High Level Computer Vision

Deep Learning for Computer Vision
Part 4

Bernt Schiele - schiele@mpi-inf.mpg.de
Mario Fritz - mfritz@mpi-inf.mpg.de

https://www.mpi-inf.mpg.de/hlcv
Overview

• Recurrent Neural Networks
  ▸ motivation for recurrent neural networks
  ▸ a particularly successful RNN: Long Short Term Memory (LSTM)
  ▸ slide credit to Andrej Karpathy, Jeff Donahue and Marcus Rohrbach

• Yann LeCun…
  ▸ What’s Wrong With Deep Learning (keynote June 2015)
  ▸ slide credit to Yann LeCun (and Xiaogang Wang)
Recurrent Networks offer a lot of flexibility:
Sequences in Vision

Sequences in the input… (many-to-one)

- Running
- Jumping
- Dancing
- Fighting
- Eating

slide credit: Jeff Donahue
Sequences in Vision

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

A happy brown dog.

slide credit: Jeff Donahue
Sequences in Vision

Sequences everywhere!
(many-to-many)

A dog jumps over a hurdle.
ConvNets

Krizhevsky et al., NIPS 2012
Problem #1

fixed-size, static input

slide credit: Jeff Donahue
Problem #1

fixed-size, static input
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
- [x] a big brown dog
- [ ] a happy red dog
- [ ] a big red dog
- [ ] ...

slide credit: Jeff Donahue
Recurrent Networks offer a lot of flexibility:
Language Models

Word-level language model. Similar to:

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: 
\[ P(\text{cat} \mid [<S>]) \] 
\[ P(\text{sat} \mid [<S>, \text{cat}]) \] 
\[ P(\text{on} \mid [<S>, \text{cat}, \text{sat}]) \] 
\[ P(\text{mat} \mid [<S>, \text{cat}, \text{sat}, \text{on}]) \] 
\[ P(<E> \mid [<S>, \text{cat}, \text{sat}, \text{on}, \text{mat}]) \]
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”

300 (learnable) numbers associated with each word in vocabulary

hidden layer (e.g. 500-D vectors)

\[ h_4 = \tanh(0, W_{xh} \ast x_4) \]
"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)
\[ h_4 = \text{tanh}(0, W_{xh} \times x_4) \]

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

“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)
\[ h_4 = \tanh(0, W_{xh} \times x_4 + W_{hh} \times h_3) \]

300 (learnable) numbers associated with each word in vocabulary
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

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

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

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

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

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

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

\[ P(\text{next word} \mid \text{previous words}) \]
“straw hat”

training example
“straw hat”

training example
“straw hat”

training example
“straw hat”

training example

<START> straw hat
before: \[ h_0 = \tanh(0, W_{xh} \times x_0) \]

now: \[ h_0 = \tanh(0, W_{xh} \times x_0 + W_{ih} \times v) \]
test image

sample!

slide credit: Andrej Karpathy
test image
slide credit: Andrej Karpathy
test image
<START>

sample!

<END> token

=> finish.

slide credit: Andrej Karpathy
Sequence Learning

- Instances of the form $\mathbf{x} = \langle x_1, x_2, x_3, \ldots, x_T \rangle$

- Variable sequence length $T$

- Learn a transition function $f$ with parameters $W$:
  
- $f$ should update hidden state $h_t$ and output $y_t$

\[
\begin{align*}
  h_0 & := 0 \\
  \text{for } t = 1, 2, 3, \ldots, T: \\
  \langle y_t, h_t \rangle & = f_{W}(x_t, h_{t-1})
\end{align*}
\]
Sequence Learning

Equivalent to a T-layer deep network, unrolled in time

\[ f \]

\[ 0 \rightarrow f \rightarrow h_1 \rightarrow f \rightarrow h_2 \rightarrow \ldots \rightarrow h_{T-1} \rightarrow f \rightarrow h_T \]

\[ X_1 \]
\[ Z_1 \]

\[ X_2 \]
\[ Z_2 \]

\[ X_T \]
\[ Z_T \]
Sequence Learning

• What should the transition function $f$ be?

