9. Sequential Neural Models

CS 519 Deep Learning, Winter 2017
Fuxin Li

With materials from Andrej Karpathy, Bo Xie, Zsolt Kira
Sequential and Temporal Data

• Many applications exhibited by dynamically changing states
  – Language (e.g. sentences)
  – Temporal data
    • Speech
    • Stock Market
Image Captioning

"man in black shirt is playing guitar."
"construction worker in orange safety vest is working on road."
"two young girls are playing with lego toy."
"boy is doing backflip on wakeboard."
"girl in pink dress is jumping in air."
"black and white dog jumps over bar."
"young girl in pink shirt is swinging on swing."
"man in blue wetsuit is surfing on wave."
Machine Translation

• Have to look at the entire sentence (or, many sentences)

Economic growth has slowed down in recent years.

Das Wirtschaftswachstum hat sich in den letzten Jahren verlangsamt.

La croissance économique s'est ralentie ces dernières années.
Sequence Data

• Many data are sequences and have different inputs/outputs

(Image classification)

(Image captioning)

(Sentiment Analysis)

(Machine Translation)

(Video Classification)

(cf. Andrej Karpathy blog)
Previous: Autoregressive Models

• Autoregressive models
  – Predict the next term in a sequence from a fixed number of previous terms using “delay taps”.

• Neural Autoregressive models
  – Use neural net to do so
Previous: Hidden Markov Models

- Hidden states
- Outputs are generated from hidden states
  - Does not accept additional inputs
  - Discrete state-space
    - Need to learn all discrete transition probabilities!
Recurrent Neural Networks

• Similar to
  – Linear Dynamic Systems
    • E.g. Kalman filters
  – Hidden Markov Models
  – But not generative
• “Turing-complete”
Vanilla RNN Flow Graph

\[ a_t = b + W h_{t-1} + U x_t \]
\[ h_t = \tanh(a_t) \]
\[ y_t = c + V h_t \]

U – input to hidden
V – hidden to output
W – hidden to hidden
Examples

Now let’s look at some simple examples of RNNs.

This one sums its inputs:
Examples

This one determines if the total values of the first or second input are larger:
Finite State Machines

• Each node denotes a state
• Reads input symbols one at a time
• After reading, transition to some other state – e.g. DFA, NFA
• States = hidden units
The parity Example

Assume we have a sequence of binary inputs. We’ll consider how to determine the parity, i.e. whether the number of 1’s is even or odd.

We can compute parity incrementally by keeping track of the parity of the input so far:

Parity bits: \[0 \ 1 \ 1 \ 0 \ 1 \ 1 \ \rightarrow\]
Input: \[0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1\]

Each parity bit is the XOR of the input and the previous parity bit.
RNN Parity

- At each time step, compute parity between input vs. previous parity bit
- How to unroll the loop?
RNN Universality

• RNN can simulate any finite state machines
  — is Turing complete with infinite hidden nodes
    (Siegelmann and Sontag, 1995)
  — e.g., a computer (Zaremba and Sutskever 2014)

Training data:

```
Input:
j=8584
    for x in range(8):
        j+=920
    b=(1500+j)
    print((b+7567))
Target: 25011.
```

```
Input:
i=8827
    c=(i-5347)
    print((c+8704) if 2641<8500 else 5308)
Target: 12184.
```
RNN Universality

• Testing programs

Input:
\[
d = 8640;
\]
\[
\text{print}((7135 \text{ if } 6710 > ((d+7080)\times 14) \text{ else } 7200)).
\]

Target: 7200.
"Baseline" prediction: 7200.
"Naive" prediction: 7200.
"Mix" prediction: 7200.
"Combined" prediction: 7200.

Input:
\[
\text{print}(((841 \text{ if } 2076 < 7326 \text{ else } 1869) \times 10) \text{ if } 7827 < 317 \text{ else } 7192)).
\]

Target: 7192.
"Baseline" prediction: 7192.
"Naive" prediction: 7192.
"Mix" prediction: 7192.
"Combined" prediction: 7192.
RNN Universality
(if only you can train it!)

