CS 466/566
Introduction to Deep Learning

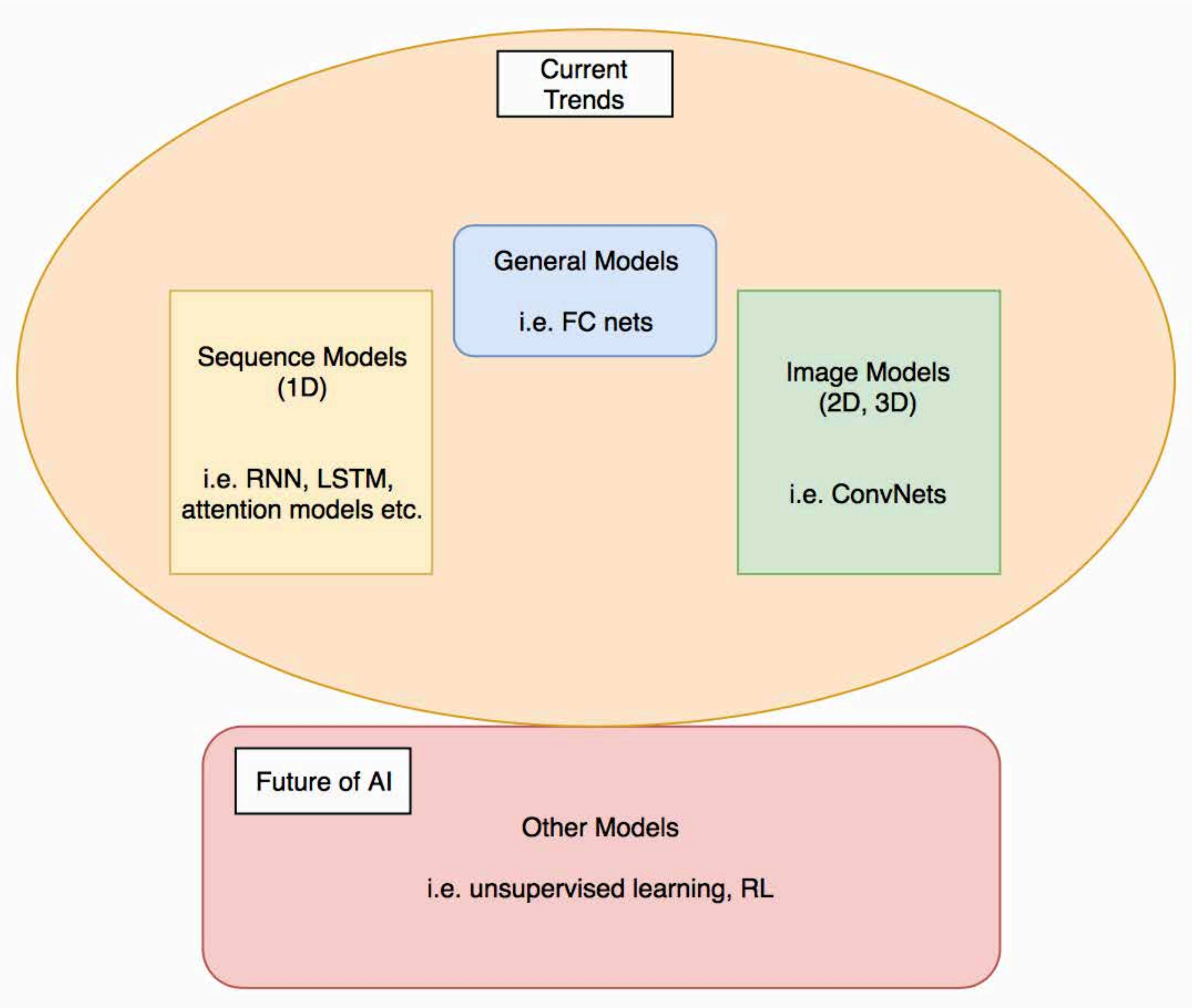
Lecture 11 – Recurrent Neural Networks – Part 1

Compiled from various resources

Recall: ConvNet Demo

http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

What were we doing? Where are we?



Sequence Modelling

- Humans don’t start their thinking from scratch every now and then.
  - You understand each word based on your understanding of previous words.
  - We don’t throw everything away and restart thinking from scratch. Thoughts have persistence.
- Traditional neural networks approach can’t do this, and it is a major shortcoming.
  - Imagine classifying what kind of event is happening at every point in a movie.
  - It’s unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones.

Borrowed from Christopher Olah

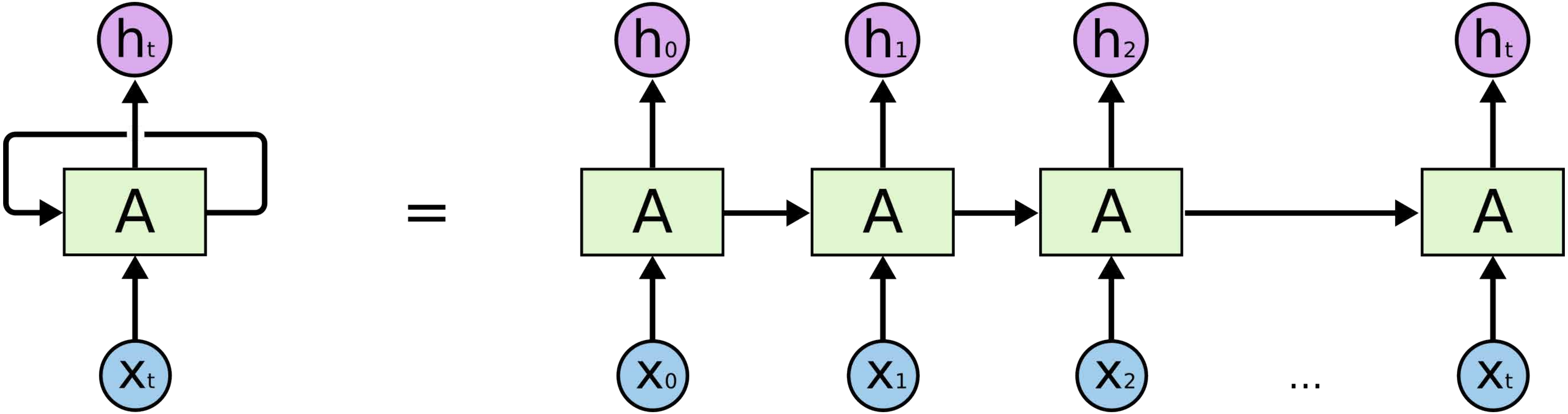
Sequence Modelling

- Humans don’t start their thinking from scratch every now and then.
  - You understand each word based on your understanding of previous words.
  - We don’t throw everything away and restart thinking from scratch. Thoughts have persistence.
- Traditional neural networks approach can’t do this, and it is a major shortcoming.
  - Imagine classifying what kind of event is happening at every point in a movie.
  - It’s unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones.

Borrowed from Christopher Olah

Recurrent Neural Networks (RNNs)

- RNNs solve this issue. They are networks with loops in them, allowing information to persist.



Borrowed from Christopher Olah

Recurrent Neural Networks (RNNs)

- Limitation of Traditional NNs (including CNNs) is that they are too constrained:
  - accept a fixed-sized vector as input (e.g. an image)
  - produce a fixed-sized vector as output (e.g. probabilities of different classes).
  - perform this mapping by using a fixed amount of computational steps (e.g. the number of layers in the model).
- The core reason that recurrent nets are more exciting is that they allow us to operate over *sequences* of vectors: Sequences in the input, the output, or in the most general case both.

Borrowed from Andrej Karpathy

Recurrent Neural Networks (RNNs)

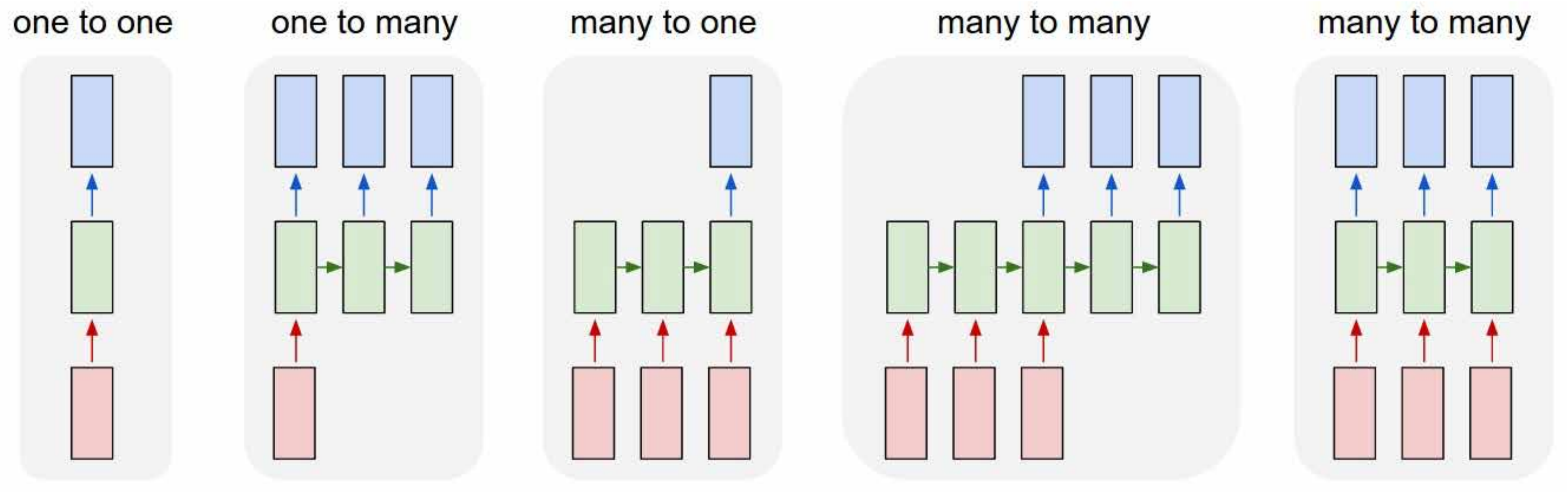


Image Classification | Image Captioning | Sentiment Analysis | Google Translate | Video Frame Classification

red: input, blue: output, green: RNN state

Borrowed from Andrej Karpathy

Recurrent Neural Networks (RNNs)

- The sequential operation is much more powerful compared to fixed networks.
- Moreover, RNNs combine the input vector with their state vector with a fixed (but learned) function to produce a new state vector.
- This can in programming terms be interpreted as running a fixed program with certain inputs and some internal variables.
- Viewed this way, RNNs are essentially similar to programs. They can theoretically simulate programs.

