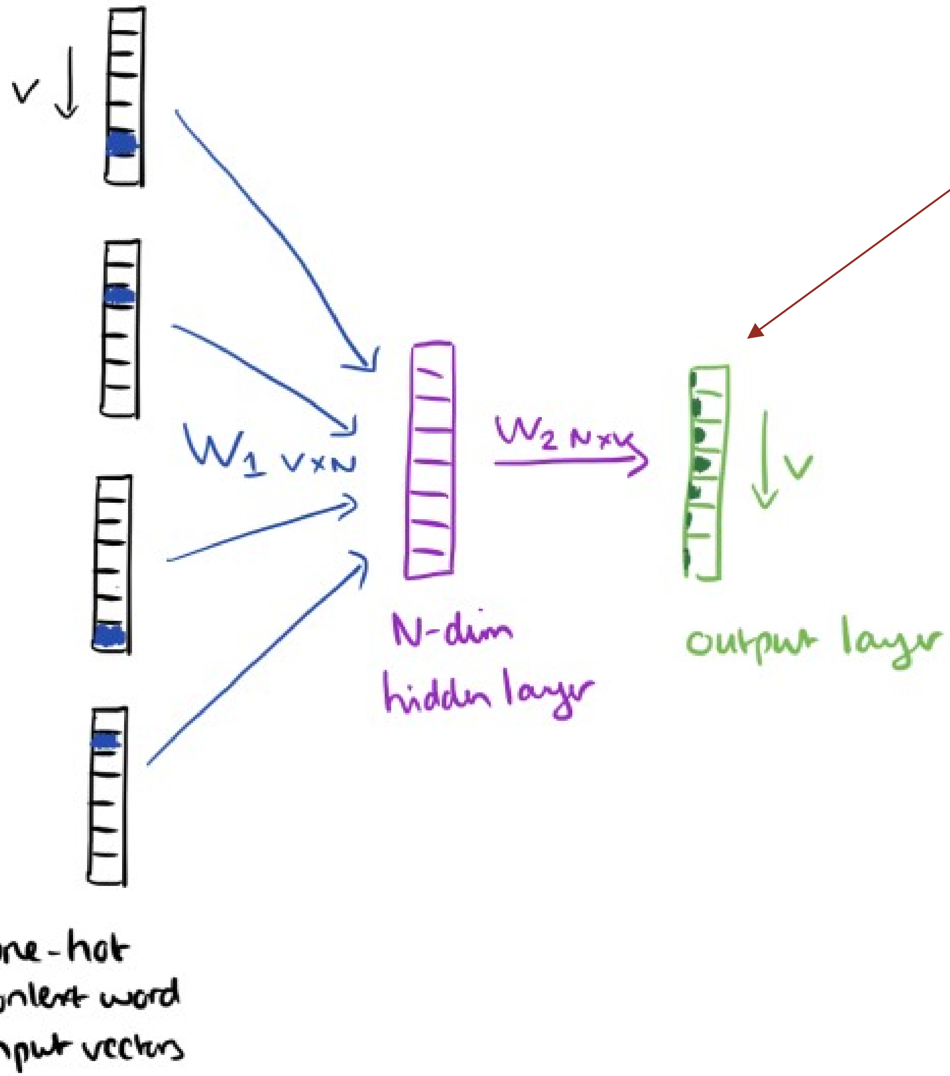
```
class CBOW(nn.Module):
    def __init__(self, vocab_size, embedding_size, context_size):
        super(CBOW, self).__init__()
        self.embeddings = nn.Embedding(vocab_size, embedding_size)
        self.lin = nn.Linear(context_size * 2 * embedding_size, vocab_size)

    def forward(self, inp):
        out = self.embeddings(inp).view(1, -1)
        out = self.lin(out)
        return F.log_softmax(out, dim=1)

    def get_word_vector(self, word_idx):
        word = torch.LongTensor([word_idx])
        return self.embeddings(word).view(1, -1)

