Lecture 11:
Vision + Language
RNNs, LSTMs
Administrative

- Project proposal grades should be out by tonight
  ○ Meet your assigned TA in their office hours for further guidance on the projects
  ○ If concerned, come to my office hours.
Administrative

- A2 is due Friday April 28th, 11:59pm
Administrative: Midterm

Covers material through (including) Lecture 10 (Tue May 2nd everything that we get through).

- It is worth **15%** of your grade.
- available in class. You have 1 hour 20 mins.
- **1 page HANDWRITTEN cheat sheet (double sided) allowed**
- Only make private posts during the next 72 hours after the midterm
Administrative

**Midterm review session:** Fri April 28th discussion section

Sample midterm has been released on EdStem.

**Accommodations:** If you have not received an email from us, please reach out to us over a private EdStem post.
Last Time: Self-supervised learning

**Step 1:** Pretrain a network on a pretext task that doesn’t require supervision

- Input Image: $x$
- Features: $\phi(x)$
- Prediction: $\hat{y}$
- Encoder: $\phi$
- Decoder: $\psi$
- Loss: $L(\hat{y}, y)$

**Step 2:** Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning

- Input Image: $x$
- Features: $\phi(x)$
- Downstream tasks: Image classification, object detection, semantic segmentation
Last Time: Contrastive learning with SimCLR, MoCo, MoCov2, DINO
Today: Vision + language
“Vanilla” Neural Network

Vanilla Neural Networks
Recurrent Neural Networks: Process Sequences

e.g. Image Captioning
image -> sequence of words
Recurrent Neural Networks: Process Sequences

e.g. action prediction, sentiment classification
sequence of video frames -> action class
Recurrent Neural Networks: Process Sequences

- One to one
- One to many
- Many to one
- Many to many

E.g. Video Captioning
Sequence of video frames -> caption
Recurrent Neural Networks: Process Sequences

- **one to one**
- **one to many**
- **many to one**
- **many to many**

**e.g.** Video classification on frame level
Sequential Processing of Non-Sequence Data

Classify images by taking a series of “glimpses”

Figure copyright Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra, 2015. Reproduced with permission.
Sequential Processing of Non-Sequence Data
Generate images one piece at a time!

Gregor et al, "DRAW: A Recurrent Neural Network for Image Generation", ICML 2015
Figure copyright Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra, 2015. Reproduced with permission.
Why existing neural networks are insufficient?

Variable sequence length inputs and outputs!

**Example task:** video captioning

**Input** video can have variable number of frames

**Output** captions can be variable length.

Krishna, et al. Dense captioning Events in Videos. ICCV 2019
Figure copyright Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles, 2019. Reproduced with permission.
Let's start with a task that takes a variable input and produces an output at every step.
Recurrent Neural Network
Recurrent Neural Network

Key idea: RNNs have an “internal state” that is updated as a sequence is processed.
Unrolled RNN

\[ x_1 \xrightarrow{RNN} y_1 \]
\[ x_2 \xrightarrow{RNN} y_2 \]
\[ x_3 \xrightarrow{RNN} y_3 \]
\[ \vdots \]
\[ x_t \xrightarrow{RNN} y_t \]
We can process a sequence of vectors $x$ by applying a **recurrence formula** at every time step:

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

- $h_t$: new state
- $h_{t-1}$: old state
- $x_t$: input vector at some time step

**Function** $f_W$ is some function with parameters $W$. 

**Diagram:**
- Input $x$ into the RNN, output $y$.
RNN output generation

We can process a sequence of vectors \( \mathbf{x} \) by applying a **recurrence formula** at every time step:

\[
\mathbf{y}_t = f_{W_{hy}}(h_t)
\]

output  
new state  
another function with parameters \( W_o \)
Recurrent Neural Network
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.
(Simple) Recurrent Neural Network

The state consists of a single “hidden” vector $h_t$:

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

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

$$y_t = W_{hy}h_t$$

Sometimes called a “Vanilla RNN” or an “Elman RNN” after Prof. Jeffrey Elman
RNN: Computational Graph

![Diagram of RNN computational graph with inputs and outputs labeled as $h_0$, $f_W$, $h_1$, and $x_1$.]
RNN: Computational Graph

\[ h_0 \xrightarrow{f_W} h_1 \xrightarrow{f_W} h_2 \]

\[ x_1 \xrightarrow{} h_1 \]

\[ x_2 \xrightarrow{} h_2 \]
RNN: Computational Graph

\[ h_0 \xrightarrow{f_W} h_1 \xrightarrow{f_W} h_2 \xrightarrow{f_W} h_3 \rightarrow \ldots \rightarrow h_T \]
RNN: Computational Graph