Input:
`print((4*7054)).`

Target: 28216.
"Baseline" prediction: 28216.
"Naive" prediction: 28116.
"Mix" prediction: 28216.
"Combined" prediction: 28216.

Input:
`print((4635-5257)).`

Target: -622.
"Baseline" prediction: -688.
"Naive" prediction: -628.
"Mix" prediction: -692.
"Combined" prediction: -632.

Input:
e=1079
for x in range(10): e+=4729
print(e).

Target: 48369.
"Baseline" prediction: 48017.
"Naive" prediction: 48011.
"Mix" prediction: 48101.
"Combined" prediction: 48009.

Input:
`print((8*(5051-648))).`

Target: 35224.
"Baseline" prediction: 34044.
"Naive" prediction: 32180.
"Mix" prediction: 33284.
"Combined" prediction: 33004.
One way to use RNNs to model text:

- Time 1 hidden units:
  - Input = "the"
  - Target = "quick"

- Time 2 hidden units:
  - Input = "quick"
  - Target = "brown"

- Time 3 hidden units:
  - Input = "brown"
  - Target = "fox"
Generate Text from RNN

One way to use RNNs to model text:

- target = "quick"
- target = "brown"
- target = "fox"

**Diagram:**

```
  time 1
  hidden units
  input = "the"
  time 2
  hidden units
  input = "quick"
  time 3
  hidden units
  input = "brown"

  time 4
  hidden units
  "fox"
```
RNN Sentence Model

• Hypothetical: Different hidden units for:
  – Subject
  – Verb
  – Object (different type)
<table>
<thead>
<tr>
<th>Query</th>
<th>cell 25</th>
<th>cell 26</th>
<th>cell 27</th>
<th>cell 30</th>
</tr>
</thead>
<tbody>
<tr>
<td>al yo yo sauce</td>
<td></td>
<td></td>
<td></td>
<td>diet</td>
</tr>
<tr>
<td>atkins diet lasagna</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>blender recipes</td>
<td></td>
<td>recipes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cake bakery edinburgh</td>
<td></td>
<td></td>
<td>bakery</td>
<td></td>
</tr>
<tr>
<td>canning corn beef hash</td>
<td></td>
<td></td>
<td>com, beef</td>
<td></td>
</tr>
<tr>
<td>torre de pizza</td>
<td></td>
<td></td>
<td>pizza</td>
<td></td>
</tr>
<tr>
<td>famous desserts</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fried chicken</td>
<td></td>
<td></td>
<td>chicken</td>
<td></td>
</tr>
<tr>
<td>smoked turkey recipes</td>
<td></td>
<td></td>
<td>recipes</td>
<td></td>
</tr>
<tr>
<td>italian sausage hoagies</td>
<td></td>
<td></td>
<td>sausage</td>
<td></td>
</tr>
<tr>
<td>do you get allergy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>much pain will after total knee replacement</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>how to make whiter teeth</td>
<td></td>
<td></td>
<td>whiter</td>
<td></td>
</tr>
<tr>
<td>illini community hospital</td>
<td></td>
<td></td>
<td></td>
<td>hospital</td>
</tr>
<tr>
<td>implant infection</td>
<td></td>
<td></td>
<td>infection</td>
<td></td>
</tr>
<tr>
<td>introductory psychology</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>narcotics during pregnancy side effects</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fight sinus infections</td>
<td></td>
<td></td>
<td>infections</td>
<td></td>
</tr>
<tr>
<td>health insurance high blood pressure</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>all antidepressant medications</td>
<td></td>
<td></td>
<td>antidepressant</td>
<td></td>
</tr>
</tbody>
</table>
Another approach is to model text *one character at a time*!
Realistic Wiki Hidden Unit

Above: Green for excited, blue for not excited
Below: top-5 guesses for the next character
Realistic Wiki Hidden Unit

Above: Green for excited, blue for not excited
Below: top-5 guesses for the next character
Vanilla RNN Flow Graph

\[ a_t = b + Wh_{t-1} + Ux_t \]
\[ h_t = \tanh(a_t) \]
\[ y_t = c + Vh_t \]

