High Level Computer Vision

Recurrent Neural Networks
@ June 5, 2019

Bernt Schiele & Mario Fritz

www.mpi-inf.mpg.de/hlcv/

Max Planck Institute for Informatics & Saarland University,
Saarland Informatics Campus Saarbrücken
Overview Today’s Lecture

• Recurrent Neural Networks (RNNs)
  ▸ Motivation & flexibility of RNNs (some recap from last week)
  ▸ Language modeling
    - including “unreasonable effectiveness of RNNs”
  ▸ RNNs for image description / captioning
  ▸ Standard RNN and a particularly successful RNN: Long Short Term Memory (LSTM)
    - including “visualizations of RNN cells”
Recurrent Networks offer a lot of flexibility:
Sequences in Vision

Sequences in the input…
(many-to-one)

Running
Jumping
Dancing
Fighting
Eating
Sequences in Vision

Sequences in the output…
(one-to-many)

A happy brown dog.
Sequences in Vision

Sequences everywhere! (many-to-many)

A dog jumps over a hurdle.
Problem #1

fixed-size, static input
Problem #1

fixed-size, static input

slide credit: Jeff Donahue
Problem #2

Krizhevsky et al., NIPS 2012
Problem #2

output is a single choice from a fixed list of options

- cat
- **dog**
- horse
- fish
- snake
Problem #2

output is a single choice from a fixed list of options

- a happy brown dog
- a big brown dog ✔
- a happy red dog
- a big red dog
- ...

slide credit: Jeff Donahue
Recurrent Networks offer a lot of flexibility:
Recurrent Neural Network (RNN)

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Recurrent Neural Network (RNN)

usually want to predict a vector at some time steps
Recurrent Neural Network (RNN)

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

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

- new state
- old state
- input vector at some time step

some function with parameters $W$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Recurrent Neural Network (RNN)

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.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
(Simple) Recurrent Neural Network

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

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

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

$$y_t = W_{hy} h_t$$

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
RNN: Computational Graph

\[ h_t = f_W(h_{t-1}, x_t) \]
RNN: Computational Graph

\[ h_t = f_W(h_{t-1}, x_t) \]
RNN: Computational Graph

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

Slide credit: Fei-Fei, Justin Johnson, Serena Yeung
RNN: Computational Graph: Many to Many

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
RNN: Computational Graph: Many to Many

Slide credit: Fei-Fei, Justin Johnson, Serena Yeung
RNN: Computational Graph: Many to Many

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

\[ y_1 \rightarrow L_1 \rightarrow y_2 \rightarrow L_2 \rightarrow y_3 \rightarrow L_3 \rightarrow y_T \rightarrow L_T \]

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
RNN: Computational Graph: Many to One
RNN: Computational Graph: One to Many

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Sequence to Sequence

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

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Recurrent Networks offer a lot of flexibility:
Language Models

Recurrent Neural Network Based Language Model
[Tomas Mikolov, 2010]
Suppose we had the training sentence “cat sat on mat”

We want to train a language model:
\[
P(\text{next word} \mid \text{previous words})
\]

i.e. want these to be high:
\[
P(\text{cat} \mid [\langle S\rangle])
P(\text{sat} \mid [\langle S\rangle, \text{cat}])
P(\text{on} \mid [\langle S\rangle, \text{cat}, \text{sat}])
P(\text{mat} \mid [\langle S\rangle, \text{cat}, \text{sat}, \text{on}])
P(\langle E\rangle \mid [\langle S\rangle, \text{cat}, \text{sat}, \text{on}, \text{mat}])
\]
Suppose we had the training sentence “cat sat on mat”

We want to train a **language model**:
\[ P(\text{next word} \mid \text{previous words}) \]

First, suppose we had only a finite, 1-word history:
i.e. want these to be high:
\[ P(\text{cat} \mid <S>) \]
\[ P(\text{sat} \mid \text{cat}) \]
\[ P(\text{on} \mid \text{sat}) \]
\[ P(\text{mat} \mid \text{on}) \]
\[ P(<E> \mid \text{mat}) \]
“cat sat on mat”