```

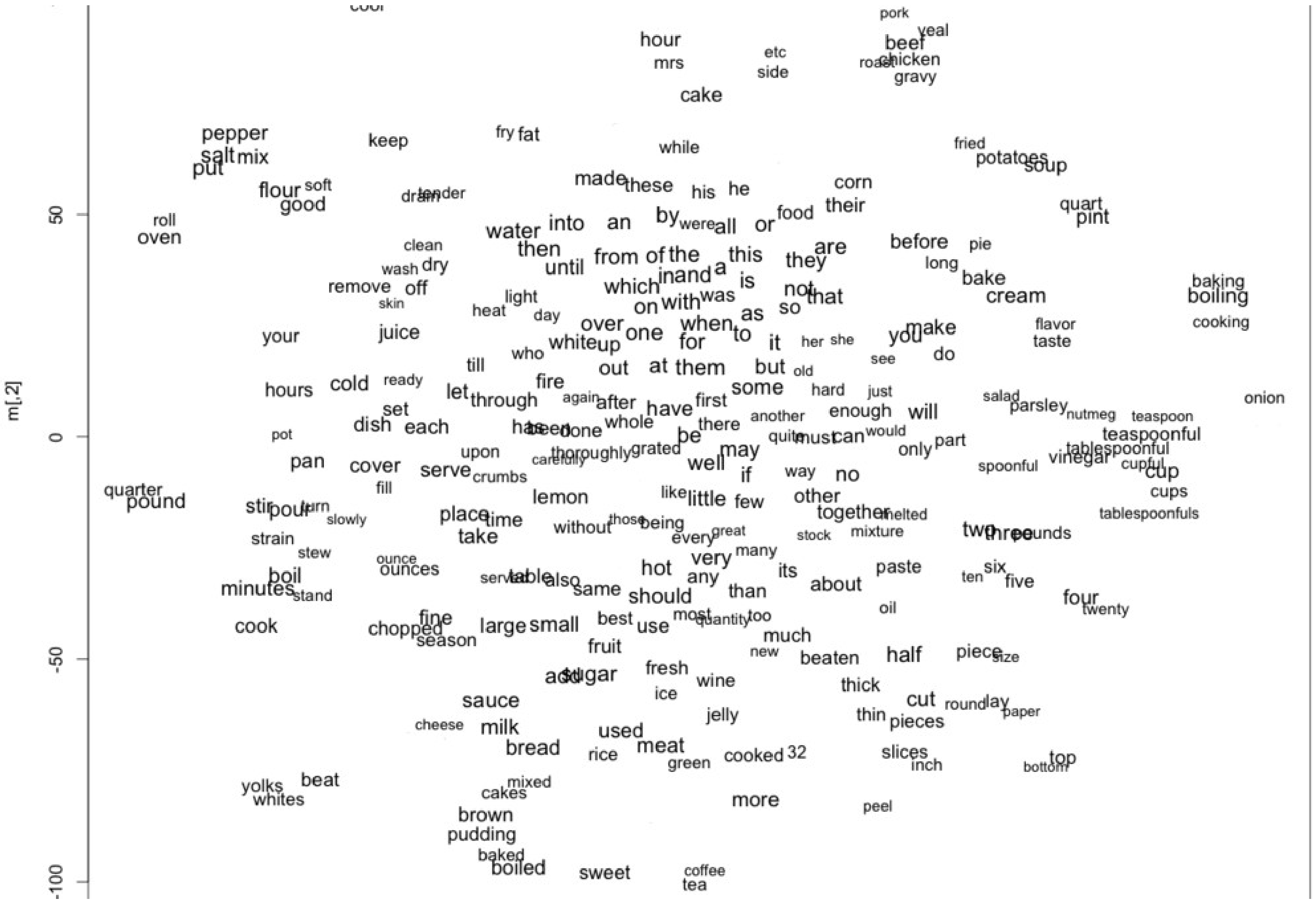
Continous Bag of Words



Probability distribution of all words in dictionary. Can be > 1 million words, so smarter training techniques are typically used:

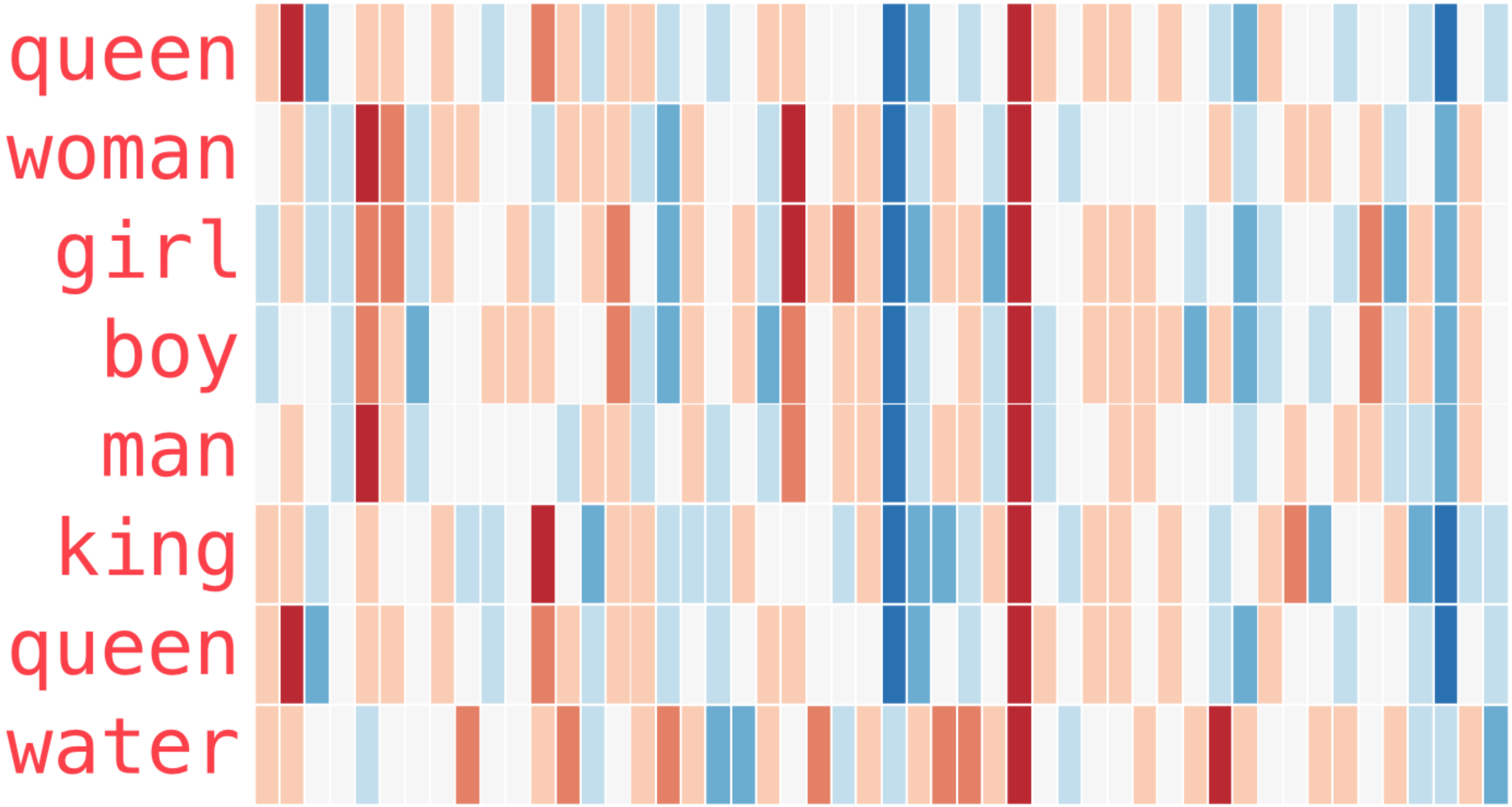
“Negative sampling”