U – input to hidden
V – hidden to output
W – hidden to hidden
Training RNN

• “Backpropagation through time”
  = Backpropagation

• What to do with this if
  \[ w = w_1 = w_2 = w_3? \]

\[
\frac{\partial E}{\partial w} = \frac{\partial E}{\partial y_1} \frac{\partial y_1}{\partial w} + \frac{\partial E}{\partial y_2} \frac{\partial y_2}{\partial w} + \frac{\partial E}{\partial y_3} \frac{\partial y_3}{\partial w}
\]
Training RNN

• Again, assume
  \[ w = w_1 = w_2 \]

\[ \frac{\partial E}{\partial w} = \frac{\partial E}{\partial h_2} \frac{\partial h_2}{\partial w} + \frac{\partial E}{\partial h_3} \frac{\partial h_3}{\partial w} \]

\[ \frac{\partial E}{\partial h_2} = \frac{\partial E}{\partial y_2} \frac{\partial y_2}{\partial h_2} + \frac{\partial E}{\partial h_3} \frac{\partial h_3}{\partial h_2} \]
k timesteps?

\[
\frac{\partial E}{\partial h_2} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial h_k} \frac{\partial h_k}{\partial h_{k-1}} \ldots \frac{\partial h_3}{\partial h_2} + \frac{\partial E}{\partial y_{k-1}} \frac{\partial y_{k-1}}{\partial h_{k-1}} \ldots \frac{\partial h_3}{\partial h_2} + \ldots
\]

• What’s the problem?

\[
\frac{\partial h_k}{\partial h_{k-1}} = \frac{\partial h_{k-1}}{\partial h_{k-2}} = \ldots = \tanh(a_t) W
\]

• There will be terms like \(W^k\)

\[
a_t = b + W h_{t-1} + U x_t \\
h_t = \tanh(a_t) \\
y_t = c + V h_t
\]
What’s wrong with $W^k$?

• Suppose $W$ is diagonalizable for simplicity

\[ W = UDU^T \]

\[ W^k = UD^k U^T \]

• What if,
  – $W$ has an eigenvalue of 4?
  – $W$ has an eigenvalue of 0.25?
  – Both?
Cannot train it with backprop

Vanishing gradients

- Vanishing long term gradient \((g^t)\) is very small if \(g < 1\)
- Strong short term gradient
Do we need long-term gradients?

• Long-term dependency is one main reason we want temporal models
  – Example:

  Rob Ford told the flabbergasted reporters assembled at the press conference that ________.

  German for “travel”

  Die Koffer waren gepackt, und er reiste, nachdem er seine Mutter und seine Schwestern geküsst und noch ein letztes Mal sein angebetetes Gretchen an sich gedrückt hatte, das, in schlichten weißen Musselin gekleidet und mit einer einzelnen Nachthyazinthe im üppigen braunen Haar, kraftlos die Treppe herabgetaumelt war, immer noch blass von dem Entsetzen und der Aufregung des vorangegangenen Abends, aber voller Sehnsucht, ihren armen schmerzenden Kopf noch einmal an die Brust des Mannes zu legen, den sie mehr als ihr eigenes Leben liebte, ab.“

  Only now we understand the travel started, not ended (an)
LSTM: Long short-term Memory

• Need memory!
  – Vanilla RNN has volatile memory (automatically transformed every time-step)
  – More “fixed” memory stores info longer so errors don’t need to be propagated very far

• Complex architecture with memory
LSTM Starting point

• Instead of using volatile state transition
  \[ h_t = \tanh(W h_{t-1} + U x_t + b) \]

• Use fixed transition and learn the difference
  \[ c_t = c_{t-1} + \tanh(W y_{t-1} + U x_t + b) \]

  – Now we can truncate the BPTT safely after several timesteps

• However, this has the drawback of \( c_t \) being stored for too long
  – Add a weight? (subject to vanishing as well)
  – Add an “adaptive weight”
Forget Gate

• Decide how much of \( c_{t-1} \) should we forget

\[
c_t = f_t \odot c_{t-1} + z_t \\
z_t = \tanh(W y_{t-1} + U x_t + b)
\]