300 (learnable) numbers associated with each word in vocabulary
“cat sat on mat”

hidden layer (e.g. 500-D vectors)
$h_4 = \tanh(0, W_{xh} \ast x_4)$

300 (learnable) numbers associated with each word in vocabulary
“cat sat on mat”

10,001-D class scores:
Softmax over 10,000 words and a special <END> token.
\[ y_4 = \text{Why} \times h_4 \]

hidden layer (e.g. 500-D vectors)
\[ h_4 = \tanh(0, W_{xh} \times x_4) \]

300 (learnable) numbers associated with each word in vocabulary
Recurrent Neural Network:

```
<table>
<thead>
<tr>
<th>x0</th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;START&gt;</td>
<td>&quot;cat&quot;</td>
<td>&quot;sat&quot;</td>
<td>&quot;on&quot;</td>
<td>&quot;mat&quot;</td>
</tr>
</tbody>
</table>
```

10,001-D class scores:
Softmax over 10,000 words and a special <END> token.
\[ y4 = \text{Why} \times h4 \]

hidden layer (e.g. 500-D vectors)
\[ h4 = \tanh(0, Wxh \times x4 + Whh \times h3) \]

300 (learnable) numbers associated with each word in vocabulary.

slide credit: Andrej Karpathy
Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

\[ P(\text{next word} \mid \text{previous words}) \]
Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

\[ P(\text{next word} \mid \text{previous words}) \]
Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

\[ P(\text{next word} \mid \text{previous words}) \]
Training this on a lot of sentences would give us a language model. A way to predict

\[ P(\text{next word} \mid \text{previous words}) \]
Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

\[ P(\text{next word} \mid \text{previous words}) \]
Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

\[ P(\text{next word} \mid \text{previous words}) \]
Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

\[ P(\text{next word} \mid \text{previous words}) \]
Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

$$P(\text{next word} \mid \text{previous words})$$
Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

\[ P(\text{next word} \mid \text{previous words}) \]
Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

\[ P(\text{next word} | \text{previous words}) \]
Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

$$P(\text{next word} \mid \text{previous words})$$
Example...

Example:
Character-level
Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”
Example...

Example:
Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”
Example…

Example:
Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: “hello”

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Example...

Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Example:
Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Learning via Backpropagation...

Backpropagation through time

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Learning via Backpropagation...

**Truncated Backpropagation through time**

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Learning via Backpropagation...

Truncated Backpropagation through time

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Learning via Backpropagation…

**Truncated** Backpropagation through time

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
“The Unreasonable Effectiveness of Recurrent Neural Networks”

karpathy.github.io
Character-level language model example

Vocabulary: [h,e,l,o]

Example training sequence: “hello”

\[ h_{t+1} = \tanh(W_{hh}h_t + W_{xh}x_t) \]
Sonnet 116 – Let me not ...

by William Shakespeare

Let me not to the marriage of true minds
Admit impediments. Love is not love
Which alters when it alteration finds,
Or bends with the remover to remove:
O no! it is an ever-fixed mark
That looks on tempests and is never shaken;
It is the star to every wandering bark,
Whose worth's unknown, although his height be taken.
Love's not Time's fool, though rosy lips and cheeks
Within his bending sickle's compass come:
Love alters not with his brief hours and weeks,
But bears it out even to the edge of doom.
If this be error and upon me proved,
I never writ, nor no man ever loved.
<table>
<thead>
<tr>
<th>Part</th>
<th>Chapter</th>
<th>online</th>
<th>TeX source</th>
<th>view pdf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminaries</td>
<td>1. Introduction</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
<tr>
<td></td>
<td>2. Conventions</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
<tr>
<td></td>
<td>3. Set Theory</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
<tr>
<td></td>
<td>4. Categories</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
<tr>
<td></td>
<td>5. Topology</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
<tr>
<td></td>
<td>6. Sheaves on Spaces</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
<tr>
<td></td>
<td>7. Sites and Sheaves</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
<tr>
<td></td>
<td>8. Stacks</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
<tr>
<td></td>
<td>9. Fields</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
<tr>
<td></td>
<td>10. Commutative Algebra</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
</tbody>
</table>