Borrowed from Andrej Karpathy

Recurrent Neural Networks (RNNs)

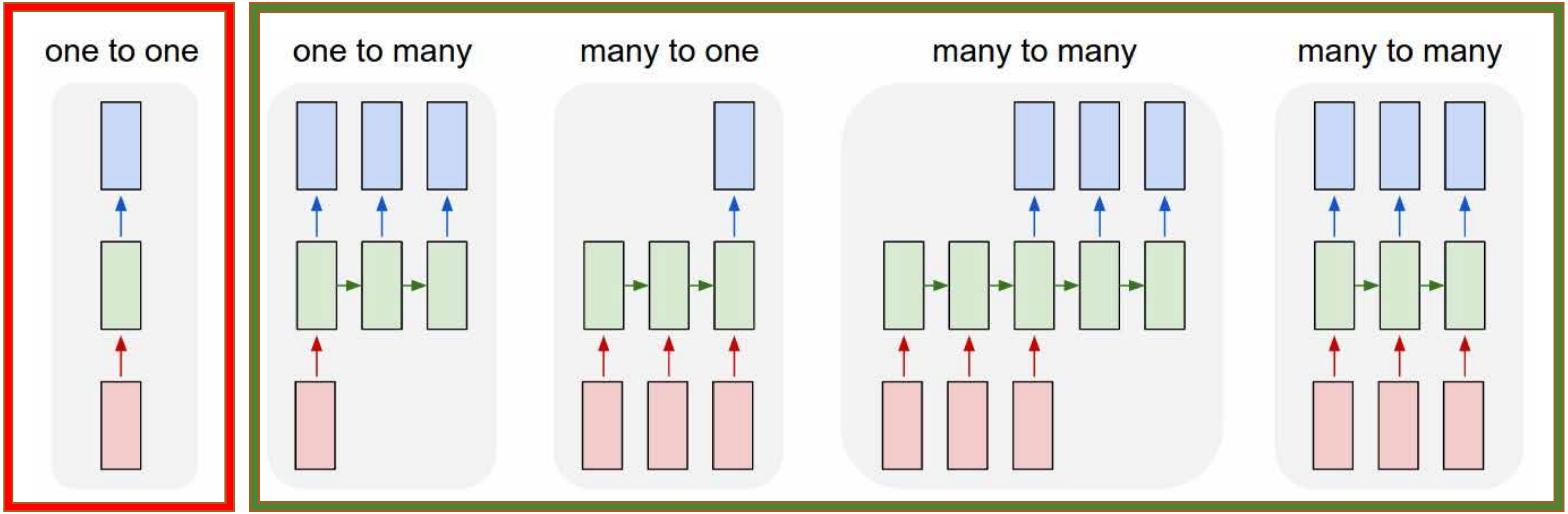
If training vanilla neural nets is optimization over functions, training recurrent nets is optimization over programs.

Borrowed from Andrej Karpathy

Recurrent Neural Networks (RNNs)

Optimization over functions

Optimization over programs



Borrowed from Andrej Karpathy

Recurrent Neural Network Handwriting Generation Demo

- http://www.cs.toronto.edu/~graves/handwriting.html

Four effective ways to learn an RNN

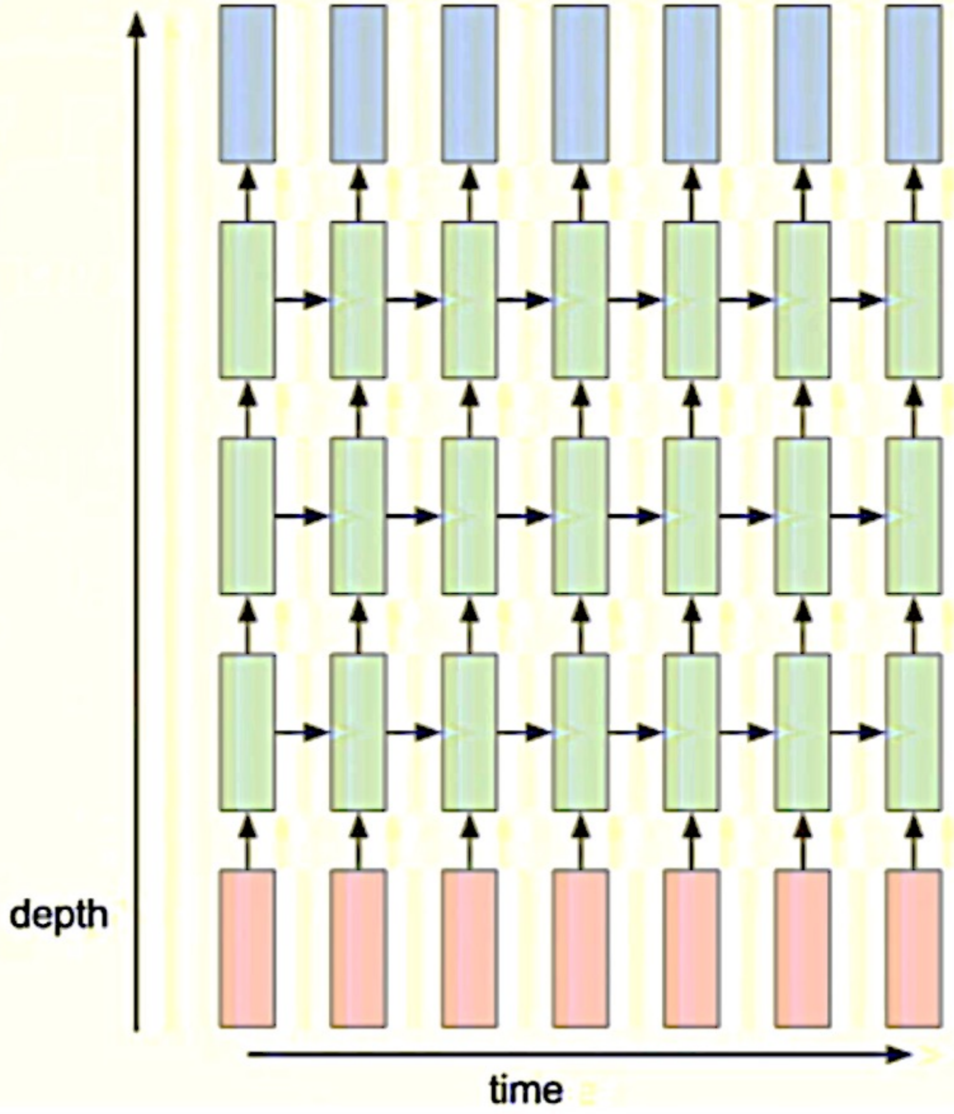
- **Long Short Term Memory (LSTM)**
  Make the RNN out of little modules that are designed to remember values for a long time.
- Hessian Free Optimization:
  Deal with the vanishing gradients problem by using a fancy optimizer that can detect directions with a tiny gradient but even smaller curvature.
  - The HF optimizer (Martens & Sutskever, 2011) is good at this.
- Echo State Networks:
  Initialize the input→hidden and hidden→hidden and output→hidden connections very carefully so that the hidden state has a huge reservoir of weakly coupled oscillators which can be selectively driven by the input.
  - ESNs only need to learn the hidden→output connections.
- Good initialization with momentum
  Initialize like in Echo State Networks, but then learn all of the connections using momentum.

Deep Recurrent Neural Networks (RNNs)

RNN:

$$\boxed{h_t^l = \tanh W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}}$$

$h \in \mathbb{R}^n$ $\quad W^l \; [n \times 2n]$



Borrowed from Andrej Karpathy

Deep Recurrent Neural Networks (RNNs)

RNN:

$$\boxed{h_t^l = \tanh W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}}$$