Vectors

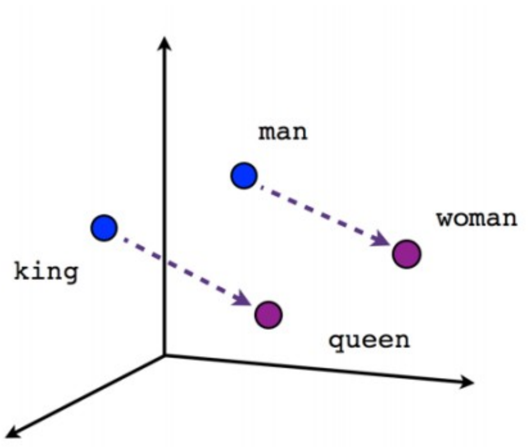


m[,2]
50
0
-50
-100
pork
veal
hour
beef
etc
mrs
chicken
side
gravy
cake
pepper
keep
fry
fat
while
fried
salt
mix
potatoes
soup
put
made
these
his
he
corn
flour
soft
tender
good
by
food
their
quart
roll
water
into
an
were
all
or
pint
oven
then
clean
before
pie
until
from
of
the
this
they
are
long
wash
dry
a
is
bake
remove
off
which
in
and
was
not
that
cream
baking
boiling
skin
light
heat
day
on
with
so
over
when
as
you
make
flavor
cooking
your
juice
white
up
one
for
to
it
her
she
taste
who
see
do
till
out
at
them
but
old
hours
cold
ready
fire
some
hard
just
let
through
again
after
have
first
enough
will
salad
parsley
onion
set
another
nutmeg
teaspoon
dish
each
whole
there
would
teaspoonful
pot
done
be
quite
must
can
only
part
tablespoonful
upon
carefully
thoroughly
grated
well
may
vinegar
cupful
pan
cover
spoonful
cup
serve
crumbs
if
way
no
fill
lemon
like
little
few
other
cups
quarter
pound
stir
pour
turn
slowly
place
time
together
melted
tablespoonfuls
without
those
being
great
stock
mixture
two
three
pounds
strain
take
every
stew
many
ounce
very
ounces
hot
its
paste
six
boil
also
any
ten
five
minutes
stand
same
should
than
about
oil
four
twenty
fine
best
use
too
cook
chopped
large
small
quantity
much
season
fruit
new
beaten
half
piece
size
fresh
sugar
wine
thick
ice
sauce
cut
round
lay
paper
jelly
thin
pieces
cheese
milk
used
bread
meat
rice
cooked
32
slices
top
green
inch
bottom
beat
mixed
yolks
cakes
whites
more
peel
brown
pudding
baked
boiled
sweet
coffee
tea