### 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
For $\bigoplus_{i=1,\ldots,m} \mathcal{L}_{m_i} = 0$, 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 comparably in the fibre product covering we have to prove the lemma generated by $\prod Z \times_U U \to V$. Consider the maps $M$ along the set of points $\text{Sch}_{pfp}$ 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 $\text{Sh}(G)$ such that $\text{Spec}(R') \to S$ is smooth or an

$$U = \bigcup_i U_i \times S, U_i$$

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 $GL_{S'}(x'/s'')$ and we win.

To prove study we see that $\mathcal{F}/\mathcal{U}$ is a covering of $\mathcal{X}'$, and $U_0$ is an object of $\mathcal{F}/\mathcal{U}$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_{\mathcal{X}}$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module.

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

$$\widetilde{M}^* = \mathcal{M}^* \otimes_{\text{Spec}(R)} \mathcal{O}_{S', \ast} - 1 \ast \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)_{pfpf}, (\text{Sch}/S)_{ppf}$$

and

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

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 $X_{\text{space, etale}}$ which gives an open subspace of $X$ and $T$ equal to $S_{\text{zar}}$, 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 $\text{Proj}_X(A) = \text{Spec}(B)$ 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 $\mathcal{Q} \to \mathcal{C}_{X, \mathcal{O}_X}$ 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 $S = \text{Spec}(R)$ and $Y = \text{Spec}(R)$.

Proof. This is form all sheaves of sheaves in $X$. But given a scheme $U$ and a surjective étale morphism $U \to X$. Let $U \cap U = \prod_{i=1,\ldots,n} U_i$ be the scheme $X$ over $S$ at the schemes $X_i \to X$ and $U = \lim_i X_i$.

The following lemma surjective restrikrecomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{x_{i=0}}$.

Lemma 0.2. Let $X$ be a locally Noetherian scheme over $S$, $E = \mathcal{F}_{\mathcal{X}/S}$. Set $\mathcal{I} = \mathcal{I}_i \subset \mathcal{I}_n$. Since $\mathcal{T}_n \subset \mathcal{T}_n$ are nonzero over $i \leq p$ is a subset of $\mathcal{J}_n \circ \mathcal{A}_2$ 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

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

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

**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 $O$-modules. We have to show that

$$O_{O_X} = O_X(L)$$

Proof. This is an algebraic space with the composition of sheaves $F$ on $X_{etale}$ we have

$$O_X(F) = \{ \text{morph }_1 \times O_X(G, F) \}$$

where $G$ defines an isomorphism $F \to F$ of $O$-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.

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 $O_X$-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 $O_X(U)$ which is locally of finite type.
Try it yourself: **char-rnn** on Github (uses Torch7)
Cooking
Recipes

Title: BASIC CHEESE WINGS:
Categories: Desserts
Yield: 6 Servings

3 Eggs
2 tb Chopped fresh curry
   -or cooking spray
1 c Water; cooked
2 Lemons minced mushrooms
3 oz Sweet cooked rice
1/2 Onion; chopped
3 c Butter, melted
2 ts Soy sauce
1 ts Cinnamon
2 md Sugar or food coloring;
   -stems cored bowl
2 tb Salt and freshly grated
1/4 ts Ground ginger
1/2 c Flour
1 tb Water; fresh parsley
1 c Water (or or)
1 Clove garlic, minced

Preheat oven to 350F. Combine sugar, salt, baking soda, celery and sugar. Add
the chicken broth well. Add the cornstarch to the pan; cool. Add the olive
oil, oil, and basil or cooking spray. Pour the onions until melted.
Good afternoon. God bless you.