Re-use the same weight matrix at every time-step
RNN: Computational Graph: Many to Many

\[
\begin{align*}
    h_0 & \xrightarrow{f_W} h_1 \\
    x_1 & \xleftarrow{W} h_0 \\
    f_W & \xrightarrow{f_W} h_1 \\
    h_1 & \xrightarrow{f_W} h_2 \\
    x_2 & \xleftarrow{W} h_1 \\
    f_W & \xrightarrow{f_W} h_2 \\
    h_2 & \xrightarrow{f_W} h_3 \\
    x_3 & \xleftarrow{W} h_2 \\
    f_W & \xrightarrow{f_W} h_3 \\
    \vdots & \xrightarrow{f_W} \ldots \\
    h_T & \xrightarrow{f_W} h_T \\
    y_1 & \xleftarrow{f_W} h_1 \\
    & \text{\ldots} \\
    y_T & \xleftarrow{f_W} h_T
\end{align*}
\]
RNN: Computational Graph: Many to Many

\[ h_0 \xrightarrow{f_W} h_1 \xrightarrow{f_W} h_2 \xrightarrow{f_W} \ldots \xrightarrow{f_W} h_T \]

\[ x_1 \xrightarrow{W} h_0 \]

\[ y_1 \xrightarrow{L_1} y_2 \xrightarrow{L_2} y_3 \xrightarrow{L_3} y_T \]
RNN: Computational Graph: Many to Many
RNN: Computational Graph: Many to One

\[
\begin{align*}
  h_0 &\rightarrow f_W & h_1 &\rightarrow f_W & h_2 &\rightarrow f_W & h_3 &\rightarrow \ldots & h_T \\
  x_1 &\rightarrow & x_2 &\rightarrow & x_3 &\rightarrow & \ldots & y \\
  W &\rightarrow & & & & & & 
\end{align*}
\]
RNN: Computational Graph: Many to One
RNN: Computational Graph: One to Many

\[ h_0 \xrightarrow{f_W} h_1 \xrightarrow{f_W} h_2 \xrightarrow{f_W} h_3 \rightarrow \ldots \rightarrow h_T \]

\[ y_1 \xrightarrow{W} x \rightarrow y_2 \rightarrow y_3 \rightarrow y_T \]
RNN: Computational Graph: One to Many

\[ h_0 \xrightarrow{f_W} h_1 \xrightarrow{f_W} h_2 \xrightarrow{f_W} h_3 \xrightarrow{\ldots} h_T \]

\[ \begin{array}{cccccccccc}
W & x & ? & ? & ? & ? & \ldots & \end{array} \]
RNN: Computational Graph: One to Many

\[ h_0 \xrightarrow{f_W} h_1 \xrightarrow{f_W} h_2 \xrightarrow{f_W} h_3 \xrightarrow{f_W} \ldots \xrightarrow{f_W} h_T \]

\[ x \xrightarrow{W} \]

\[ y_1 \xrightarrow{f_W} y_2 \xrightarrow{f_W} y_3 \xrightarrow{f_W} \ldots \xrightarrow{f_W} y_T \]

\[ 0 \xrightarrow{f_W} 0 \xrightarrow{f_W} 0 \xrightarrow{f_W} \ldots \xrightarrow{f_W} 0 \]
RNN: Computational Graph: One to Many

\[ h_0 \rightarrow f_W \rightarrow h_1 \rightarrow f_W \rightarrow h_2 \rightarrow f_W \rightarrow h_3 \rightarrow \ldots \rightarrow h_T \]

\[ W \rightarrow x \rightarrow y_1 \rightarrow y_2 \rightarrow y_3 \rightarrow \ldots \rightarrow y_{T-1} \]
Sequence to Sequence: Many-to-one + one-to-many

**Many to one:** Encode input sequence in a single vector

Sutskever et al, “Sequence to Sequence Learning with Neural Networks”, NIPS 2014
Sequence to Sequence: Many-to-one + one-to-many

**Many to one**: Encode input sequence in a single vector

**One to many**: Produce output sequence from single input vector

---

Sutskever et al, “Sequence to Sequence Learning with Neural Networks”, NIPS 2014
Example:
Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”
Example:
Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: “hello”
Example:
Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: “hello”
Example: Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: “hello”
Example: Character-level Language Model