$h \in \mathbb{R}^n$ $\quad W^l \ [n \times 2n]$

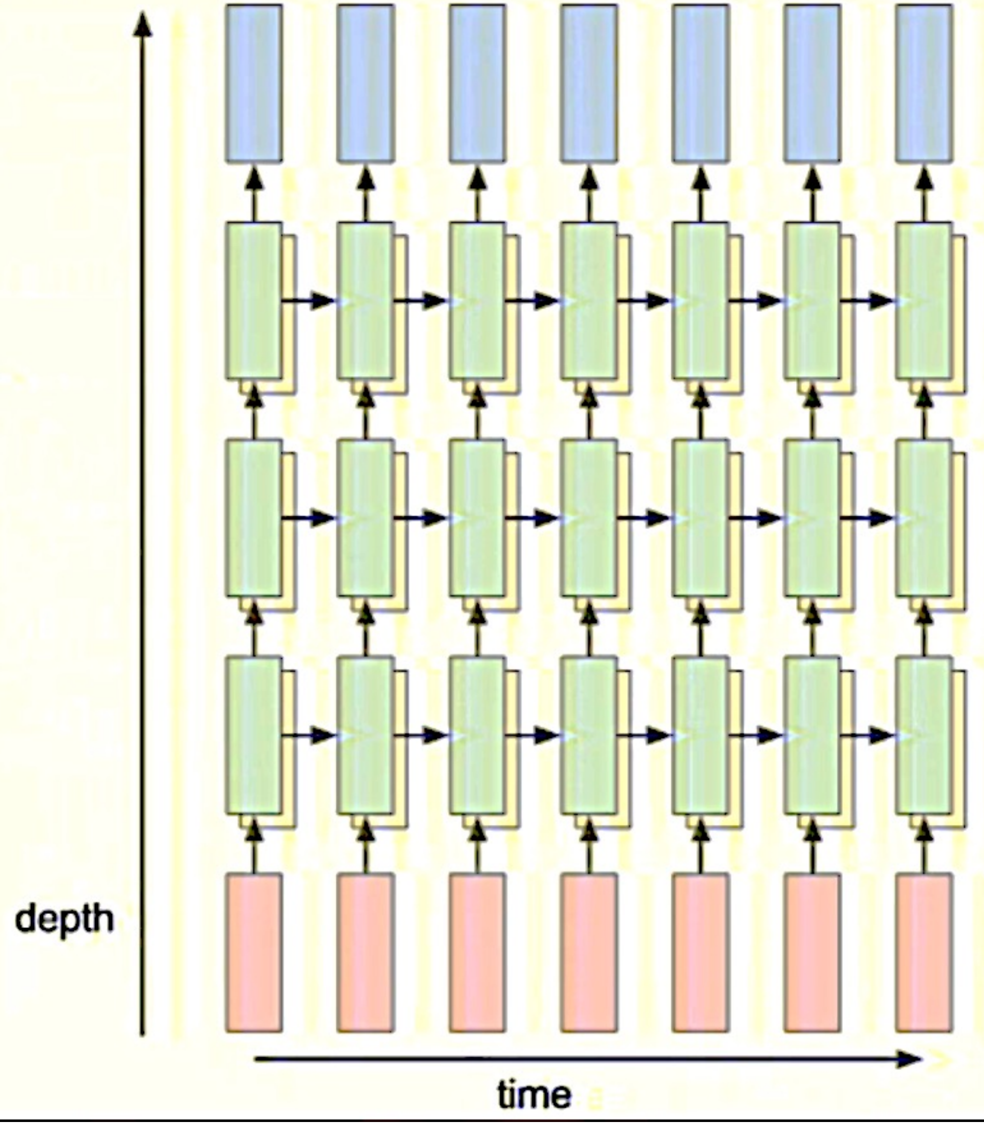
LSTM:

$W^l \ [4n \times 2n]$

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \mathrm{sigm} \\ \mathrm{sigm} \\ \mathrm{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

$$h_t^l = o \odot \tanh(c_t^l)$$



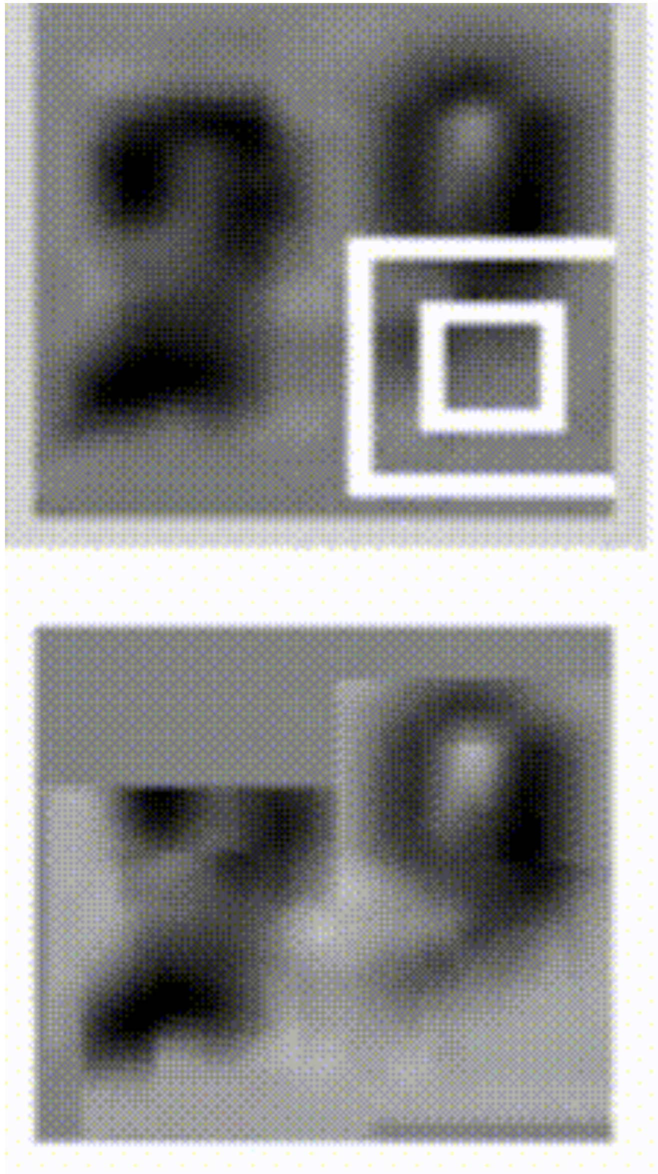
Borrowed from Andrej Karpathy

Finding sequence in absence of sequence

- Even if your data is not in form of sequences, you can still formulate and train powerful models that learn to process it sequentially.
- You’re actually learning stateful programs that process your fixed-sized data.