• Forget neurons also trained

\[
f_t = \sigma(W_f x_t + R_f y_{t-1} + p_f \odot c_{t-1} + b_f)
\]

• How much we forget is dependent on:
  – Previous output
  – Current input
  – Previous memory
Input Modulation

- Memory is supposed to be “persistent”
- Some input might be corrupt and should not affect our memory
- We may want to decide which input affects our memory
- Input Gate:
  \[ i^t = \sigma(W_i x^t + R_i y^{t-1} + p_i \odot c^{t-1} + b_i) \]  
  \textit{input gate}
- Final memory update:
  \[ c^t = i^t \odot z^t + f^t \odot c^{t-1} \]
Output Modulation

• Do not always “tell” what we remembered

\[
o^t = \sigma(W_o x^t + R_o y^{t-1} + p_o \odot c^t + b_o) \quad \text{output gate}
\]
\[
y^t = o^t \odot h(c^t) \quad \text{block output}
\]

• Only output if we “feel like it”

• The output part can vary a lot depending on applications
LSTM

- Hochreiter & Schmidhuber (1997)
- Use gates to remember things for a long period of time
- Use gates to modulate input and output
LSTM Architecture

- “Official version” with a lot of peepholes

\[
\begin{align*}
    z^t &= g(W_z x^t + R_z y^{t-1} + b_z) \\
    i^t &= \sigma(W_i x^t + R_i y^{t-1} + p_i \odot c^{t-1} + b_i) \\
    f^t &= \sigma(W_f x^t + R_f y^{t-1} + p_f \odot c^{t-1} + b_f) \\
    c^t &= i^t \odot z^t + f^t \odot c^{t-1} \\
    o^t &= \sigma(W_o x^t + R_o y^{t-1} + p_o \odot c^t + b_o) \\
    y^t &= o^t \odot h(c^t)
\end{align*}
\]

Cf. LSTM: a search space odyssey
Google Speech recognition

- **Task:**
  - Google Now/Voice search / mobile dictation
  - Streaming, real-time recognition in 50 languages

- **Model:**
  - Deep Projection Long-Short Term Memory Recurrent Neural networks
  - Distributed training with asynchronous gradient descent across hundreds of machines.
  - Cross-entropy objective (truncated backpropagation through time) followed by sequence discriminative training (sMBR).
  - 40-dimensional filterbank energy inputs
  - Predict 14,000 acoustic state posteriors

Slide provided by Andrew Senior, Vincent Vanhoucke, Hasim Sak (June 2014)
LSTM Large vocabulary speech recognition

<table>
<thead>
<tr>
<th>Models</th>
<th>Parameters</th>
<th>Cross-Entropy</th>
<th>sMBR sequence training</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReLU DNN</td>
<td>85M</td>
<td>11.3</td>
<td>10.4</td>
</tr>
<tr>
<td>Deep Projection LSTM RNN</td>
<td>13M</td>
<td>10.7</td>
<td>9.7</td>
</tr>
<tr>
<td>(2 layer)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling** H. Sak, A. Senior, F. Beaufays to appear in Interspeech 2014
- **Sequence Discriminative Distributed Training of Long Short-Term Memory Recurrent Neural Networks** H. Sak, O. Vinyals, G. Heigold A. Senior, E. McDermott, R. Monga, M. Mao to appear in Interspeech 2014

Voice search task; Training data: 3M utterances (1900 hrs); models trained on CPU clusters

Slide provided by Andrew Senior, Vincent Vanhoucke, Hasim Sak (June 2014)
Bidirectional LSTM

Both forward and backward paths
Still DAG!