So far: encode inputs as one-hot-vector

\[
\begin{bmatrix}
    w_{11} & w_{12} & w_{13} & w_{14} \\
    w_{21} & w_{22} & w_{23} & w_{14} \\
    w_{31} & w_{32} & w_{33} & w_{14} \\
    0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
    1 \\
    0 \\
    0 \\
    0
\end{bmatrix}
= 
\begin{bmatrix}
    w_{11} \\
    w_{21} \\
    w_{31} \\
    0
\end{bmatrix}
\begin{bmatrix}
    1 \\
    0 \\
    0 \\
    0
\end{bmatrix}
\]

Matrix multiply with a one-hot vector just extracts a column from the weight matrix. Often extract this into a separate embedding layer.
Example:
Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model
Example:
Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model
Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model
Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

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

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
Truncated Backpropagation through time

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

Ranjay Krishna, Aditya Kusupati
```python
# min-char-rnn.py gist: 112 lines of Python

```
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 art now the world's fresh ornament,
And only herald to the gaudy spring,
Within thine own bud buried thy content,
And tender churl mak'st waste in niggardling:
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 desirest thou this 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:

"Tmont thithey" fomesserliund
Keushey. Thom here
sheulke, amerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennnc Phe lism thond hon at. MeiDimoration in ther thize."

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.

"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftended 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 strain 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 VINCENZIO:
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: open source algebraic geometry textbook

The Stacks Project

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
http://stacks.math.columbia.edu/
The stacks project is licensed under the GNU Free Documentation License
For $\mathcal{L}_{m*} = 0$, 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

$$ S = \text{Spec}(R) = U \times X U \times X U $$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\prod I_a \times X U \to V$. Consider the maps $M$ along the set of points $\text{Sch}_{pp} x U \to U$ 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 $\text{Sh}(G)$ such that $\text{Spec}(R) \to S$ is smooth or an

$$ U = \bigcup U_i \times S_i $$

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

To prove study we see that $\mathcal{F}_i$ is a covering of $X'$, and $\mathcal{F}_i$ is an object of $\mathcal{F}_X/S$ for $i > 0$ and $\mathcal{F}_0$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $C$ as a $C$-module.

In particular $\mathcal{F} = U \mathcal{F}$ we have to show that

$$ \overline{M}^* = T^* \otimes_{\text{Spec}(k)} \mathcal{O}_{S, a} \otimes \mathcal{O}(F) $$

is a unique morphism of algebraic stacks. Note that

$$ \text{Arrows} = (\text{Sch}/S)_{pp}^{opp}, (\text{Sch}/S)_{ppj} $$

and

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

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 $X_{\text{space}, \text{etale}}$ which gives an open subspace of $X$ and $T$ equal to $S_{\text{etale}}$, 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 $\text{Proj}_X(A) = \text{Spec}(B)$ over $U$ compatible with the complex

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

When in this case of to show that $Q \to C_{1/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definiion ?? (without element is when the closed subschemes are etale).

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 $S = \text{Spec}(R)$ and $Y = \text{Spec}(R)$.

Proof. This is form all sheaves of sheaves on $X$. But given a scheme $S$ and a surjective etale morphism $U \to X$. Let $U \cap U = \lim U_i$ be the scheme $X$ over $S$ at the schemes $X_i \to X$ and $U = \lim X_i$.

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

**Lemma 0.2.** Let $X$ be a locally Noetherian scheme over $S$, $E = \mathcal{F}_{X/S}$. Set $T = J_1 \subset T'$, Since $T' \subset T^n$ are nonzero over $i_0 \leq p$ is a subset of $\mathcal{J}_{a, 0} \otimes \mathcal{J}_{a, 2}$ works.

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

Proof. We will use the property we see that $p$ is the next functor (??). On the other hand, by Lemma ?? we see that

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

where $K$ is an $F$-algebra where $\delta_{a+1}$ is a scheme over $S$. 

---

Ranjay Krishna, Aditya Kusupati  
Lecture 11 - 59  
May 02, 2023
Lemma 0.1. Let \( C \) be a set of the construction.
Let \( C \) be a gerber covering. Let \( F \) 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 \( F \) on \( X_{\text{etale}} \) we have
\[
\mathcal{O}_X(F) = \{\text{morph}_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}
\]
where \( \mathcal{G} \) defines an isomorphism \( F \to F \) of \( \mathcal{O} \)-modules.