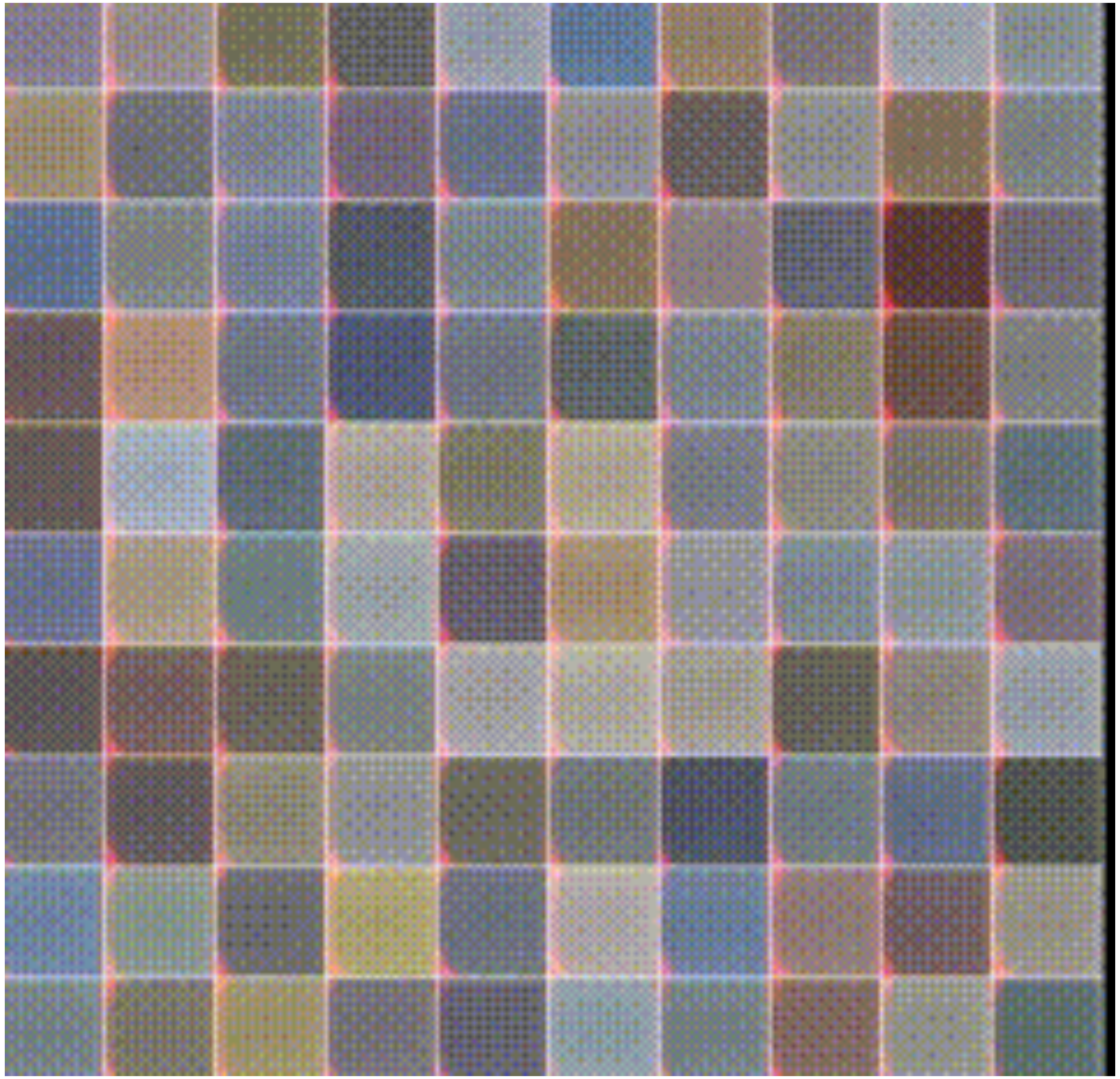
Borrowed from Andrej Karpathy

Finding sequence in absence of sequence



Borrowed from Andrej Karpathy

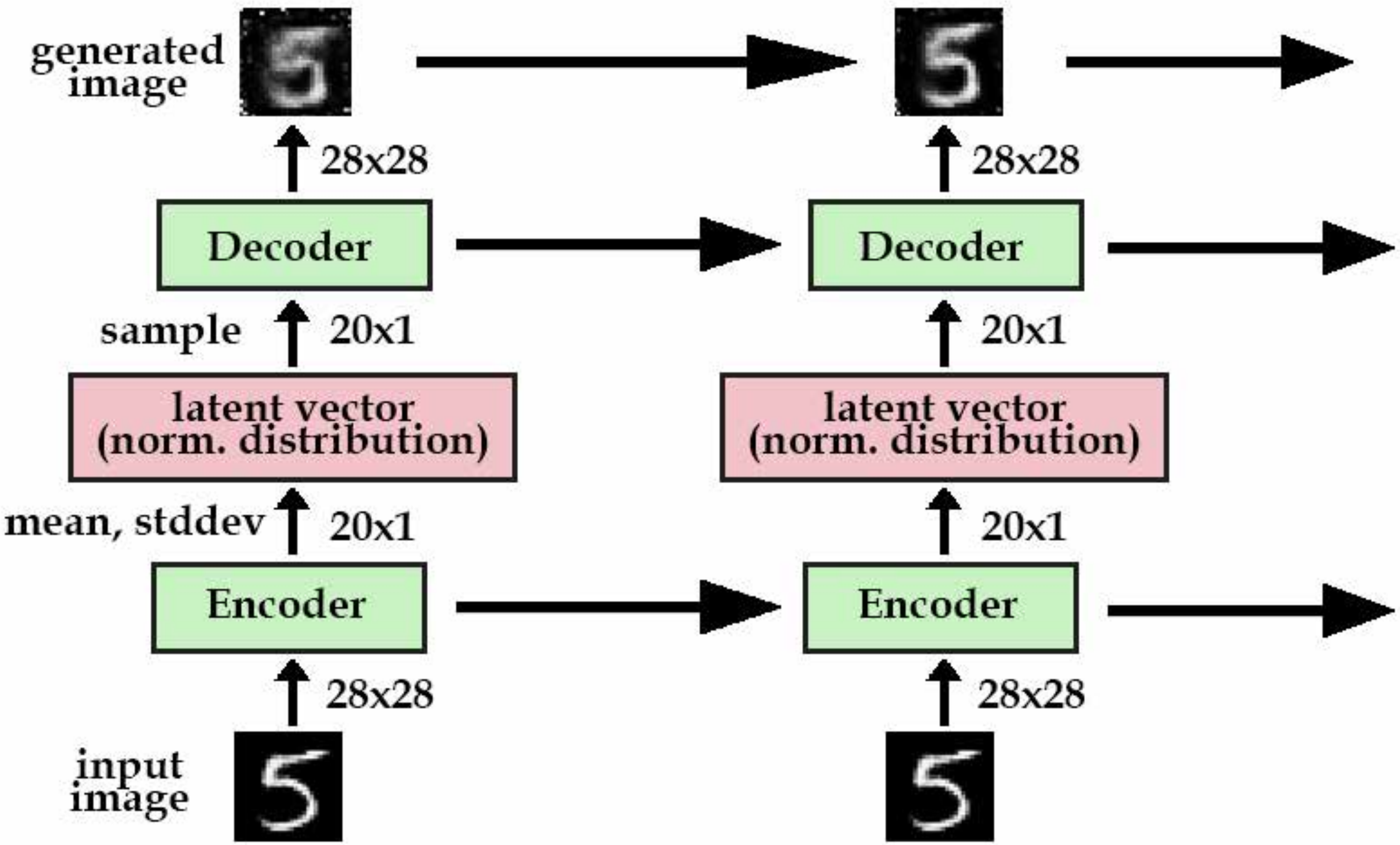
Finding sequence in absence of sequence



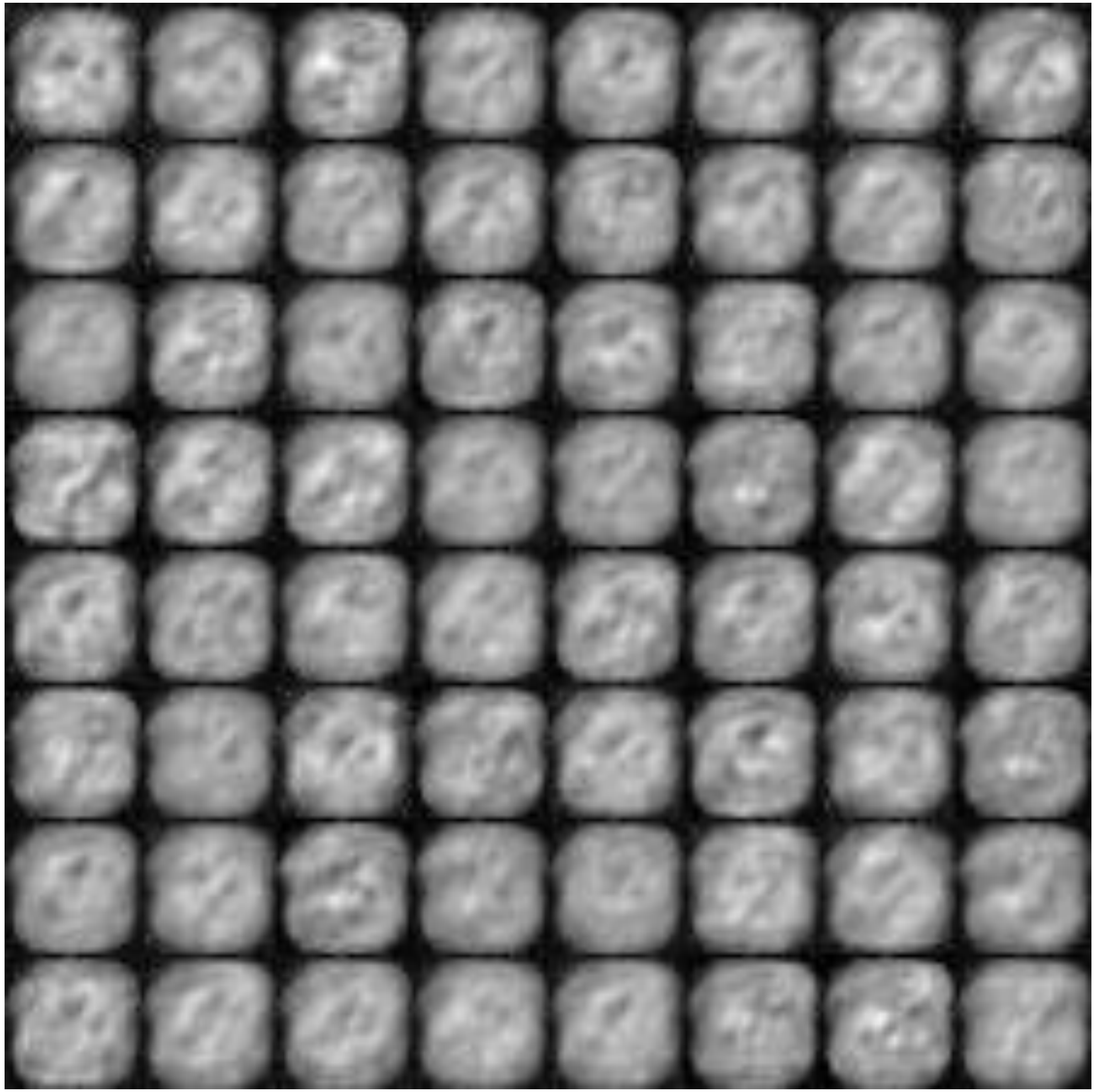
Borrowed from Andrej Karpathy

Teaser #1: DRAW: Deep Recurrent Attentive Writer

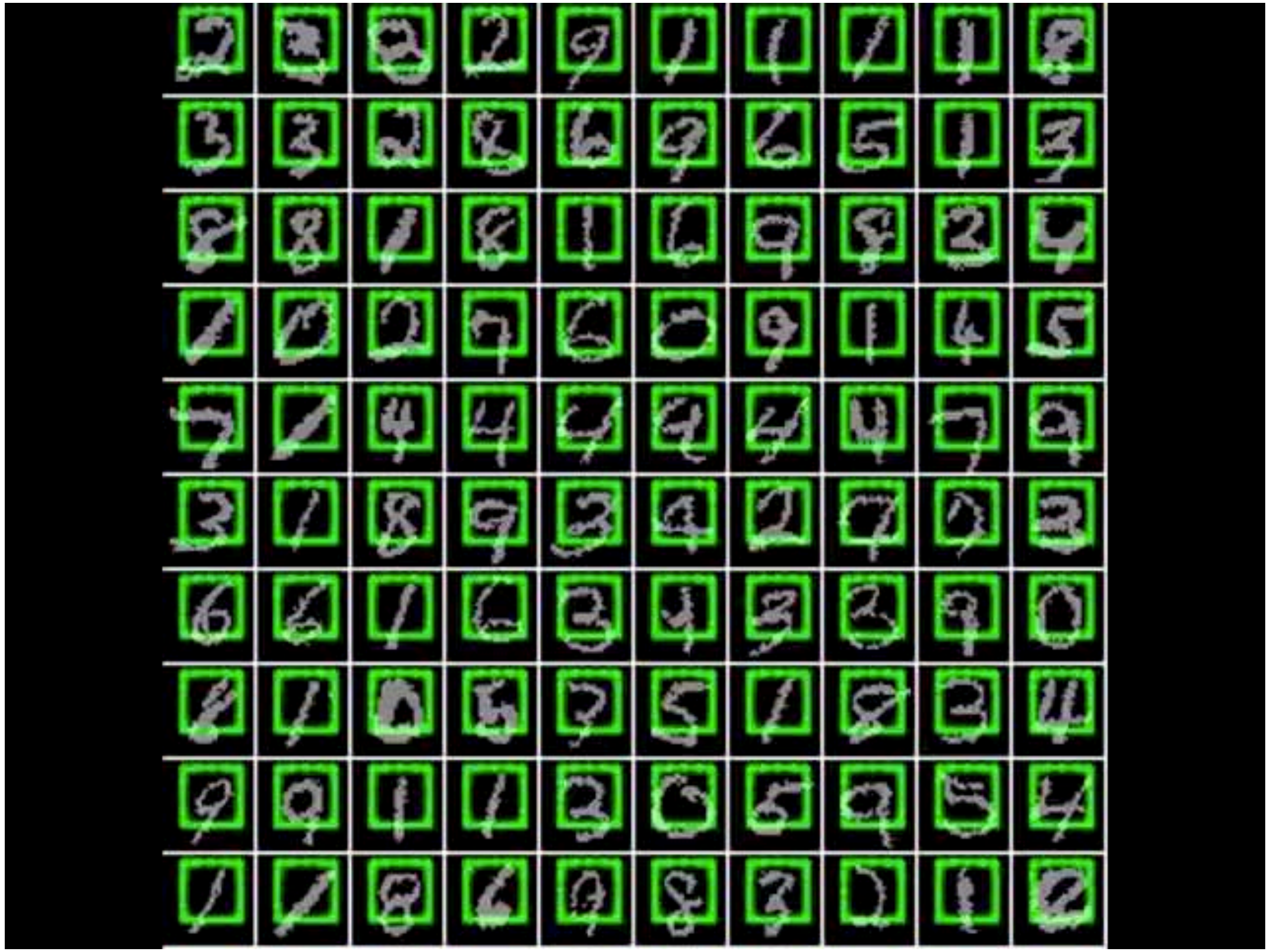
- It is a recurrent version of Variational Autoencoder!