Fig. 2. Bidirectional Recurrent Neural Network
Pen trajectories

• Task: generate pen trajectories by predicting one \((x,y)\) point at a time

• Data: IAM online handwriting, 10K training sequences, many writers, unconstrained style, captured from a whiteboard

So you say to your neighbours,
would find the bus safe and sound
would be the vineyards

• First problem: what to use for the density model?
Recurrent Mixture Density Networks

- Network outputs parameterise a mixture distribution (usually Gaussian)
- Every prediction conditioned on all inputs so far

\[ \text{Pr}(x_{t+1}|x_{1:t}) = \sum_{k} w_k(x_{1:t}) \mathcal{N}(x_{t+1}|\mu_k(x_{1:t}), \Sigma_k(x_{1:t})) \]

- Number of components is number of choices for what comes next

Network details

\[ x_t \in \mathbb{R} \times \mathbb{R} \times \{0, 1\} \]
\[ y_t = \left( c_t, \{\pi_t^j, \mu_t^j, \sigma_t^j, \rho_t^j\}_{j=1}^{M} \right) \]
\[ \hat{y}_t = \left( \hat{c}_t, \{\hat{\mu}_t^j, \hat{\sigma}_t^j, \hat{\rho}_t^j\}_{j=1}^{M} \right) = b_y + \sum_{i}^{N} W_{hny} h_{t}^{n} \]

\[ c_t = \frac{1}{1 + \exp(\hat{c}_t)} \]
\[ \pi_t^j = \frac{\exp(\hat{\pi}_t^j)}{\sum_{j'=1}^{M} \exp(\hat{\pi}_t^{j'})} \]
\[ \mu_t^j = \hat{\mu}_t^j \]
\[ \sigma_t^j = \exp(\hat{\sigma}_t^j) \]
\[ \rho_t^j = \tanh(\hat{\rho}_t^j) \]

\[ \Rightarrow c_t \in (0, 1) \]
\[ \Rightarrow \pi_t^j \in (0, 1), \quad \sum_j \pi_t^j = 1 \]
\[ \Rightarrow \mu_t^j \in \mathbb{R} \]
\[ \Rightarrow \sigma_t^j > 0 \]
\[ \Rightarrow \rho_t^j \in (-1, 1) \]

Illustration of mixture density
Synthesis

- Adding text input
Learning text windows

\[ \phi(t, u) = \sum_{k=1}^{K} \alpha_i^k \exp \left( -\beta_i^k (\kappa_i^k - u)^2 \right) \]

\[ w_t = \sum_{u=1}^{U} \phi(t, u) c_u \]
A demonstration of online handwriting recognition by an RNN with Long Short Term Memory (from Alex Graves)

http://www.cs.toronto.edu/~graves/handwriting.html
LSTM Architecture Explorations

• “Official version” with a lot of peepholes

\[
\begin{align*}
    z^t &= g(W_z x^t + R_z y^{t-1} + b_z) \\
    i^t &= \sigma(W_i x^t + R_i y^{t-1} + p_i \odot c^{t-1} + b_i) \\
    f^t &= \sigma(W_f x^t + R_f y^{t-1} + p_f \odot c^{t-1} + b_f) \\
    c^t &= i^t \odot z^t + f^t \odot c^{t-1} \\
    o^t &= \sigma(W_o x^t + R_o y^{t-1} + p_o \odot c^t + b_o) \\
    y^t &= o^t \odot h(c^t)
\end{align*}
\]

Cf. LSTM: a search space odyssey
A search space odyssey

• What if we remove some parts of this?

1. No Input Gate (NIG)
2. No Forget Gate (NFG)
3. No Output Gate (NOG)
4. No Input Activation Function (NIAF)
5. No Output Activation Function (NOAF)
6. No Peepholes (NP)
7. Coupled Input and Forget Gate (CIFG)
8. Full Gate Recurrence (FGR)

Cf. LSTM: a search space odyssey
Datasets

- **TIMIT**
  - Speech data
  - Framewise classification
  - 3696 sequences, 304 frames per sequence
- **IAM**
  - Handwriting stroke data
  - Map handwriting strokes to characters
  - 5535 sequences, 334 frames per sequence
- **JSB**
  - Music Modeling
  - Predict next note
  - 229 sequences, 61 frames per sequence
Results

1. No Input Gate (NIG)
2. No Forget Gate (NFG)
3. No Output Gate (NOG)
4. No Input Activation Function (NIAF)
5. No Output Activation Function (NOAF)
6. No Peepholes (NP)
7. Coupled Input and Forget Gate (CIFG)
8. Full Gate Recurrence (FGR)

Cf. LSTM: a search space odyssey
Impact of Parameters

• Analysis method: fANOVA (Hutter et al. 2011, 2014)

• (Random) Decision forests trained on the parameter space to partition the parameter space and find the best parameter

• Given trained (random) decision forest, can go to each leave node and count the impact of missing one predictor
Impact of Parameters

Figure 3. Pie charts showing which fraction of variance of the test set performance can be attributed to each of the hyperparameters. The percentage of variance that is due to interactions between multiple parameters is indicated as “higher order.”

