CS 4803 / 7643: Deep Learning

Topics:
- Recurrent Neural Networks (RNNs)
  - RNN visualizations
  - Image Captioning, Beam Search
  - LSTMs

Dhruv Batra
Georgia Tech
Administrativia

• **HW3 Reminder**
  – Due: 10/20 11:59pm
  – Theory: Convolutions, Representation Capacity, Double Descent
  – Implementation: Saliency methods (e.g. Grad-CAM) in Python and PyTorch/Captum

• **HW2 grades coming soon**
Administrativia

• Guest Lecture: Ishan Misra (FAIR)
  – Thurs 10/21
  – Self-Supervised Learning for Vision

http://imisra.github.io/
Administrativia

• Guest Lecture: Michael Auli (FAIR)
  – Tue 10/26
  – Self-Supervised Learning for Speech

https://michaelauli.github.io/
Administrativia

• Guest Lecture: Arjun Majumdar
  – Thurs 10/28
  – Transformers, BERT, ViLBERT

https://arjunmajum.github.io/
Recap from last time
Recurrent Neural Network

usually want to predict a vector at some time steps

“cell”

“state”

Recurrent unit of computation
\[ P(y_t | y^{t-1} x_t) \approx P(y_t | h_t) \]

**(Vanilla) Recurrent Neural Network**

The state consists of a single "hidden" vector \( h \):

\[
y_t = \text{softmax}(W_{hy} h_t + b_y)
\]

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

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Example:
Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: “hello”
Training Time: \( \text{MLE} \) / “Teacher Forcing”

Example:
Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: “hello”

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Test Time: Sample / Argmax / Beam Search

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 Li, Justin Johnson, Serena Yeung, CS 231n
Test Time: Sample / Argmax / Beam Search

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
Plan for Today

• Recurrent Neural Networks (RNNs)
  – (Finish) Visualization in Character RNNs
  – Inference: Beam Search
    • Example: Image Captioning
  – Multilayer RNNs
  – Problems with gradients in “vanilla” RNNs
  – LSTMs (and other RNN variants)
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 thinke own bud buried thy content,
And tender churl mak'st waste in niggarding:
Plie the world, or else this gluton 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 tattered weed of small worth held;
Then being asked, where all thy beauty lies,
Where all the treasure of thy lusty days;
To say, within thinke own deep sunken eyes,
Were an all-eating shame, and thriftless praise.
How much more praise deserv'd thy 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.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
at first:

"Tmont thithey" formesserliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogenn Phe lism thond hon at. MeiDimorotion 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 overetical and ofter.

"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```c
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 & 0x00000000fffffff8) & 0x000000f) << 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_contols(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 ");
}
```
/*
 * 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/setew.h>
#include <asm/pgproto.h>
Searching for interpretable cells
Searching for interpretable cells

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
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."

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Searching for interpretable cells

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

Karin, Johnson, and Fei-Fei: CS 231n
Searching for interpretable cells

static int __dequeu_e_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

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Searching for interpretable cells