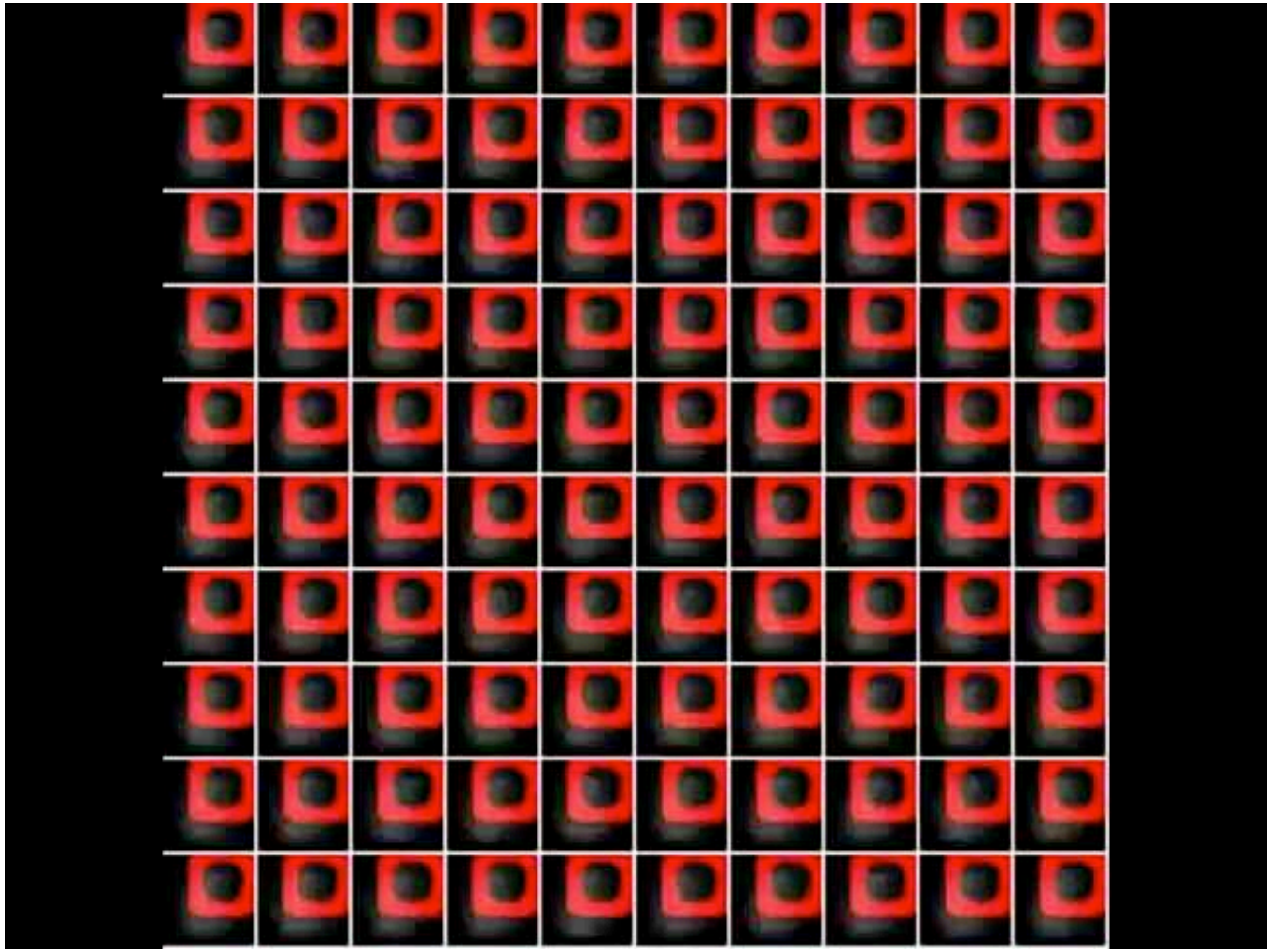
DRAW: Deep Recurrent Attentive Writer



DRAW: Deep Recurrent Attentive Writer
Attention Gate and Encoder



DRAW: Deep Recurrent Attentive Writer
Attention Gate and Decoder

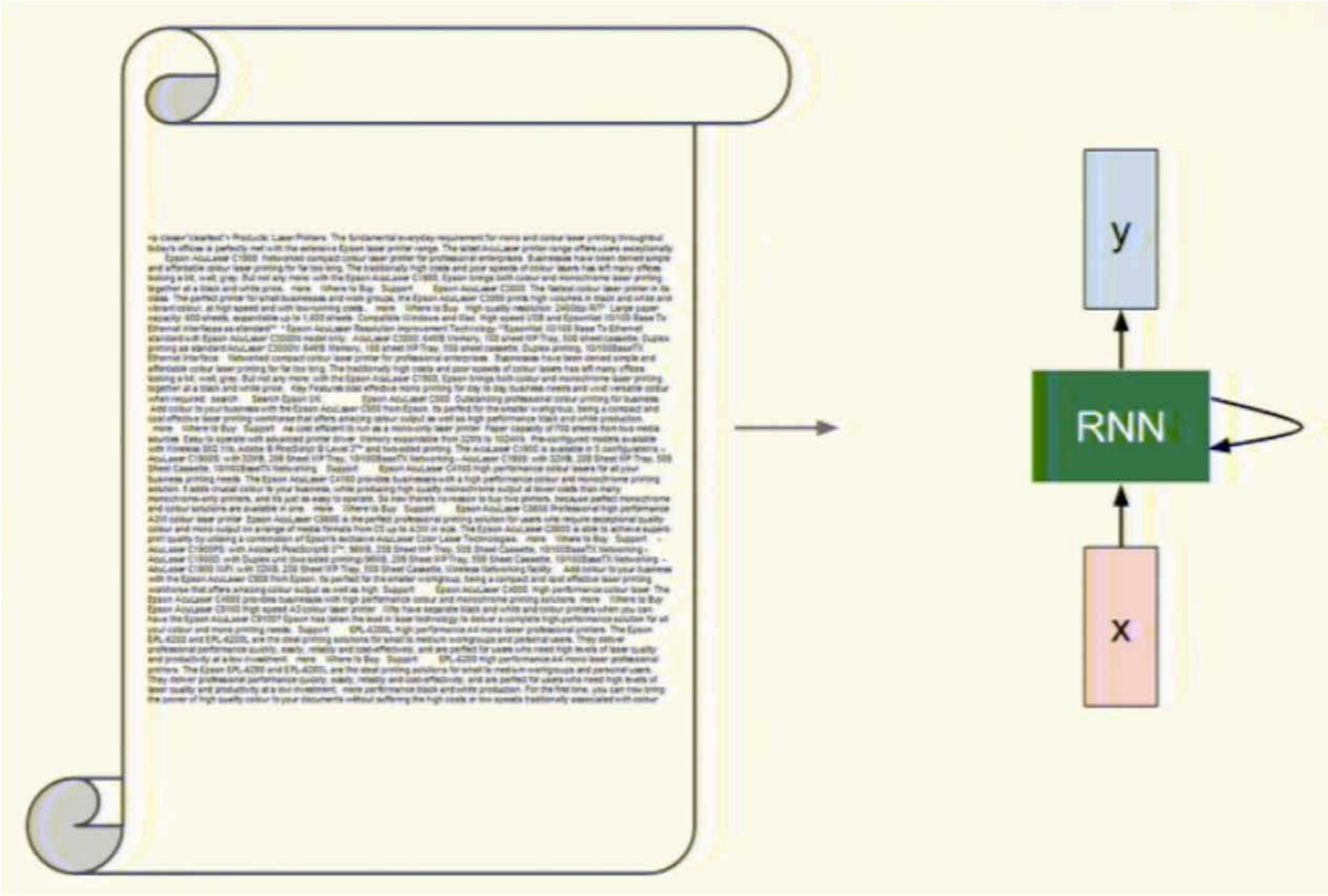


Teaser #2: Fun example: Character Level Networks

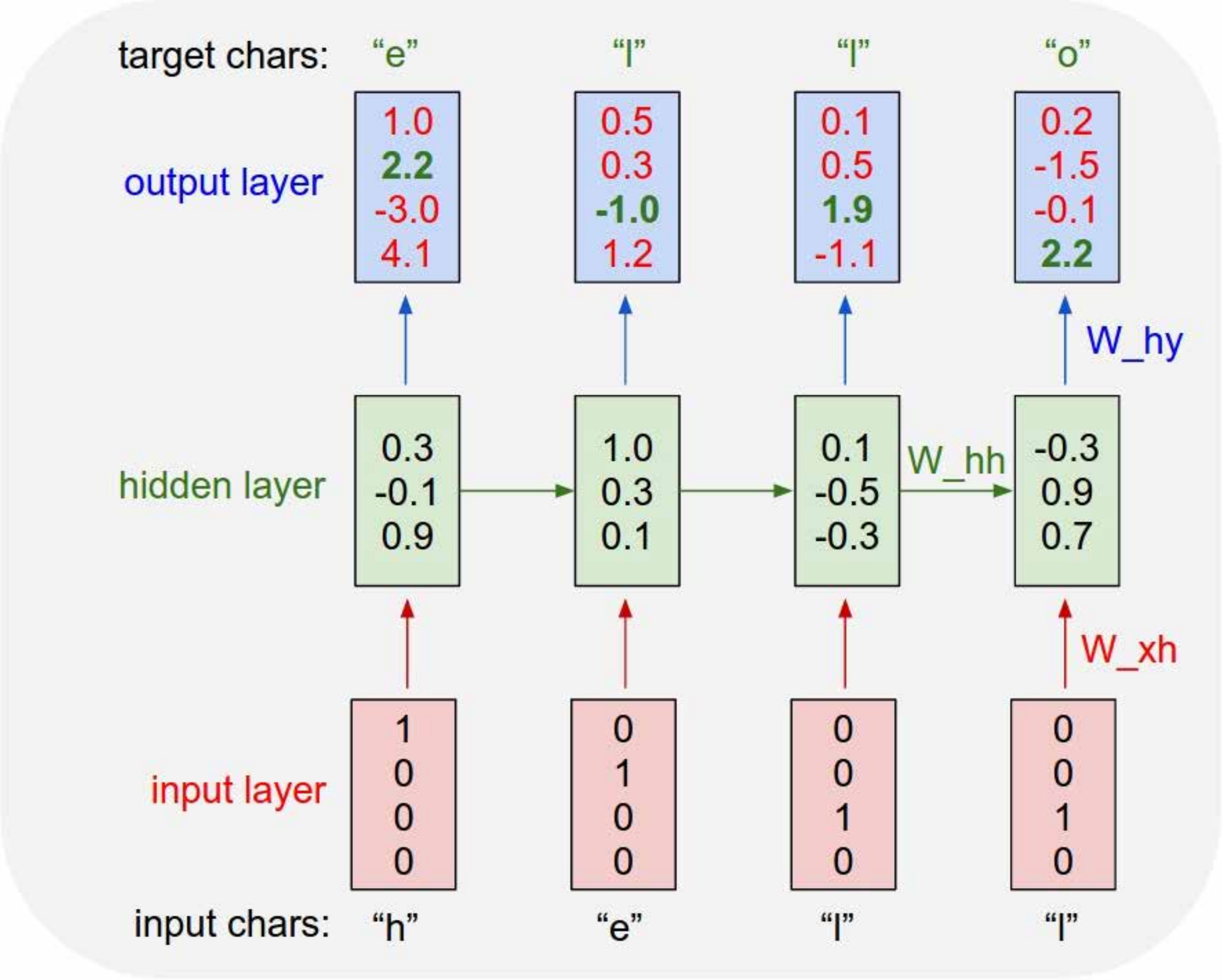
- give the RNN a huge chunk of text and ask it to model the probability distribution of the next character in the sequence given a sequence of previous characters.
- generate new text one character at a time.
- assume we have a vocabulary of 4 possible letters: “helo”
- Now, train an RNN on a sequence like: “hello”
- For each letter, the context will be the letters before it.

Borrowed from Andrej Karpathy

Fun example: Character Level Networks



Borrowed from Andrej Karpathy



We want:

Green numbers to be **high**

Red numbers to be **low**

We use the standard Softmax classifier (cross-entropy loss) on every output vector simultaneously.

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

An example RNN with 4-dimensional input and output layers, and a hidden layer of 3 units (neurons)

Borrowed from Andrej Karpathy

Evolution of Samples during Training