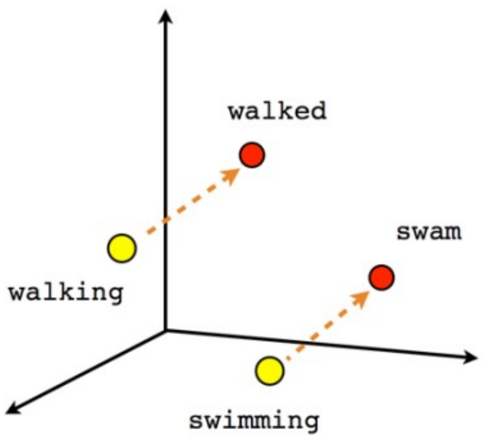
Word2Vec Vectors



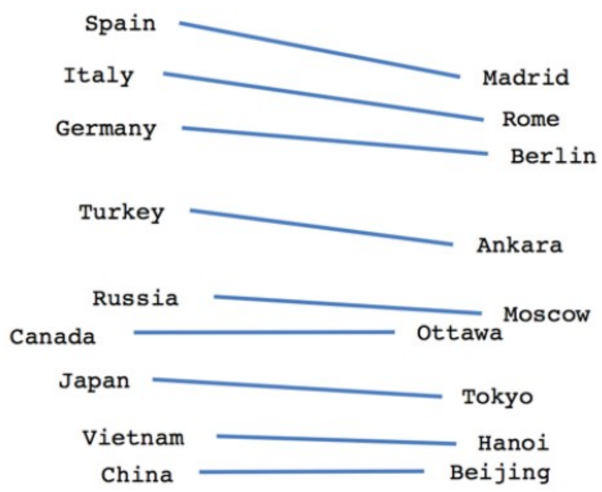
Word2Vec Vectors



Male-Female



Verb tense



Country-Capital

King – Man + Woman = Queen

Representing sentences

Using word embeddings sentences become “pictures”:

$$\text{“I think therefore I am”} = \begin{pmatrix} 1.2 & -0.5 & \cdots \\ -0.1 & 2.3 & \cdots \\ 1.0 & 1.1 & \cdots \\ 1.2 & -0.5 & \cdots \\ -1.1 & 0.1 & \cdots \end{pmatrix}$$

5 x 200 matrix

Representing sentences

If you have a enough data, word embeddings
can simply be trained as part of your network,
but typically it is better to use pretrained embeddings.

Pretrained word vectors

- Glove: https://nlp.stanford.edu/projects/glove/
- FastText: https://fasttext.cc/docs/en/crawl-vectors.html
- ELMo: https://github.com/HIT-SCIR/ELMoForManyLangs

Trained on Wikipedia and “common crawl”

Can be used as-is or further trained on specific corpus

Natural Language Processing

Exercise: Guess the rating from review text

[redacted]

The greatest thing put on film

tomwolf13 4 December 2008

The day 'The Wire' ended was a sad day to me. Having to see some of my favourite characters in any medium (novels, TV, movies, etc.) for the last time felt like saying goodbye to my friends. Knowing that I will never be so involved in a series ever again is saddening. At the same time, however, I'm proud that 'The Wire' was taken off the air before it could have been potentially bastardized like many series before it.

This show is a pinnacle in entertainment, and though never acclaimed with awards as it should have been, will go down as perhaps the greatest television series in history...and perhaps the greatest thing ever put to film. Literally, perfect.

Natural Language Processing

Exercise: Guess the rating from review text

★ 10/10

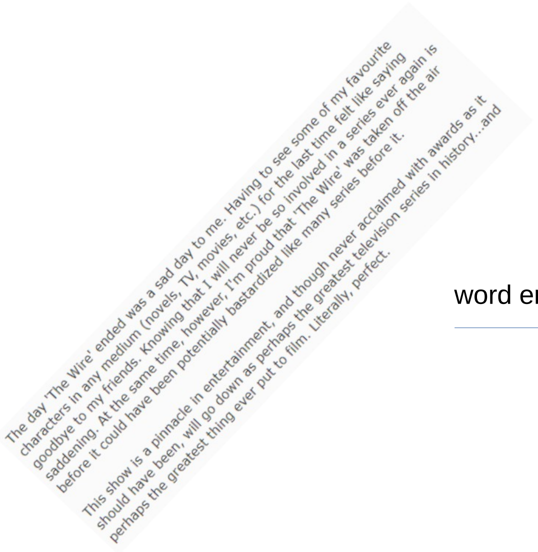
The greatest thing put on film

tomwolf13 4 December 2008

The day 'The Wire' ended was a sad day to me. Having to see some of my favourite characters in any medium (novels, TV, movies, etc.) for the last time felt like saying goodbye to my friends. Knowing that I will never be so involved in a series ever again is saddening. At the same time, however, I'm proud that 'The Wire' was taken off the air before it could have been potentially bastardized like many series before it.

This show is a pinnacle in entertainment, and though never acclaimed with awards as it should have been, will go down as perhaps the greatest television series in history...and perhaps the greatest thing ever put to film. Literally, perfect.