Lemma 0.2. This is an integer \( Z \) 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 \( U \subset X \) 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 \( F \) be a quasi-coherent sheaf of \( \mathcal{O}_X \)-modules. The following are equivalent
1. \( F \) 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.
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 & (i << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
      ((count & 0x0000000000000000) & 0x0000000f) << 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, &offset);
  /* 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 ");
}
/*
 * Copyright (c) 2006-2010, Intel Mobile Communications. All rights reserved.
 *
 * This program is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License version 2 as published by
 * the Free Software Foundation.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 *
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software Foundation,
 * Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
 */

#include <linux/kexec.h>
#include <linux/errno.h>
#include <linux/io.h>
#include <linux/platform_device.h>
#include <linux/multi.h>
#include <linux/ckevent.h>

#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/setev.h>
#include <asm/pgproto.h>
```c
#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/setuw.h>
#include <asm/pgproto.h>

#define REG_PG   vesa_slot_addr_pack
#define PFM_NOCOMP AFSR(0, load)
#define STACK_DDR(type) (func)

#define SWAP_ALLOCATE(nr) (e)
#define emulate_sigs() arch_get_unaligned_child()
#define access_rw(TST) asm volatile("movd %esp, %0, %3 : : "r" (0)); \n   if (__type & DO_READ)

static void stat_PC_SEC __read_mostly offsetof(struct seq_argsqueue, \n   pC>[1]);

static void os_prefix(unsigned long sys)
{
    #ifdef CONFIG_PREEMPT
        PUT_PARAM_RAID(2, sel) = get_state_state();
        set_pid_sum((unsigned long)state, current_state_str(),
                (unsigned long)-1->lr_full; low;
    ```
Input: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Output: The scientist named the population, after their distinctive horn, Ovid’s Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.
**Prompt:** Can you generate a pyplot for the following data: $x = [1, 3, 5, 6, 8], y_1 = [2, 3, 5, 18, 1], y_2 = [3, 5, 6, 8, 1], y_3 = [5, 1, 2, 3, 4], y_4 = [9, 7, 2, 3, 1]$. I want $y_1, y_2$ to be in the same plot, but $y_3$ is in another plot next to that plot, $y_4$ is in below. I want the legend of $y_1$ to be "bob", $y_2$ to be "alice", $y_3$ to be "bilbo", $y_4$ to be "allie". I want the x-axis to be labeled with "time" and y axis to be labeled with "money". I want to add a 10%-40% random error bar to each curve, through all times (including non-integers). I want smoothed curves to show the plot, and smoothed error bar. Do not use linear interpolation, use smooth interpolation! I want to also add some small zig-zag to the smoothed curve to make it look more real. I want to put a baseline as the mean of each line. I want to put a pie chart below indicating the fraction of the four people in each time step. I also want an animation to show how the fractions are changing in the pie chart in **continuous time**. Interpolate the missing fractions! I also want vertical line animation in other three plots to match the pie chart. I want the fanciest plot. Please add as many fancy things as possible.

Here is a possible code to generate a pyplot for the following data, with some comments to explain the steps:

**GPT-4:** [We give two snapshots from the animation resulting from the produced code]

![Figure 1](image1.png)  ![Figure 2](image2.png)  ![Figure 3](image3.png)  ![Figure 4](image4.png)

Bubek et al. Sparks of AGI. ArXiv 2023
Language modeling leads to reasoning capabilities for GPT-4

Prompt: Here we have a book, 9 eggs, a laptop, a bottle and a nail. Please tell me how to stack them onto each other in a stable manner.

GPT-4:

One possible way to stack the objects onto each other in a stable manner is:

- Place the book flat on a level surface, such as a table or a floor. The book will serve as the base of the stack and provide a large and sturdy support.
- Arrange the 9 eggs in a 3 by 3 square on top of the book, leaving some space between them. The eggs will form a second layer and distribute the weight evenly. Make sure the eggs are not cracked or broken, and handle them gently to avoid dropping them.
- Place the laptop on top of the eggs, with the screen facing down and the keyboard facing up. The laptop will fit snugly within the boundaries of the book and the eggs, and its flat and rigid surface will provide a stable platform for the next layer. Make sure the laptop is closed and turned off, and do not press any keys or buttons.