- At iteration 100 the model samples random jumbles:

```
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
```

- At 300 iterations the model starts to get an idea about quotes and periods:

```
"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."
```

- At 500 iterations the words are now also separated with spaces and the model starts to get the idea about periods at the end of a sentence.

```
we counter. He stutn co des. His stanted out one ofler that concossions and was
to gearang reay Jotrets and with fre colt otf paitt thin wall. Which das stimn
```

Borrowed from Andrej Karpathy

Evolution of Samples during Training

- At iteration 700 we're starting to see more and more English-like text emerge:

```
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.
```

- At iteration 1200 we're now seeing use of quotations and question/exclamation marks. Longer words have now been learned as well:

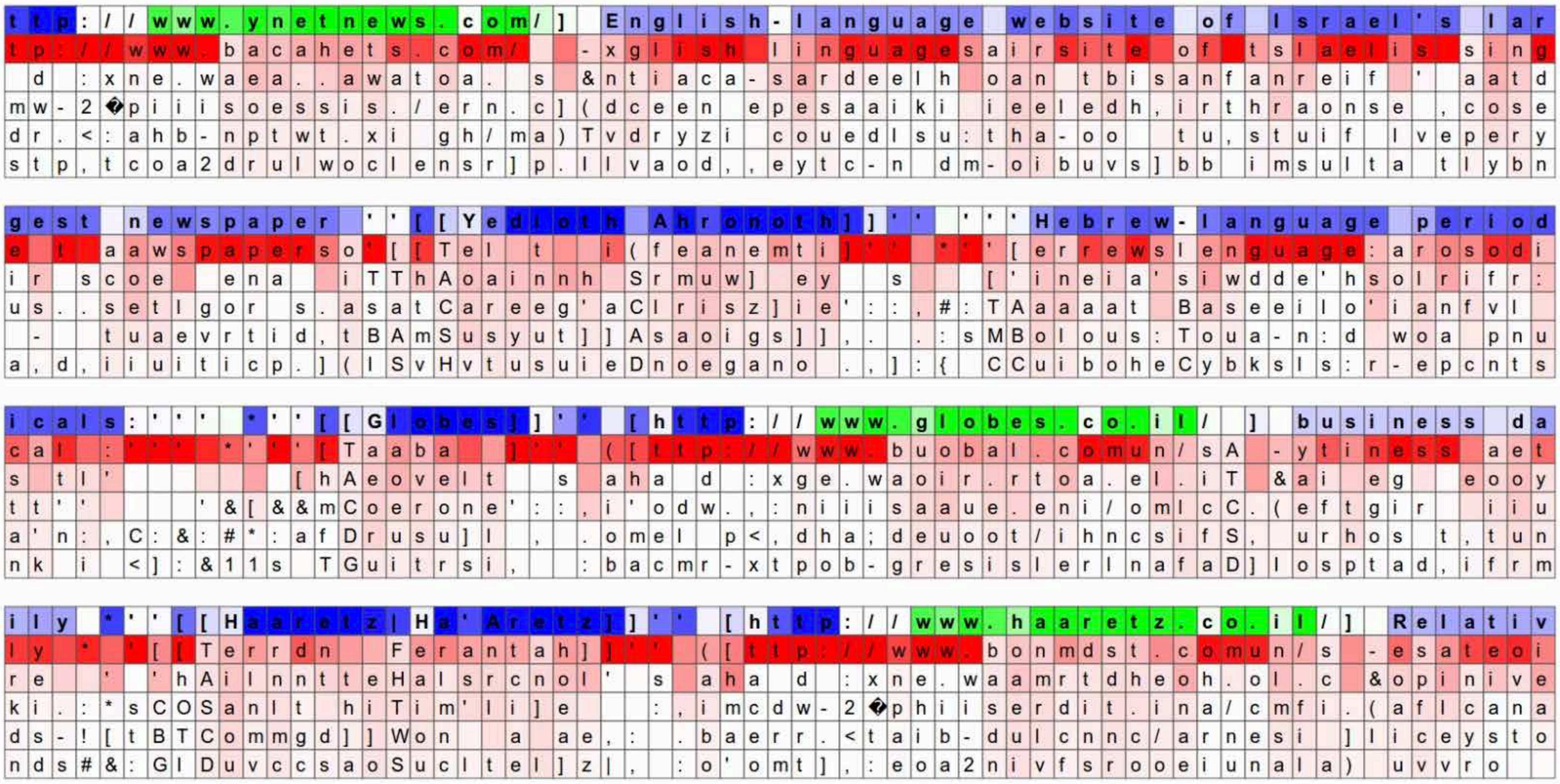
```
"Kite vouch!" he repeated by her
door. "But I would be done and quarts, feeling, then, son is people...."
```

- we start to get properly spelled words, quotations, names, and so on by about iteration 2000:

```
"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.
```