At a minimum, we want something **non-linear** and **differentiable**.
Sequence Learning

• First attempt — a “vanilla” RNN:

\[ h_t = \sigma(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \]

\[ z_t = \sigma(W_{hz}h_t + b_z) \]

• Problems

  • Difficult to train — vanishing/exploding gradients
  • Unable to “select” inputs, hidden state, outputs
Sequence Learning

• LSTM - Long Short Term Memory
  [Hochreiter & Schmidhuber, 1997]
• Selectively propagate or forget hidden state
• Allows long-term dependencies to be learned

• Effective for
  • speech recognition
  • handwriting recognition
  • translation
  • parsing

![LSTM Unit Diagram](image)
LSTM for sequence modeling

\[
\begin{align*}
    i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
    f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\
    o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
    g_t &= \phi(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
    c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
    h_t &= o_t \odot \phi(c_t)
\end{align*}
\]
LSTM for sequence modeling

\[ i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \]
\[ f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \]
\[ o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \]
\[ g_t = \phi(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \]
\[ c_t = f_t \odot c_{t-1} + i_t \odot g_t \]
\[ h_t = o_t \odot \phi(c_t) \]
Sequence Learning

Exactly remember previous cell state — discard input

LSTM
(Hochreiter & Schmidhuber, 1997)
Sequence Learning

Forget previous cell state — only use modulated input

LSTM
(Hochreiter & Schmidhuber, 1997)
LRCN

• Long-term Recurrent Convolutional Networks

• End-to-end trainable framework for sequence problems in vision
Image Description

Sequential output

CNN → LSTM → LSTM → LSTM → LSTM → LSTM

<BOS> → a → dog → is → jumping → <EOS>

slide credit: Jeff Donahue
Image Description

sequential output

= embed a one-hot vector
Image Description

Single LSTM layer

= embed a one-hot vector

slide credit: Jeff Donahue
two LSTM layers

CNN

LSTM

LSTM

LSTM

LSTM

LSTM

LSTM

<LSTM>

a
dog
is
jumping
<EOS>

Image Description
two LSTM layers, factored
# Image Description

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Flickr30k [1] Caption-to-Image Recall@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Layer</td>
<td>14.1%</td>
</tr>
<tr>
<td>Two Layer</td>
<td>3.8%</td>
</tr>
<tr>
<td>Two Layer, Factored</td>
<td>17.5%</td>
</tr>
<tr>
<td>Four Layer, Factored</td>
<td>15.8%</td>
</tr>
</tbody>
</table>

# Image Description

<table>
<thead>
<tr>
<th></th>
<th>CaffeNet</th>
<th>VGGNet</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Raw</strong></td>
<td>68.8%</td>
<td>77.3%</td>
</tr>
<tr>
<td><strong>Finetuned</strong></td>
<td>75.8%</td>
<td><strong>83.9%</strong></td>
</tr>
</tbody>
</table>

---


A female tennis player in action on the court.

A group of young men playing a game of soccer.

A man riding a wave on top of a surfboard.
Image Description

A black and white cat is sitting on a chair.

A large clock mounted to the side of a building.

A bunch of fruit that are sitting on a table.
Activity Recognition

CNN, CNN, CNN, CNN

LSTM, LSTM, LSTM, LSTM

studying, running, jumping, jumping

Average

jumping

sequential input

slide credit: Jeff Donahue
# Activity Recognition

<table>
<thead>
<tr>
<th></th>
<th>UCF101 Class. Acc.</th>
<th>RGB</th>
<th>Optical Flow</th>
<th>RGB+ Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single-Frame CNN</strong></td>
<td>67.7%</td>
<td>72.2%</td>
<td>78.8%</td>
<td></td>
</tr>
<tr>
<td><strong>LRCN</strong></td>
<td>68.2%</td>
<td>77.5%</td>
<td>82.7%</td>
<td></td>
</tr>
</tbody>
</table>