- Place the bottle on top of the laptop, with the cap facing up and the bottom facing down. The bottle will add some height and balance to the stack, and its cylindrical shape will reduce the contact area with the laptop. Make sure the bottle is empty and closed, and do not shake or tilt it.
- Place the nail on top of the bottle cap, with the pointy end facing up and the flat end facing down. The nail will be the final and smallest object in the stack, and its sharp tip will prevent it from rolling or sliding off the bottle. Make sure the nail is clean and straight, and do not touch or move it.

ChatGPT:

It would be difficult to stack all of these objects on top of each other in a stable manner without using any extra objects. The book and laptop are likely too large and heavy to stack on top of the eggs without damaging them. The bottle and nail may be too small to provide a stable base for the other objects.

Here is one possible way to stack the objects:

- Place the bottle on a flat surface.
- Carefully balance the nail on top of the bottle.
- Place the eggs on top of the nail, making sure they are balanced and not tilting to one side.
- Place the laptop on top of the eggs, again making sure it is balanced and not tilting to one side.
- Finally, place the book on top of the laptop.

This stack may not be very stable, so it is important to be careful when handling it. Alternatively, you could try stacking the objects in a different order or using some of the objects as supports for others to create a more stable stack.
RNN tradeoffs

RNN Advantages:
- Can process any length input
- Computation for step $t$ can (in theory) use information from many steps back
- Model size doesn’t increase for longer input
- Same weights applied on every timestep, so there is symmetry in how inputs are processed.

RNN Disadvantages:
- Recurrent computation is slow
- In practice, difficult to access information from many steps back
Image Captioning

Explain Images with Multimodal Recurrent Neural Networks, Mao et al.
Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy and Fei-Fei
Show and Tell: A Neural Image Caption Generator, Vinyals et al.
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
Recurrent Neural Network

Convolutional Neural Network
test image
before:
\[ h = \tanh(W_{\text{xh}} \cdot x + W_{\text{hh}} \cdot h) \]

now:
\[ h = \tanh(W_{\text{xh}} \cdot x + W_{\text{hh}} \cdot h + W_{\text{ih}} \cdot v) \]
test image

sample!

Ranjay Krishna, Aditya Kusupati
Lecture 11 - May 02, 2023
test image
test image

sample

<END> token => finish.
Image Captioning: Example Results

A cat sitting on a suitcase on the floor
A cat is sitting on a tree branch
A dog is running in the grass with a frisbee
A white teddy bear sitting in the grass

Two people walking on the beach with surfboards
A tennis player in action on the court
Two giraffes standing in a grassy field
A man riding a dirt bike on a dirt track
Image Captioning: Failure Cases

A woman is holding a cat in her hand

A woman standing on a beach holding a surfboard

A bird is perched on a tree branch

A man in a baseball uniform throwing a ball

A person holding a computer mouse on a desk

Captions generated using neuraltalk2
All images are CC0 Public domain: fur coat, handstand, spider web, baseball
Visual Question Answering (VQA)

Q: What endangered animal is featured on the truck?
A: A bald eagle.
A: A sparrow.
A: A humming bird.
A: A raven.

Q: Where will the driver go if turning right?
A: Onto 24 ¾ Rd.
A: Onto 25 ¾ Rd.
A: Onto 23 ¾ Rd.
A: Onto Main Street.

Q: When was the picture taken?
A: During a wedding.
A: During a bar mitzvah.
A: During a funeral.
A: During a Sunday church service.

Q: Who is under the umbrella?
A: Two women.
A: A child.
A: An old man.
A: A husband and a wife.

Figure from Zhu et al, copyright IEEE 2016. Reproduced for educational purposes.
Visual Question Answering: RNNs with Attention

“How many horses are in this image?”

Visual Dialog: Conversations about images

Das et al., "Visual Dialog", CVPR 2017
Figures from Das et al, copyright IEEE 2017. Reproduced with permission.
Agent encodes instructions in language and uses an RNN to generate a series of movements as the visual input changes after each move.

**Instruction**

Turn right and head towards the *kitchen*. Then turn left, pass a *table* and enter the *hallway*. Walk down the hallway and turn into the *entry way* to your right without doors. Stop in front of the *toilet*.

---