Borrowed from Andrej Karpathy

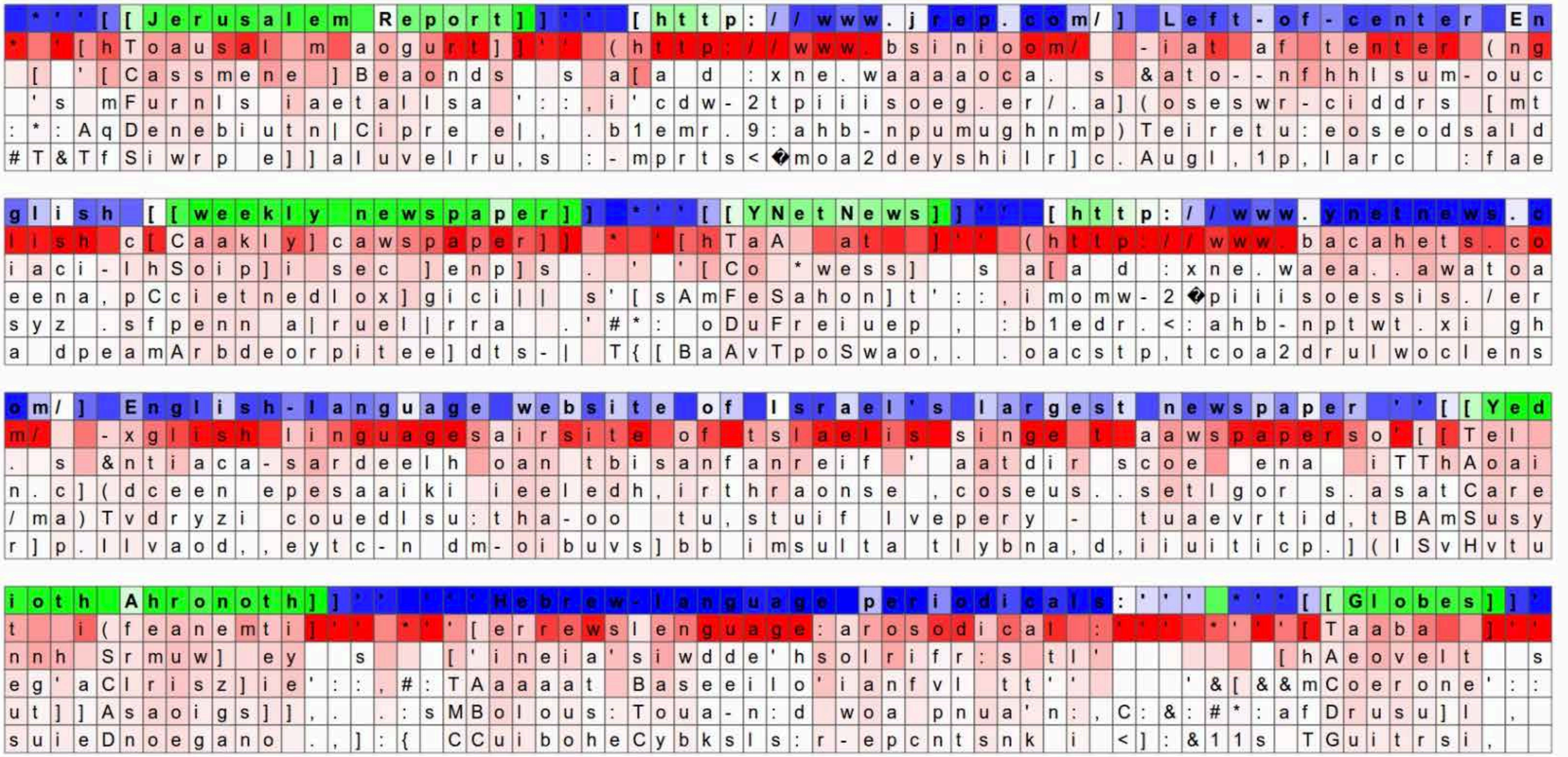
Visualizing predictions and “neuron” firings in RNN



The neuron highlighted in this image seems to get very excited about URLs and turns off outside of the URLs. The LSTM is likely using this neuron to remember if it is inside a URL or not.

Borrowed from Andrej Karpathy

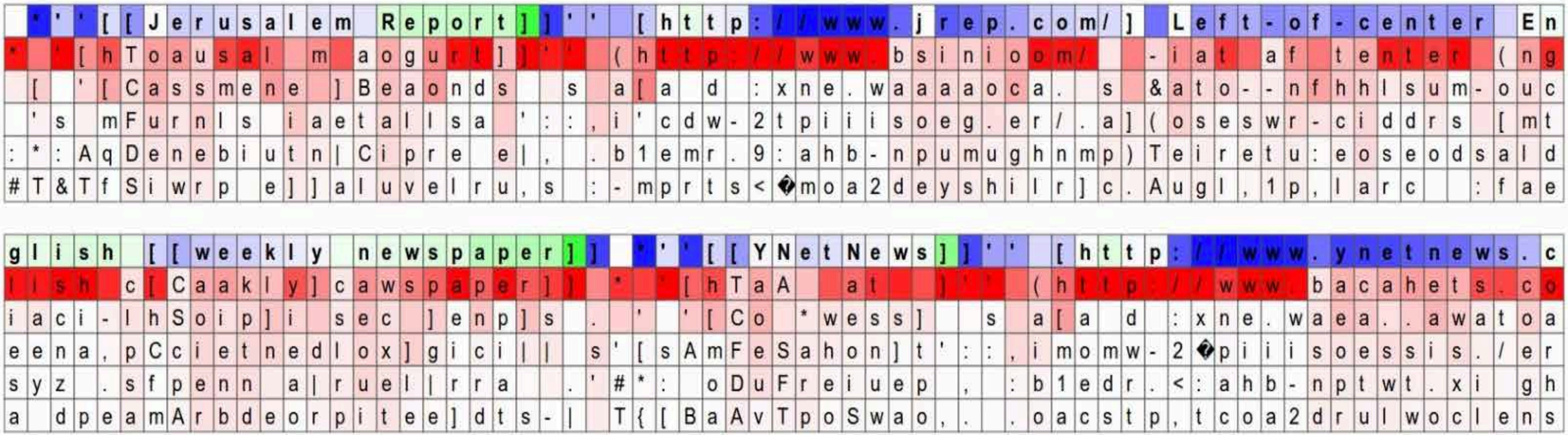
Visualizing predictions and “neuron” firings in RNN



The highlighted neuron here gets very excited when the RNN is inside the [[ ]] markdown environment and turns off outside of it. Interestingly, the neuron can't turn on right after it sees the character "[", it must wait for the second "[" and then activate. This task of counting whether the model has seen one or two "[" is likely done with a different neuron.

Borrowed from Andrej Karpathy

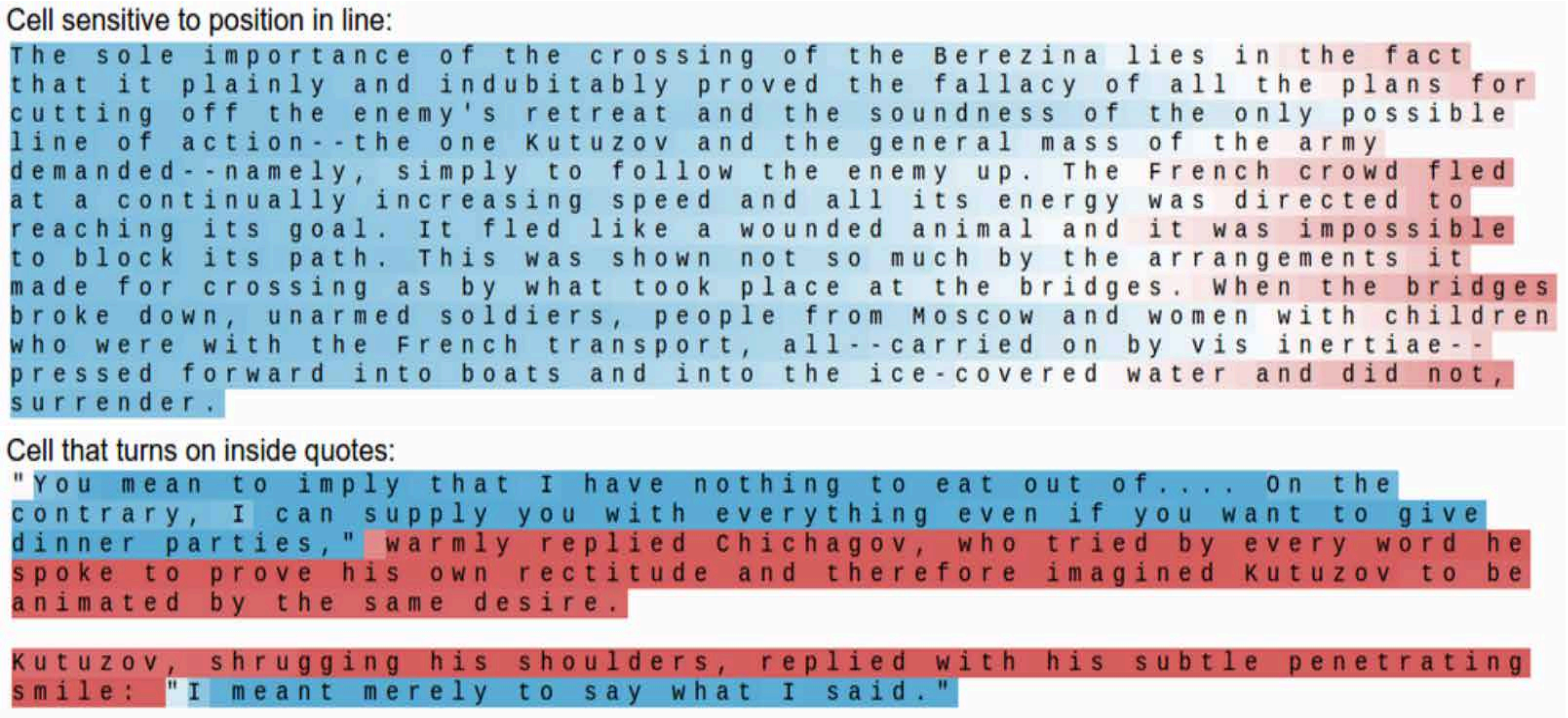
Visualizing predictions and “neuron” firings in RNN



Here we see a neuron that varies seemingly linearly across the [[ ]] environment. In other words its activation is giving the RNN a time-aligned coordinate system across the [[ ]] scope. The RNN can use this information to make different characters more or less likely depending on how early/late it is in the [[ ]] scope (perhaps?).