Natural Language Processing


The day 'The Wire' ended was a sad day to me. Having to see some of my favourite characters in any medium (novels, TV, movies, etc.) for the last time felt like saying goodbye to my friends. Knowing that I will never be so involved in a series ever again is saddening. At the same time, however, I'm proud that 'The Wire' was taken off the air before it could have been potentially bastardized like many series before it.

This show is a pinnacle in entertainment, and though never acclaimed with awards as it should have been, will go down as perhaps the greatest television series in history...and perhaps the greatest thing ever put to film. Literally, perfect.

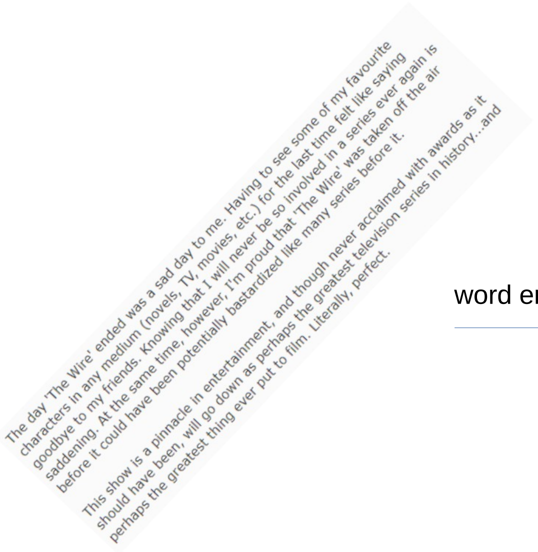
word embedding

$$= \begin{pmatrix} 1.2 & -0.5 & \cdots \\ -0.1 & 2.3 & \cdots \\ 1.0 & 1.1 & \cdots \\ 1.2 & -0.5 & \cdots \\ -1.1 & 0.1 & \cdots \end{pmatrix}$$

#words × embedding dim

neural network

Natural Language Processing



word embedding

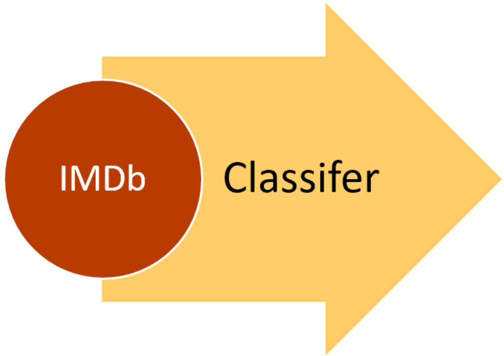
$$= \begin{pmatrix} 1.2 & -0.5 & \cdots \\ -0.1 & 2.3 & \cdots \\ 1.0 & 1.1 & \cdots \\ 1.2 & -0.5 & \cdots \\ -1.1 & 0.1 & \cdots \end{pmatrix}$$

#words × embedding dim

neural network

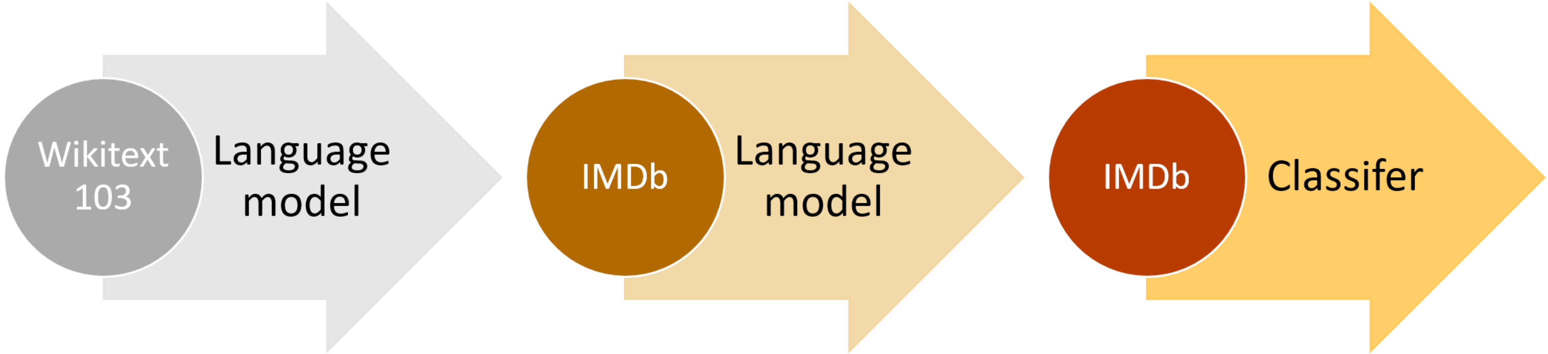
(actually, you will just be predicting *positive* or *negative*)