The United States will step up to the cost of a new challenges of the American people that will share the fact that we created the problem. They were attacked and so that they have to say that all the task of the final days of war that I will not be able to get this done. The promise of the men and women who were still going to take out the fact that the American people have fought to make sure that they have to be able to protect our part. It was a chance to stand together to completely look for the commitment to borrow from the American people. And the fact is the men and women in uniform and the millions of our country with the law system that we should be a strong strects of the forces that we can afford to increase our spirit of the American people and the leadership of our country who are on the Internet of American lives.

Thank you very much. God bless you, and God bless the United States of America.
32:22 And they shall be the children of Israel, and they that shall come upon us, that they may be their God.

2:11 Therefore shall they see thy chastisement for them, they shall live: I will sing praise to thee in the night thy servant.

8:26 And they set the book of the law which Michal the Baptist came near to Man.
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 << i))
            pipe = (in_use & UMXTHREAD_UNCCA) +
            ((count & 0x00000000ffffff8) & 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, &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 ");
}
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 & 0x00000000fffffff8L) & 0x0000000fL) << 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/cpufreq.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/setew.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));  \
    if (__type & DO_READ)

static void stat_PC_SEC __read_mostly offsetof(struct seq_argsqueue,
   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)-l->lr_full; low;
    }

Yoav Goldberg n-gram experiments

Order 10 ngram model on Shakespeare:

APEMANTUS:
Ho, ho! I laugh to see your beard!

BOYET:
Madam, in great extremes of passion as she discovers it.

PAROLLES:
By my white head and her wit
Values itself: to the sepulchre!
With this, my lord,
That I have some business: let's away.

First Keeper:
Forbear to murder: and wilt thou not say he lies,
And lies, and let the devil would have said, sir, their speed
Hath been balm to heal their woes,
But on Linux:

```c
~/*
  * linux/kernel/time.c
  * Please report this on hardware.
  */
void irq_mark_irq(unsigned long old_entries, eval);

  /*
   * Divide only 1000 for ns^2 -> us^2 conversion values
don't overflow:
   seq_puts(m, "\ttramp: %ps",
            (void *)class->contending_point
]++;
   if (likely(t->flags & WQ_UNBOUND)) {
      /*
       * Update inode information. If the
       * slowpath and sleep time (abs or rel)
       * @rmtip: remaining (either due
       * to consume the state of ring buffer size. */
       * header_size - size, in bytes, of the chain.
       */
       BUG_ON(!error);
   } while (cgrp) {
   if (old) {
      if (kdb_continue_catastrophic;
  #endif
```
“straw hat”
training example
“straw hat”

training example
“straw hat”

training example
“straw hat” training example
before: \[ h_0 = \tanh(0, W_{xh} \ast x_0) \]

now: \[ h_0 = \tanh(0, W_{xh} \ast x_0 + W_{ih} \ast v) \]
<START>

sample!

<END> token

=> finish.
Wow I can’t believe that worked

a group of people standing around a room with remotes
logprob: -9.17

a young boy is holding a baseball bat
logprob: -7.61

a cow is standing in the middle of a street
logprob: -8.84
Wow I can’t believe that worked

a cat is sitting on a toilet seat
logprob: -7.79

a display case filled with lots of different types of donuts
logprob: -7.78

a group of people sitting at a table with wine glasses
logprob: -6.71
Well, I can kind of see it

- A man standing next to a clock on a wall
  logprob: -10.08

- A young boy is holding a baseball bat
  logprob: -7.65

- A cat is sitting on a couch with a remote control
  logprob: -12.45

slide credit: Jeff Donahue
Well, I can kind of see it
Not sure what happened there...

- a toilet with a seat up in a bathroom, logprob: -13.44
- a woman holding a teddy bear in front of a mirror, logprob: -9.65
- a horse is standing in the middle of a road, logprob: -10.34
Vanilla RNN...

Vanilla RNN Gradient Flow

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

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Vanilla RNN...