Figures from Wang et al., copyright IEEE 2017. Reproduced with permission.
Visual Question Answering: Dataset Bias

What is the dog playing with?

Frisbee

Image

Question

Answer

Model

Yes or No

Multilayer RNNs

Each layer has a different set of weights

Outputs from one layer become inputs to the layer above.
Vanilla RNN Gradient Flow

\[ h_t = \tanh(W_{hh}h_{t-1} + W_{hx}x_t) \]
\[ = \tanh \begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \]
\[ = \tanh \begin{pmatrix} W \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \]
Vanilla RNN Gradient Flow

Backpropagation from $h_t$ to $h_{t-1}$ multiplies by $W$ (actually $W_{hh}^T$)

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh \left( (W_{hh} \quad W_{hx}) \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)$$
$$= \tanh \left( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)$$

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Vanilla RNN Gradient Flow

Backpropagation from $h_t$ to $h_{t-1}$ multiplies by $W$ (actually $W_{hh}^T$)

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh \left( \begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)$$
$$= \tanh \left( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)$$

$$\frac{\partial h_t}{\partial h_{t-1}} = \tanh'(W_{hh}h_{t-1} + W_{xh}x_t)W_{hh}$$
Vanilla RNN Gradient Flow

Backpropagation from $h_t$ to $h_{t-1}$ multiplies by $W$ (actually $W_{hh}^T$)

$$ \frac{\partial h_t}{\partial h_{t-1}} = \tanh'(W_{hh} h_{t-1} + W_{xh} x_t) W_{hh} $$

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Vanilla RNN Gradient Flow

\[
\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}
\]

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Vanilla RNN Gradient Flow

Gradients over multiple time steps:

\[
\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}
\]

\[
\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \frac{\partial h_T}{\partial h_{T-1}} \cdots \frac{\partial h_1}{\partial W}
\]

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994

Vanilla RNN Gradient Flow

Gradients over multiple time steps:

\[
\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}
\]

\[
\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \frac{\partial h_T}{\partial h_{t-1}} \cdots \frac{\partial h_1}{\partial W} = \frac{\partial L_T}{\partial h_T} \left( \prod_{t=2}^{T} \frac{\partial h_t}{\partial h_{t-1}} \right) \frac{\partial h_1}{\partial W}
\]

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Vanilla RNN Gradient Flow
Gradients over multiple time steps:

\[
\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}
\]

\[
\frac{\partial h_t}{\partial h_{t-1}} = \tanh'(W_{hh} h_{t-1} + W_{xh} x_t) W_{hh}
\]

\[
\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \frac{\partial h_T}{\partial h_{t-1}} \cdots \frac{\partial h_1}{\partial W} = \frac{\partial L_T}{\partial h_T} \left( \prod_{t=2}^{T} \frac{\partial h_t}{\partial h_{t-1}} \right) \frac{\partial h_1}{\partial W}
\]

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Vanilla RNN Gradient Flow

Gradients over multiple time steps:

\[ \frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W} \]

Almost always < 1

Vanishing gradients

\[ \frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} (\prod_{t=2}^{T} tanh'(W_{hh} h_{t-1} + W_{xh} x_t))W_{hh}^{T-1} \frac{\partial h_1}{\partial W} \]
Vanilla RNN Gradient Flow

Gradients over multiple time steps:

\[
\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}
\]

What if we assumed no non-linearity?
Vanilla RNN Gradient Flow

Gradients over multiple time steps:

\[ \frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W} \]

\[ \frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} W^{T-1} \frac{\partial h_1}{\partial W} \]

What if we assumed no non-linearity?

Largest singular value > 1: Exploding gradients

Largest singular value < 1: Vanishing gradients

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Vanilla RNN Gradient Flow

Gradients over multiple time steps:

\[
\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}
\]

What if we assumed no non-linearity?

Largest singular value > 1: Exploding gradients

Largest singular value < 1: Vanishing gradients

What if we assumed no non-linearity?

Gradient clipping:
Scale gradient if its norm is too big

\[
\text{grad\_norm} = \text{np.sum(grad * grad)}
\]
\[
\text{if grad\_norm > threshold:}
\]
\[
\text{grad *= (threshold / grad\_norm)}
\]
Vanilla RNN Gradient Flow

Gradients over multiple time steps:

What if we assumed no non-linearity?

\[
\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}
\]

Largest singular value > 1: Exploding gradients