Transfer learning



$$P(\text{rating}) = f(x; h; p)$$

Transfer learning



$$\mathbf{h}_1 = \text{LSTM}(\text{"the"}, 0) \quad (1)$$
$$\mathbf{h}_2 = \text{LSTM}(\text{"pinnacle"}, \mathbf{h}_1) \quad (2)$$
$$\mathbf{h}_3 = \text{LSTM}(\text{"of"}, \mathbf{h}_2) \quad (3)$$
$$\mathbf{h}_4 = \text{LSTM}(\text{"entertainment"}, \mathbf{h}_3) \quad (4)$$
$$\mathbf{p}_{\text{word}} = \mathcal{W}_w \mathbf{h}_4 \quad (5)$$
$$\mathbf{p}_{\text{rating}} = \mathcal{W}_r \mathbf{h}_4 \quad (6)$$

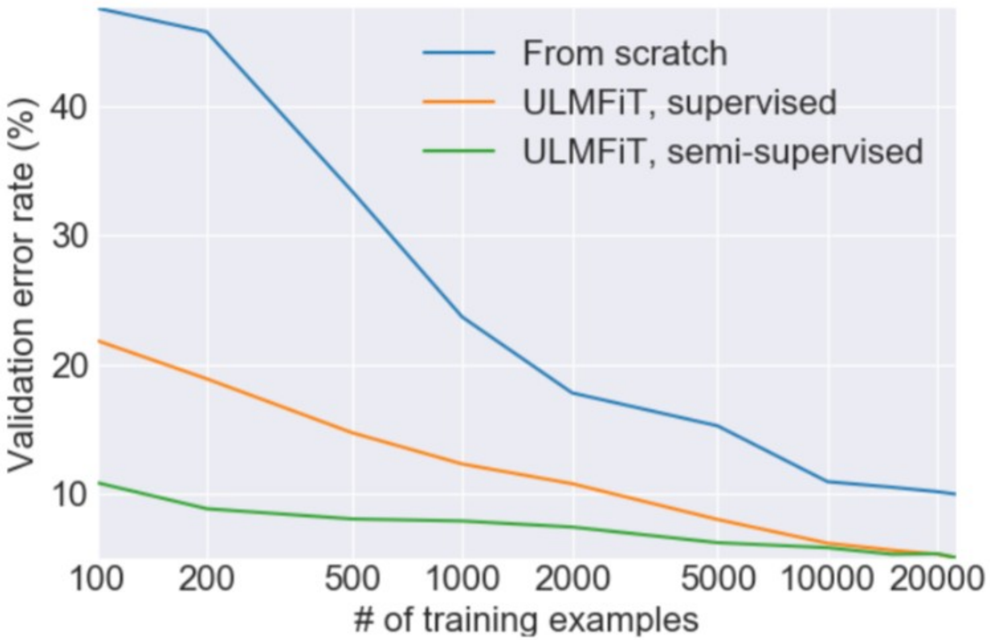
The Strength of Transfer learning

IMDB: What if only 1 % of reviews included a rating?
can the remaining 99 % reviews be used for anything?

Language model!