Vanilla RNN Gradient Flow

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

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

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Vanilla RNN...

Vanilla RNN Gradient Flow

Computing gradient of $h_0$ involves many factors of $W$ (and repeated tanh)

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Vanilla RNN...

Vanilla RNN Gradient Flow

Computing gradient of $h_0$ involves many factors of $W$ (and repeated tanh)

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Vanilla RNN...

Vanilla RNN Gradient Flow

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

Computing gradient of $h_0$ involves many factors of $W$ (and repeated tanh)

Largest singular value $> 1$: Exploding gradients
Largest singular value $< 1$: Vanishing gradients

Gradient clipping: Scale gradient if its norm is too big

```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
    grad *= (threshold / grad_norm)
```
Vanilla RNN...

Vanilla RNN Gradient Flow

Computing gradient of $h_0$ involves many factors of $W$ (and repeated tanh)

Largest singular value $> 1$: Exploding gradients

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
**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 \left( h_{t-1}, x_t \right) \\

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]

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Long Short Term Memory (LSTM)

Long Short Term Memory (LSTM)  
[Hochreiter et al., 1997]

i: **Input gate**, whether to write to cell  
f: **Forget gate**, Whether to erase cell  
o: **Output gate**, How much to reveal cell  
g: **Gate gate** (?), How much to write to cell

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

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

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Long Short Term Memory (LSTM)

Long Short Term Memory (LSTM)
[Hochreiter et al., 1997]

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Long Short Term Memory (LSTM): Gradient Flow

Long Short Term Memory (LSTM)  
[Hochreiter et al., 1997]

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

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

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Long Short Term Memory (LSTM): Gradient Flow

Long Short Term Memory (LSTM)
[Hochreiter et al., 1997]

Uninterrupted gradient flow!

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Long Short Term Memory (LSTM): Gradient Flow

Long Short Term Memory (LSTM)
[Hochreiter et al., 1997]

Uninterrupted gradient flow!

Similar to ResNet!

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Long Short Term Memory (LSTM): Gradient Flow

Long Short Term Memory (LSTM)  
[Hochreiter et al., 1997]

Uninterrupted gradient flow!

In between:  
Highway Networks

\[ g = T(x, W_T) \]
\[ y = g \odot H(x, W_H) + (1 - g) \odot x \]

Srivastava et al, "Highway Networks", 
ICML DL Workshop 2015

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Alternatives

Other RNN Variants

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

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

[**LSTM**: A Search Space Odyssey, Greff et al., 2015]

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

---

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
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
- Better understanding (both theoretical and empirical) is needed.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung
Visualizing and Understanding Recurrent Networks
Andrej Karpathy*, Justin Johnson*, Li Fei-Fei
(on arXiv.org)
Hunting interpretable cells

\[
\begin{pmatrix}
i \\ f \\ o \\ g
\end{pmatrix} =
\begin{pmatrix}
\text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{tanh}
\end{pmatrix}
W^l
\begin{pmatrix}
h^{l-1}_t \\ h^l_t
\end{pmatrix}
\]

\[
c^l_t = f \circ c^{l-1}_t + i \circ g
\]

\[
h^l_t = o \circ \tanh(c^l_t)
\]
Hunting interpretable cells

/* Unpack a filter field's string representation from user-space buffer. */
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. */
}
Hunting 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
Hunting 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
Hunting interpretable cells

```
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;
}
```
Hunting interpretable cells

```c
/* 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;
```
quote/comment cell
Hunting interpretable cells

code depth cell
Hunting interpretable cells

```c
char *audit_unpack_string(void **bump, size_t *remain, si
{
    char *str;
    if (!(bump || (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(-ENAMEETOLONG);
    str = kmalloc(len + 1, GFP_KERNEL);
    if (unlikely(!str))
        return ERR_PTR(-ENOMEM);
    memcpy(str, *bump, len);
    str[len] = 0;
    *bump += len;
    *remain -= len;
    return str;
```