\[
\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} W^{T-1} \frac{\partial h_1}{\partial W}
\]

Largest singular value < 1: Vanishing gradients

Change RNN architecture

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Long Short Term Memory (LSTM)

Vanilla RNN

\[ h_t = \tanh \left( W \left( h_{t-1}, x_t \right) \right) \]

LSTM

\[
\begin{pmatrix}
  i \\
  f \\
  o \\
  g
\end{pmatrix} = \begin{pmatrix}
  \sigma \\
  \sigma \\
  \sigma \\
  \tanh
\end{pmatrix} W \begin{pmatrix}
  h_{t-1} \\
  x_t
\end{pmatrix}
\]

\[
c_t = f \odot c_{t-1} + i \odot g
\]

\[
h_t = o \odot \tanh(c_t)
\]

Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation 1997
Long Short Term Memory (LSTM)
[Hochreiter et al., 1997]
Long Short Term Memory (LSTM)  
[Hochreiter et al., 1997]

\[
\begin{align*}
&\text{Input gate, whether to write to cell} \\
&\text{Forget gate, Whether to erase cell} \\
&\text{Output gate, How much to reveal cell} \\
&\text{Info gate, How much to write to cell}
\end{align*}
\]

\[
\begin{align*}
\text{vector from below (x)} & \quad 4h \times 2h \\
\text{vector from before (h)} & \quad 4h \\
\text{g: Info gate, How much to write to cell} & \quad 4 \times h
\end{align*}
\]

\[
\begin{align*}
(i) & = \begin{pmatrix}
sigmoid \\
sigmoid \\
sigmoid \\
tanh
\end{pmatrix} \\
(f) & = \begin{pmatrix}
sigmoid \\
sigmoid \\
tanh
\end{pmatrix} \\
o & = o \\
g & = g
\end{align*}
\]

\[
\begin{align*}
&c_t = f \odot c_{t-1} + i \odot g \\
h_t = o \odot \tanh(c_t)
\end{align*}
\]
Long Short Term Memory (LSTM)  
[Hochreiter et al., 1997]

- **i**: Input gate, whether to write to cell
- **g**: Info gate, How much to write to cell
- **f**: Forget gate, Whether to erase cell
- **o**: Output gate, How much to reveal cell
- **g**: Info gate, How much to write to cell

\[
\begin{pmatrix}
i \\ f \\ o \\ g \\
\end{pmatrix} =
\begin{pmatrix}
\sigma \\ \sigma \\ \sigma \\ \tanh \\
\end{pmatrix} W \begin{pmatrix}
h_{t-1} \\ x_t \\
\end{pmatrix}
\]

\[
c_t = f \odot c_{t-1} + i \odot g \\
h_t = o \odot \tanh(c_t)
\]
Long Short Term Memory (LSTM)  
[Hochreiter et al., 1997]

- **i**: Input gate, whether to write to cell  
- **f**: Forget gate, Whether to erase cell  
- **g**: Info gate, How much to write to cell

\[
\begin{pmatrix}
i \\ f \\ o \\ g
\end{pmatrix}
=\begin{pmatrix}
sigmoid \\ sigmoid \\ sigmoid \\ \text{tanh}
\end{pmatrix}
\begin{pmatrix}
sigmoid \\ \text{tanh}
\end{pmatrix}
\begin{pmatrix}
h_{t-1} \\ x_t
\end{pmatrix}
\]

\[
c_t = f \odot c_{t-1} + i \odot g
\]

\[
h_t = o \odot \text{tanh}(c_t)
\]
Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]

\[ (i) = \begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \]

\[ c_t = f \odot c_{t-1} + i \odot g \]

\[ h_t = o \odot \tanh(c_t) \]
Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]

\[
\begin{align*}
    c_t &= f \odot c_{t-1} + i \odot g \\
    h_t &= o \odot \tanh(c_t)
\end{align*}
\]

\[
\begin{pmatrix}
    i \\
    f \\
    o \\
    g
\end{pmatrix}
= \begin{pmatrix}
    \sigma \\
    \sigma \\
    \sigma \\
    \tanh
\end{pmatrix}
W \begin{pmatrix}
    h_{t-1} \\
    x_t
\end{pmatrix}
\]
Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]

Backpropagation from \( c_t \) to \( c_{t-1} \) only elementwise multiplication by \( f \), no matrix multiply by \( W \)

\[
\begin{pmatrix}
i \\
f \\
o \\
g
\end{pmatrix} =
\begin{pmatrix}
\sigma \\
\sigma \\
\sigma \\
tanh
\end{pmatrix}
W
\begin{pmatrix}
h_{t-1} \\
x_t
\end{pmatrix}
\]

\[
c_t = f \odot c_{t-1} + i \odot g
\]

\[
h_t = o \odot \tanh(c_t)
\]
Long Short Term Memory (LSTM): Gradient Flow
[Hochreiter et al., 1997]

Uninterrupted gradient flow!

Notice that the gradient contains the $f$ gate’s vector of activations
- allows better control of gradients values, using suitable parameter updates of the forget gate.

Also notice that are added through the $f$, $i$, $g$, and $o$ gates
- better balancing of gradient values
Do LSTMs solve the vanishing gradient problem?

The LSTM architecture makes it easier for the RNN to preserve information over many timesteps

- e.g. if the $f = 1$ and the $i = 0$, then the information of that cell is preserved indefinitely.
- By contrast, it’s harder for vanilla RNN to learn a recurrent weight matrix $Wh$ that preserves info in hidden state

LSTM doesn’t guarantee that there is no vanishing/exploding gradient, but it does provide an easier way for the model to learn long-distance dependencies
Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]

Uninterrupted gradient flow!

Similar to residual connections (e.g. in ResNets and Transformers), which we will learn about soon!