Video Description

sequential input & output
Video Description

MPII TACoS Multi-Level Dataset

Video Description

Pre-trained Detector Predictions

LSTM  LSTM  LSTM  LSTM  LSTM

a  man  cuts  vegetables  <EOS>

vegetables cuts man a

slide credit: Jeff Donahue
Video Description

<table>
<thead>
<tr>
<th>Approach</th>
<th>Generation Accuracy (BLEU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMT</td>
<td>26.9%</td>
</tr>
<tr>
<td>LRCN</td>
<td>28.8%</td>
</tr>
</tbody>
</table>

slide credit: Jeff Donahue
Video Description

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, Kate Saenko.
Wow I can’t believe that worked
Wow I can’t believe that worked
Well, I can kind of see it
Well, I can kind of see it
Not sure what happened there...

- A toilet with a seal 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
“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.
The Stacks Project

### Browse chapters

<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>Introduction</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
<tr>
<td>2.</td>
<td>Conventions</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
<tr>
<td>3.</td>
<td>Set Theory</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
<tr>
<td>4.</td>
<td>Categories</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
<tr>
<td>5.</td>
<td>Topology</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
<tr>
<td>6.</td>
<td>Sheaves on Spaces</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
<tr>
<td>7.</td>
<td>Sites and Sheaves</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
<tr>
<td>8.</td>
<td>Stacks</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
<tr>
<td>9.</td>
<td>Fields</td>
<td>online</td>
<td>tex</td>
<td>pdf</td>
</tr>
<tr>
<td>10.</td>
<td>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)
  - 2356 sections
Lemma 0.1. Assume (3) and (3) by the construction in the description. Suppose $X = \varprojlim X_i$ (by the formal open covering $X$ and a single map $\overline{\text{Proj}}(A) = \text{Spec}(R)$ over $U$ compatible with the complex $\text{Set}(A) = \Gamma(X, \mathcal{O}_{X,T})$).

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

The following lemma surjective retrocomposes of this implies that $F_{\geq} = F_{X_i}$.

Lemma 0.2. Let $X$ be a locally Noetherian scheme over $S$, $E = \mathcal{F}_{X/S}$. Set $\mathcal{F} = \mathcal{F}_X \subset T'$. Since $T' \subset T^\text{aff}$ are nonzero over $i \leq p$ is a subset of $T'_{\geq 0} = \text{Aff}_X$.

Lemma 0.3. In Situation ??, hence we may assume $q' = 0$.

Proof. We will use the property we see that $p$ is the most functor (??). On the other hand, by Lemma ?? we see that $D(O_{X,S}) = O_X(D)$ where $K$ is an $F$-algebra where $\delta_{n-1}$ is a scheme over $S$. 

The result for prove any open covering follows from the less of Example ??, it may replace $S$ by $X_{\text{spars}, finite}$ 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 $S$. 

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 $Z \times_X U \to V$. Consider the maps $\mathcal{M}$ along the set of points $\text{Spec}(R')$ 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 on an

$$V = \bigcup U_i \times_X U_i$$

where has a nonzero morphism we may assume that $f_i$ is of finite presentation over $S$. We claim that $\mathcal{O}_X$ is a scheme where $x, x', x'' \in S$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}_{X,x''}$ is separated. By Algebra, Lemma ?? we can define a map of the finite presentation $\mathcal{O}_{X,x'} \to \mathcal{O}_{X,x''}$ we win.

To prove this we see that $\mathcal{F}_{\geq i}$ is a covering of $\mathcal{X}$, and $\mathcal{F}_i$ is the object of $\mathcal{F}_{\geq i}$ for $i > 0$ and $\mathcal{F}_i$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{C}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\mathcal{M}^\mathbb{C} = \mathcal{M} \times_{\text{Spec}(A)} \mathcal{C}_\mathbb{C} \to \mathcal{M}$$

is a unique morphism of algebraic stacks. Note that

$$\overline{\text{Arrows}} = \mathcal{C}_\mathbb{C}^{\text{op}} \mathcal{C}_{\text{test}}$$

and

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

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.
Proof. Omitted.