Borrowed from Andrej Karpathy

Visualizing predictions and "neuron" firings in RNN



Borrowed from Andrej Karpathy

Visualizing predictions and “neuron” firings in RNN

Cell that robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
    siginfo_t *info)
{
 int sig = next_signal(pending, mask);
 if (sig) {
  if (current->notifier) {
   if (sigismember(current->notifier_mask, sig)) {
    if (!(current->notifier)(current->notifier_data)) {
     clear_thread_flag(TIF_SIGPENDING);
     return 0;
    }
   }
  }
  collect_signal(sig, pending, info);
 }
 return sig;
}
```

Borrowed from Andrej Karpathy

Visualizing predictions and “neuron” firings in RNN

Cell that turns on inside comments and quotes:

```
/* Duplicate LSM field information.  The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
        struct audit_field *sf)
{
 int ret = 0;
 char *lsm_str;
 /* our own copy of lsm_str */
 lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
 if (unlikely(!lsm_str))
  return -ENOMEM;
 df->lsm_str = lsm_str;
 /* our own (refreshed) copy of lsm_rule */
 ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
           (void **)&df->lsm_rule);
 /* Keep currently invalid fields around in case they
  * become valid after a policy reload. */
 if (ret == -EINVAL) {
  pr_warn("audit rule for LSM \'%s\' is invalid\n",
   df->lsm_str);
  ret = 0;
 }
 return ret;
}
```

Borrowed from Andrej Karpathy

Visualizing predictions and “neuron” firings in RNN

Cell that is sensitive to the depth of an expression:

```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
  int i;
  if (classes[class]) {
    for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
      if (mask[i] & classes[class][i])
        return 0;
  }
  return 1;
}
```

Borrowed from Andrej Karpathy

Visualizing predictions and “neuron” firings in RNN

Cell that might be helpful in predicting a new line. Note that it only turns on for some “)”:

```
char *audit_unpack_string(void **bufp, size_t *remain, si
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* Of the currently implemented string fields, PATH_MAX
   * defines the longest valid length.
   */
  if (len > PATH_MAX)
    return ERR_PTR(-ENAMETOOLONG);
  str = kmalloc(len + 1, GFP_KERNEL);
  if (unlikely(!str))
    return ERR_PTR(-ENOMEM);
  memcpy(str, *bufp, len);
  str[len] = 0;
  *bufp += len;
  *remain -= len;
  return str;
}
```

Borrowed from Andrej Karpathy

How to generate character strings from the model

- Start the model with its default hidden state.
- Give it a “burn-in” sequence of characters and let it update its hidden state after each character.
- Then look at the probability distribution it predicts for the next character.
- Pick a character randomly from that distribution and tell the net that this was the character that actually occurred.
  - i.e. tell it that its guess was correct, whatever it guessed.
- Continue to let it pick characters until bored.
- Look at the character strings it produces to see what it “knows”.

He was elected President during the Revolutionary War and forgave Opus Paul at Rome. The regime of his crew of England, is now Arab women's icons in and the demons that use something between the characters‘ sisters in lower coil trains were always operated on the line of the ephemerable street, respectively, the graphic or other facility for deformation of a given proportion of large segments at RTUS). The B every chord was a "strongly cold internal palette pour even the white blade.”

Some completions produced by the model

- Sheila thrunges (most frequent)
- People thrunge (most frequent next character is space)
- Shiela, Thrungelini del Rey (first try)
- The meaning of life is literary recognition. (6^{th} try)

- The meaning of life is the tradition of the ancient human reproduction: it is less favorable to the good boy for when to remove her bigger. (one of the first 10 tries for a model trained for longer).

RNN Bible @RNN_Bible · 18 Jun 2016

24:3 And there went up the captivity of Jacob his son, and slew the trumpet of the God of Israel.

1

RNN Bible @RNN_Bible · 18 Jun 2016

107:33 Therefore they that were slain by the sword, and did eat and drink; and they stood before them.

RNN Bible @RNN_Bible · 18 Jun 2016

23:1 And Joash said unto them, Thus shall it be down in the land of Egypt, and they shall come to the most High;

RNN Bible @RNN_Bible · 18 Jun 2016

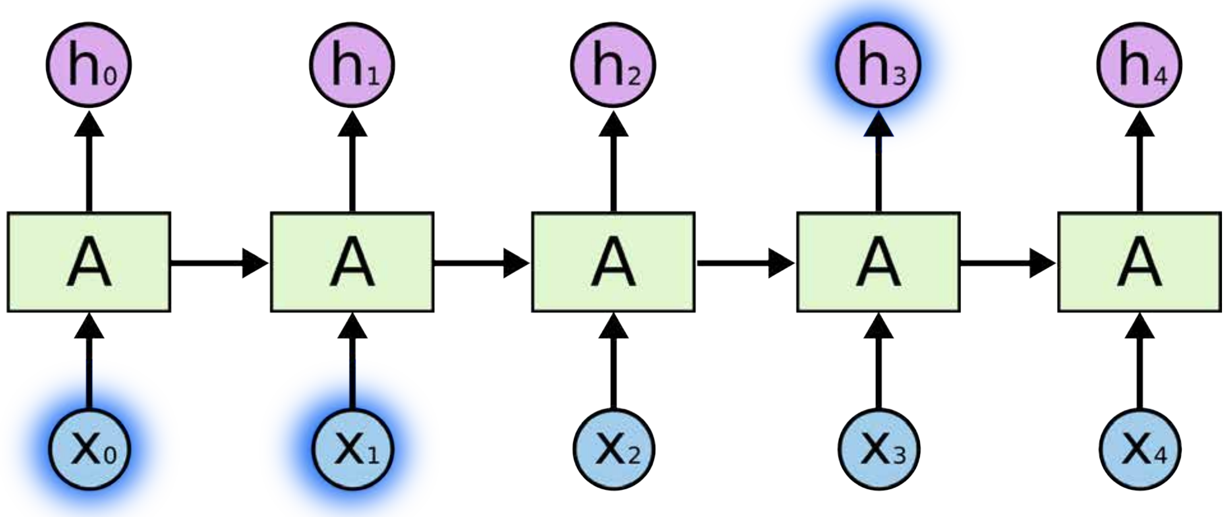
30:22 For I will break them away captive to the mountains, and there shall be no more a prince among the nations.

RNN Bible @RNN_Bible · 18 Jun 2016

2:3 And it shall come to pass, that when thou goest to possess it, that they may not eat thereof.

The Problem of Long-Term Dependencies

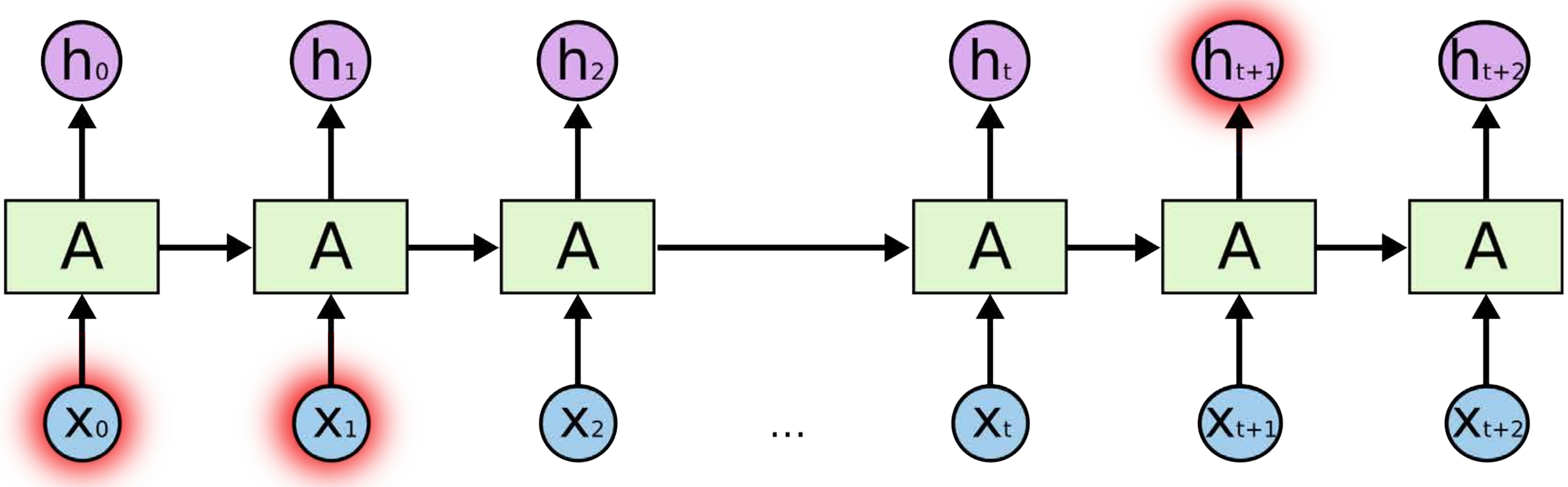
- Sometimes, we only need to look at recent information to perform the present task.
- Predict last word in the sentence:
  - “the clouds are in the *sky*”   (relevant information is very close)



Borrowed from Christopher Olah

The Problem of Long-Term Dependencies

- There are also cases where we need more context.
- Predict last word in the sentence:
  - “I grew up in France… I speak fluent *French*.” (relevant information is far away)



Borrowed from Christopher Olah

Long Short-Term Memory Networks (LSTM)

Long Short-Term Memory, 1997

Sepp Hochreiter, J. Schmidhuber

http://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735

Learning to Forget: Continual Prediction with LSTM, 2000

*F. A. Gers, **J. Schmidhuber**, F. Cummins*

http://www.mitpressjournals.org/doi/abs/10.1162/089976600300015015

LSTM recurrent networks learn simple context-free and context-sensitive languages, 2001

*F. A. Gers, **J. Schmidhuber***

http://ieeexplore.ieee.org/abstract/document/963769