Cf. LSTM: a search space odyssey
Impact of Parameters

Figure 5. Left: The predicted marginal error for combinations of learning rate and hidden size. Right: The component that is solely due to the interaction of the two and cannot be attributed to changes in one of them alone. In other words the difference to the case of them being perfectly independent. (Blue is better than red.)

learning rate $\times$ hidden size = 6.7%
learning rate $\times$ input noise = 4.4%
hidden size $\times$ input noise = 2.0%
learning rate $\times$ momentum = 1.5%
momentum $\times$ hidden size = 0.6%
momentum $\times$ input noise = 0.4%

Cf. LSTM: a search space odyssey
GRU: Gated Recurrence Unit

• Much simpler than LSTM
  – No output gate
  – Coupled input and forget gate

Cf. slideshare.net
Data

• Music Datasets:
  – Nottingham, 1200 sequences
  – MuseData, 881 sequences
  – JSB, 382 sequences

• Ubisoft Data A
  – Speech, 7230 sequences, length 500

• Ubisoft Data B
  – Speech, 800 sequences, length 8000
Results

Nottingham
Music, 1200 sequences

MuseData
Music, 881 sequences

Cf. Empirical Evaluation of Gated Recurrent Neural Network Modeling
Results

Ubisoft Data A
Speech, 7230 sequences, length 500

Ubisoft Data B
Speech, 800 sequences, length 8000
CNN+RNN Example

Recurrent Neural Network

Convolutional Neural Network
“straw hat”

training example
“straw hat”

training example
“straw hat”

training example
“straw hat”

training example
before: \( h_0 = \max(0, W_{xh} \times x_0) \)

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

sample! <END> token => finish.
sample! <END> token => finish.

- Don't have to do greedy word-by-word sampling, can also search over longer phrases with beam search
RNN vs. LSTM

“hidden” representation (e.g. 200 numbers)
\[ h_1 = \max(0, W_{xh} \cdot x_1 + W_{hh} \cdot h_0) \]
RNN vs. LSTM

“hidden” representation (e.g. 200 numbers)
\[ h_1 = \max(0, W_{xh} \times x_1 + W_{hh} \times h_0) \]

LSTM changes the form of the equation for \( h_1 \) such that:
1. more expressive multiplicative interactions
2. gradients flow nicer
3. network can explicitly decide to reset the hidden state
I STM

t=1
out
hid
inp

t=2
out
hid
inp

t=3
out
hid
inp

t=4
out
hid
inp

t=5
out
hid
inp

t=6
out
hid
inp
Image Sentence Datasets

Microsoft COCO
[Tsung-Yi Lin et al. 2014]
mscoco.org

currently:
~120K images
~5 sentences each
Transfer Learning

Training example

“straw hat”
Pre-training

use weights pretrained from ImageNet

“straw hat”

training example
use weights pretrained from ImageNet

“straw hat”

training example

use word vectors pretrained with word2vec [1]

[1] Mikolov et al., 2013
Summary of the approach

We wanted to describe images with sentences.

1. Define a single function from input -> output
2. Initialize parts of net from elsewhere if possible
3. Get some data
4. Train with SGD
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
Well, I can kind of see it

- A baby laying on a bed with a stuffed bear
  logprob: -8.66

- A table with a plate of food and a cup of coffee
  logprob: -9.93

- A young boy is playing frisbee in the park
  logprob: -9.52
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
What this approach Doesn’t do:

- There is no reasoning

- A single glance is taken at the image, no objects are detected, etc.

- We can’t just describe any image