```
   Cell that turns on inside comments and quotes:
   /* 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 */
      lsm_rule = 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
Searching for interpretable cells

```c
#define 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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Plan for Today

• Recurrent Neural Networks (RNNs)
  – Inference: Beam Search
    • Example: Image Captioning
  – Multilayer RNNs
  – Problems with gradients in “vanilla” RNNs
  – LSTMs (and other RNN variants)
Neural Image Captioning

Image Embedding (VGGNet) -> RNN

Convolution Layer + Non-Linearity
Pooling Layer
Convolution Layer + Non-Linearity
Pooling Layer
Fully-Connected MLP

4096-dim
Neural Image Captioning

Image Embedding (VGGNet)
Neural Image Captioning

P(next) P(next) P(next) P(next) P(next) P(next)

Linear RNN RNN RNN RNN RNN RNN

\[ h_t = \tanh \left( W_m h_{t-1} + W_x x_t + b_h \right) \]

Two people and two horses.

(C) Dhruv Batra
Beam Search Demo

- http://dbs.cloudcv.org/captioning&mode=interactive
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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Image Captioning: Failure Cases

A woman is holding a cat in her hand

A person holding a computer mouse on a desk

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

Captions generated using neuralTalk2
All images are CC0 Public domain: fur, coat, handstand, spider web, baseball

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Image Captioning with Attention


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Image Captioning with Attention


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Image Captioning with Attention


\[ z = \sum_{i=1}^{L} p_i v_i \]
Image Captioning with Attention

Image Captioning with Attention

Image Captioning with Attention


Image: H x W x 3
Image Captioning with Attention

Image Captioning with Attention

Figure copyright Kelvin Xu, Jimmy Lei Ba, Jamie Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio, 2015. Reproduced with permission.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Image Captioning with Attention

Figure copyright Kelvin Xu, Jimmy Lei Ba, Jamie Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio, 2015. Reproduced with permission.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Typical VQA Models

Image Embedding (VGGNet)

Question Embedding (LSTM)

"How many horses are in this image?"
Visual Question Answering: RNNs with Attention

Figures from Zhu et al., copyright IEEE 2016. Reproduced for educational purposes.
Plan for Today

• Recurrent Neural Networks (RNNs)
  – Inference: Beam Search
    • Example: Image Captioning
  – Multilayer RNNs
  – Problems with gradients in “vanilla” RNNs
  – LSTMs (and other RNN variants)
Multilayer RNNs

\[ h_t^l = \tanh(W^l h_{t-1}^l) \]

\( h \in \mathbb{R}^n \)

\( W^l [n \times 2n] \)

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Vanilla RNN Gradient Flow

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

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 $W_{hh}$)

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

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

$$\frac{\partial h_t}{\partial h_0} = \left[ \frac{\partial h_t}{\partial h_{t-1}} \right] - \frac{\partial h_t}{\partial h_0} = W \cdot W \cdot \ldots \cdot W$$

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
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**
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

Gradient clipping: Scale gradient if its norm is too big

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
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
Long Short Term Memory (LSTM)

Vanilla RNN

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

LSTM

$$\begin{align*}
\begin{pmatrix}
i \\
f \\
o \\
g
\end{pmatrix} &= \begin{pmatrix}
\sigma \\
\sigma \\
\sigma \\
\tanh
\end{pmatrix} W \begin{pmatrix}
h_{t-1} \\
x_t
\end{pmatrix} \\
c_t &= f \odot c_{t-1} + i \odot g \\
h_t &= o \odot \tanh(c_t)
\end{align*}$$

Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation 1997
Meet LSTMs

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
LSTMs Intuition: Memory

- Cell State / Memory

\[ C_t : h_t \text{ hidden state} \]
\[ C_t \in \mathbb{R}^{512} \]
\[ h_t \in \mathbb{R}^d \]

(C) Dhruv Batra

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
LSTMs Intuition: Forget Gate

- Should we continue to remember this “bit” of information or not?

\[ f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \]
LSTMs Intuition: Input Gate

• Should we update this “bit” of information or not?
  – If so, with what?

\[
\begin{align*}
    i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\
    \tilde{C}_t &= \tanh(W_C[h_{t-1}, x_t] + b_C)
\end{align*}
\]
LSTMs Intuition: Memory Update

- Forget that + memorize this

\[
C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t
\]
LSTMs Intuition: Output Gate

- Should we output this “bit” of information to “deeper” layers?

\[ o_t \in \mathbb{R}^d \]

\[ o_t = \sigma(W_o [h_{t-1}, x_t] + b_o) \]

\[ h_t = o_t \times \text{tanh}(C_t) \]
LSTMs Intuition: Additive Updates

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

Uninterrupted gradient flow!
LSTMs Intuition: Additive Updates

Uninterrupted gradient flow!

Similar to ResNet!

(C) Dhruv Batra

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
LSTMs

• A pretty sophisticated cell
LSTM Variants #1: Peephole Connections

- Let gates see the cell state / memory

\[
\begin{align*}
  f_t &= \sigma (W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f) \\
  i_t &= \sigma (W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i) \\
  o_t &= \sigma (W_o \cdot [C_t, h_{t-1}, x_t] + b_o)
\end{align*}
\]
LSTM Variants #2: Coupled Gates

- Only memorize new if forgetting old

\[
C_t = f_t \cdot C_{t-1} + (1 - f_t) \cdot \tilde{C}_t
\]
LSTM Variants #3: Gated Recurrent Units

• Changes:
  – No explicit memory; memory = hidden output
  – $Z = \text{memorize new and forget old}$

\[
\begin{align*}
  z_t &= \sigma \left( W_z \cdot [h_{t-1}, x_t] \right) \\
  r_t &= \sigma \left( W_r \cdot [h_{t-1}, x_t] \right) \\
  \tilde{h}_t &= \tanh \left( W \cdot [r_t \ast h_{t-1}, x_t] \right) \\
  h_t &= (1 - z_t) \ast h_{t-1} + z_t \ast \tilde{h}_t
\end{align*}
\]
Other RNN Variants

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

MUT1:
\[
\begin{align*}
    z &= \text{sigm}(W_{xz}x_t + b_z) \\
    r &= \text{sigm}(W_{xr}x_t + W_{hr}h_t + b_r) \\
    h_{t+1} &= \text{tanh}(W_{hh}(r \odot h_t) + \text{tanh}(x_t) \odot b_h) \odot z \\
           &+ h_t \odot (1-z)
\end{align*}
\]

MUT2:
\[
\begin{align*}
    z &= \text{sigm}(W_{xz}x_t + W_{hz}h_t + b_z) \\
    r &= \text{sigm}(x_t + W_{hr}h_t + b_r) \\
    h_{t+1} &= \text{tanh}(W_{hh}(r \odot h_t) + W_{zh}x_t + b_h) \odot z \\
           &+ h_t \odot (1-z)
\end{align*}
\]

MUT3:
\[
\begin{align*}
    z &= \text{sigm}(W_{xz}x_t + W_{hz}h_t + b_z) \\
    r &= \text{sigm}(W_{xr}x_t + W_{hr}h_t + b_r) \\
    h_{t+1} &= \text{tanh}(W_{hh}(r \odot h_t) + W_{zh}x_t + b_h) \odot z \\
           &+ h_t \odot (1-z)
\end{align*}
\]
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.