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

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

$$O_{\mathcal{C}} = O_X(\mathcal{C})$$

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

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

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \rightarrow \mathcal{F}$ 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. 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

$$\delta : X \rightarrow Y' \rightarrow Y \rightarrow Y' \times_X Y \rightarrow X$$

be an isomorphism of algbraic spaces over $S$ and $Y$.

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

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

Consider a common structure on $X$ and $X$ the functor $O_X(U)$ which is locally of finite type.

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

![Diagram]

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

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

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

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.

A reduced above we conclude that $U$ is an open covering of $C$. The functor $\mathcal{F}$ is a "field"

$$O_{X,u} \rightarrow \mathcal{F}_u \rightarrow (O_{X,u})_0 \rightarrow O_{X,u}^1 O_{X,u}(O_{X,u})$$

is an isomorphism of covering of $O_X$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that $X$ is an isomorphism.

The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $O_X$-algebra with $\mathcal{F}$ are open of finite type over $S$.

If $\mathcal{F}$ is a coherent theoretic image point.

If $\mathcal{F}$ is a finite direct sum $O_{X,u}$ is a closed immersion, see Lemma ?? This is a sequence of $\mathcal{F}$ is a similar morphism.
Try it yourself: **char-rnn** on Github
(uses Torch7)
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 Grounds ginger
1/2 c Flour
1 tb Water; fresh parsley
1 c Water (or ow)
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 melts.
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 stretches 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.
Linux kernel source tree

<table>
<thead>
<tr>
<th>Branch</th>
<th>Name</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>linux</td>
<td>Merge branch 'dtr-fixes' of git://people.freedesktop.org/~airlied/linux</td>
</tr>
</tbody>
</table>

- **520,037 commits**
- **1 branch**
- **420 releases**
- **5,839 contributors**

---

**Documentation**
- Merge git://git.kernel.org/pub/scm/linux/kernel/git/nab/target-pending
- Merge branch 'x86-urgent-lor-linus' of git://git.kernel.org/pub/scm/
- block: discard bdi_unregister() in favour of bdi_destroy()
- Merge git://git.kernel.org/pub/scm/linux/kernel/git/herbert/crypto-2.6
- Merge branch 'dtr-fixes' of git://people.freedesktop.org/~airlied/linux
- firmware/fix 2w.c: restore missing dev/ssl it in switch statement
- fe: vfs: read file_handle only once in handle_to_path
- Merge branch 'perf-urgent-for-linus' of git://git.kernel.org/pub/scm/
- int: fix regression by supporting devices with major/minor:offset: fo...
- Merge branch 'for-linus' of git://git.kernel.org/pub/scm/linux/kernel
static void do_command(struct seq_file *s, 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 & UMTHREAD_UNCC) +
            ((count & 0xffffffffffffff8) & 0x0000000f) << 8;
        if (count == 0)
            sub(pid, ppc_rd.kexec_handle, 0x2000000);
        pipe_set_bytes(i, 0);
    }
    /* Free our user pages pointer to place camera if all dash */
    subsystem info = &of_changes[PAGE_SIZE];
    rek_controles(offset, idx, &s_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 ");
}
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 & UNXTHREAD_UNCCCA) +
            ((count & 0x00000000ffffffff0) << 8);
        if (count == 0)
            sub(pid, ppc_rd.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, &scoffset);
    /* 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/cr/event.h>

#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/sew.h>
#include <asm/pgproto.h>
#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/segreg.h>
#include <asm/pgtable.h>

#define REG_PG versa_slot_addr_pack
#define CFM_RUN_CNTR APSR(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_argqueue, \
    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, lcow;
    }
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)
               * @rmtpl: 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;
```
Visualizing and Understanding Recurrent Networks
Andrej Karpathy*, Justin Johnson*, Li Fei-Fei
(on arXiv.org as of June 2015)
Hunting interpretable cells

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

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

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

---

LSTM Unit

Input Gate \(\sigma\)

Output Gate \(\sigma\)

Input Modulation Gate \(\phi\)

Forget Gate \(\sigma\)
Hunting 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. */
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

The sole importance of the crossing of the Derezina 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.
Hunting interpretable cells

Cell sensitive to position in line:

The sole importance of the crossing of the Derevina 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.
Hunting interpretable cells

```c
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;
}
```

if statement cell
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;
}
```
Hunting interpretable cells

```c
#include <config.auditsyscall.h>
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
Hunting interpretable cells

```c
char *audit_unpack_string(void **bfp, size_t *remain, size_t len)
{
    char *str;
    if (!*bfp || (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, *bfp, len);
    str[len] = 0;
    *bfp += len;
    *remain -= len;
    return str;
}
```

something interesting cell
(not quite sure what)
What's Wrong With Deep Learning?

Yann LeCun
Facebook AI Research &
Center for Data Science, NYU
yann@cs.nyu.edu
http://yann.lecun.com
Why are ConvNets a good architecture?
- Scattering transform
  Mark Tygert’s “complex ConvNet”

How many layers do we really need?
Really?

How many effective free parameters are there in a large ConvNet
The weights seem to be awfully redundant

What about Local Minima?
- Turns out almost all the local minima are equivalent
  Local minima are degenerate (very flat in most directions)
- Random matrix / spin glass theory comes to the rescue
  [Choromanska, Henaff, Mathieu, Ben Arous, LeCun AI-stats 2015]
Missing: Reasoning
System outline

Input

Deep Convolutional Neural Network

Aeroplane Coarse Score map

Fully Connected CRF

Bi-linear Interpolation

Final Output

J. Long, E. Shelhamer, T. Darrell, FCNNs for Semantic Segmentation, CVPR 15
P. Krähenbühl and V. Koltun, Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials, NIPS 2011
Deep Learning systems can be assembled into energy models AKA factor graphs:
- Energy function is a sum of factors.
- Factors can embed whole deep learning systems.
- $X$: observed variables (inputs).
- $Z$: never observed (latent variables).
- $Y$: observed on training set (output variables).

Inference is energy minimization (MAP) or free energy minimization (marginalization) over $Z$ and $Y$ given an $X$:
- $F(X,Y) = \text{MIN}_z E(X,Y,Z)$
- $F(X,Y) = -\log \text{SUM}_z \exp[-E(X,Y,Z)]$
Missing: Unsupervised Learning
An autoencoder takes an input $x \in [0, 1]^d$ and first maps it (with an encoder) to a hidden representation $y \in [0, 1]^d'$ through a deterministic mapping

$$y = s(Wx + b)$$

where $s$ is a non-linear activation function (such as sigmoid).

$y$ is mapped back (with a decoder) into a reconstruction $z$ of the same shape as $x$,

$$z = s(W'y + b')$$

$z$ is seen as a prediction of $x$. 
Autoencoder

- Encoder
  \[ y = f_\theta(x) \]

- Decoder
  \[ z = g_\theta(y) \]
  \[ \theta = \{W, W', b, b'\} \]

- It is important to add regularization in the training criterion or the parametrization to prevent the auto-encoder from learning the identity function, which would lead to zero reconstruction error everywhere.

- A particular form of regularization consists in constraining the code to have a low dimension, and this is what the classical auto-encoder or PCA do.
Deep autoencoder

Stack multiple encoders (and their corresponding decoders)

- 28x28
- 1000 neurons
- 500 neurons
- 250 neurons
- 30
- 250 neurons
- 500 neurons
- 1000 neurons
- 28x28
Deep autoencoder

- Very difficult to optimize deep autoencoders using backpropagation
- Pre-training + fine-tuning
  - First train a stack of RBMs
  - Then “unroll” them
  - Then fine-tune with backpropagation
Comparison of methods of compressing images

real data
30-D deep auto
30-D logistic
PCA

real data
30-D deep auto
30-D PCA

slide credit: Xiaogang Wang