(and this is very, very standard situation, in academia and industry)

The Strength of Transfer learning



“… we found that training our approach with only 100 labeled examples (and giving it access to about 50,000 unlabeled examples), we were able to achieve the same performance as training a model from scratch with 10,000 labeled examples.“
- Howard & Ruder (2018)

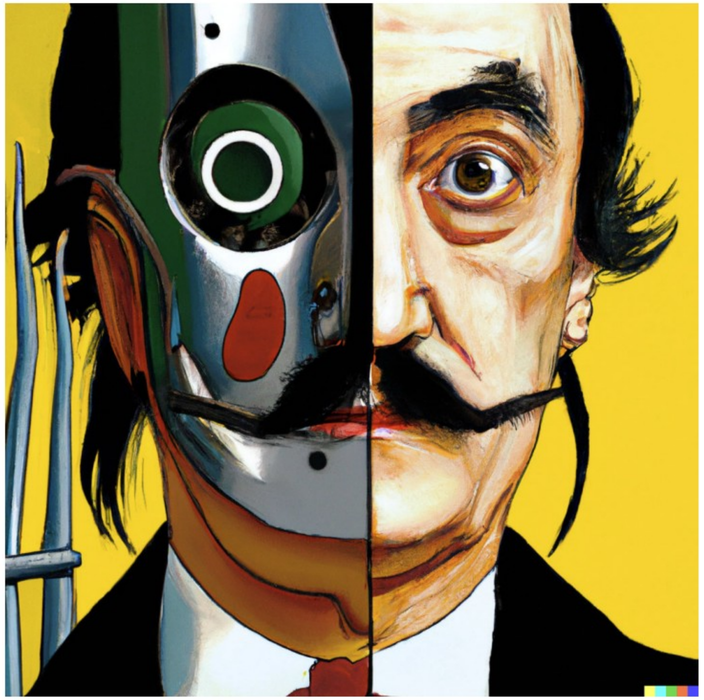
The Strength of Language Models

GPT-3

https://beta.openai.com/

DALL-E

Combining NLP and computer vision



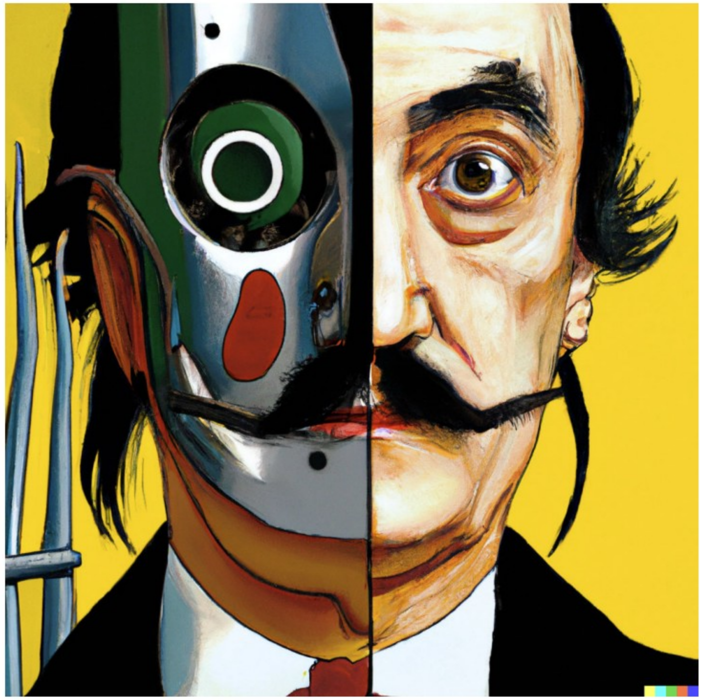
vibrant portrait painting of Salvador Dalí with a robotic half face

a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese

an espresso machine that makes coffee from human souls, artstation

DALL-E

Combining NLP and computer vision



vibrant portrait painting of Salvador Dalí with a robotic half face

a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese

an espresso machine that makes coffee from human souls, artstation

Concepts skipped

- Encoder-Decoders (sequence to sequence)
- Attention
- Transformers

See e.g. paper: “**Attention Is All You Need” (2017)**