Neural Architecture Search for RNN architectures

Zoph et Le, "Neural Architecture Search with Reinforcement Learning", ICLR 2017
Other RNN Variants

**GRU** [Learning phrase representations using rnn encoder-decoder for statistical machine translation, Cho et al. 2014]

\[
\begin{align*}
    r_t &= \sigma(W_{xr} x_t + W_{hr} h_{t-1} + b_r) \\
    z_t &= \sigma(W_{xz} x_t + W_{hz} h_{t-1} + b_z) \\
    \tilde{h}_t &= \tanh(W_{xh} x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \\
    h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t 
\end{align*}
\]

[An Empirical Exploration of Recurrent Network Architectures, Jozefowicz et al., 2015]

**MUT1:**

\[
\begin{align*}
    z &= \text{sigm}(W_{xz} x_t + b_z) \\
    r &= \text{sigm}(W_{xr} x_t + W_{hr} h_t + b_r) \\
    h_{t+1} &= \tanh(W_{hh}(r \odot h_t) + \tanh(x_t) + b_h) \odot z + h_t \odot (1 - z)
\end{align*}
\]

**MUT2:**

\[
\begin{align*}
    z &= \text{sigm}(W_{xz} x_t + W_{hz} h_t + b_z) \\
    r &= \text{sigm}(x_t + W_{hr} h_t + b_r) \\
    h_{t+1} &= \tanh(W_{hh}(r \odot h_t) + W_{xh} x_t + b_h) \odot z + h_t \odot (1 - z)
\end{align*}
\]

**MUT3:**

\[
\begin{align*}
    z &= \text{sigm}(W_{xz} x_t + W_{hz} \tanh(h_t) + b_z) \\
    r &= \text{sigm}(W_{xr} x_t + W_{hr} h_t + b_r) \\
    h_{t+1} &= \tanh(W_{hh}(r \odot h_t) + W_{xh} x_t + b_h) \odot z + h_t \odot (1 - z)
\end{align*}
\]
Recurrence for Vision

- LSTM was a good default choice until this year
- Use variants like GRU if you want faster compute and less parameters
- Use transformers (next lecture) as they are dominating NLP and also vision models
  - almost everyday there is a new transformer model

Su et al. "VI-bert: Pre-training of generic visual-linguistic representations." ICLR 2020
Li et al. "Visualbert: A simple and performant baseline for vision and language." arXiv 2019
Summary

- RNNs allow a lot of flexibility in architecture design
- Vanilla RNNs are simple but don’t work very well
- Common to use LSTM or GRU: their additive interactions improve gradient flow
- Backward flow of gradients in RNN can explode or vanish. Exploding is controlled with gradient clipping. Vanishing is controlled with additive interactions (LSTM)
- Better/simpler architectures are a hot topic of current research, as well as new paradigms for reasoning over sequences
- Better understanding (both theoretical and empirical) is needed.
Next time: Attention and transformers!
Searching for interpretable cells
Searching for interpretable cells

```
/* unpack a filter field's string representation from user-space */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
    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. */
```
Searching for interpretable cells

"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

quote detection cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016
Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission
Searching for interpretable cells

Cell sensitive to position in line:
The sole importance of the crossing of the Berezina lies in the fact that it plainly and indubitably proved the fallacy of all the plans for cutting off the enemy's retreat and the soundness of the only possible line of action--the one Kutuzov and the general mass of the army demanded--namely, simply to follow the enemy up. The French crowd fled at a continually increasing speed and all its energy was directed to reaching its goal. It fled like a wounded animal and it was impossible to block its path. This was shown not so much by the arrangements it made for crossing as by what took place at the bridges. When the bridges broke down, unarmed soldiers, people from Moscow and women with children who were with the French transport, all--carried on by vis inertiae--pressed forward into boats and into the ice-covered water and did not surrender.

line length tracking cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016
Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission
Searching for interpretable cells

if statement cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016
Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission
Searching for interpretable cells

```
/* Duplicate LSM field information. The lsm_rule is opaque, so */
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;
}
```

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016
Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission
Searching for interpretable cells

```
#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;
}
```

code depth cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016
Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission