Single Images

• Convolutional neural networks

Image → CNN → High-level information
• Depth
• Object classes
• Object poses
• Etc.
Sequential Data

- Data depends on time
  - Video
  - Sentence

UT Dallas is a rising public research university in the heart of DFW.
Sequential Data Labeling

• Video frame labeling

Sequential Data Labeling

• Part-of-speech tagging (grammatical tagging)
Sequential Data Labeling

Output label: $Y_{t-1}$, $Y_t$, $Y_{t+1}$

Classifier

Input: $X_{t-1}$, $X_t$, $X_{t+1}$

How to capture information across time?
Recurrent Neural Networks

Output label

\[ Y_{t-1} \]

\[ Y_t \]

\[ Y_{t+1} \]

Input

\[ X_{t-1} \]

\[ X_t \]

\[ X_{t+1} \]

Internal state (memory)

\[ h \]
Hidden State Update

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

- **Input at time** \( t \)
- **Output at time** \( t \)
- **Hidden state at time** \( t \)
- **Hidden state at time** \( t-1 \)
Using the Hidden State

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

\[
y_t = f_{W'}(h_t)
\]
Recurrent Neural Networks

Output label: $Y_{t-1} \downarrow \uparrow Y_t \downarrow \uparrow Y_{t+1}$

Input: $X_{t-1} \downarrow \uparrow X_t \downarrow \uparrow X_{t+1}$

Internal state (memory): $h_{t-1} \downarrow \uparrow h_t \downarrow \uparrow h_{t+1}$
Vanilla RNN

Hidden state updating rule

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

\[ y_t = W_{hy} h_t \]
The same set of weights for different time steps $f_W, f_W'$.
RNN Training
Backpropagation through Time

What is the problem in this training paradigm?
Truncated Backpropagation through Time

Run forward and backward through chunks of the sequence instead of whole sequence
Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps.
Truncated Backpropagation through Time
Vanilla RNN Gradient Flow

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

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

\[ = \tanh \left( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \]
Vanilla RNN Gradient Flow

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

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

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

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

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

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

- Vanishing gradients \( \| \frac{\partial h_t}{\partial h_{t-1}} \|_2 < 1 \)
- Exploding gradients \( \| \frac{\partial h_t}{\partial h_{t-1}} \|_2 > 1 \)

https://en.wikipedia.org/wiki/Matrix_norm
Vanilla RNN Gradient Flow

• Exploding gradients \[ \| \frac{\partial h_t}{\partial h_{t-1}} \|_2 > 1 \]

• Gradient clipping

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

• Vanishing gradients \[ \| \frac{\partial h_t}{\partial h_{t-1}} \|_2 < 1 \]

• Change RNN architecture
Long Short Term Memory (LSTM)

Vanilla RNN

$$h_t = \tanh \left( W \left( \begin{array}{c} h_{t-1} \\ x_t \end{array} \right) \right)$$

Sigmoid

$$\sigma(x) = 1/(1 + e^{-x})$$

LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \left( \begin{array}{c} h_{t-1} \\ x_t \end{array} \right)$$

Input gate
forget gate
output gate
gate gate

Cell

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

Hidden state

$$h_t = o \odot \tanh(c_t)$$

Store Cell and hidden states
Long Short Term Memory (LSTM)

- $g$: Gate gate, how much to write to cell
- $i$: Input gate, whether to write to cell
- $f$: Forget gate, whether to erase cell
- $o$: Output gate, how much to reveal cell

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

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

- Make the RNN easier to preserve information over many steps
  - E.g., f = 1 and i = 0
  - This is difficult for vanilla RNN

- LSTM does not guarantee that there is no vanishing or exploding gradient

- It provides an easier way to learn long-distance dependencies
Gated Recurrent Unit (GRU)

\[ z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \]
\[ r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \]
\[ \hat{h}_t = \phi_h(W_h x_t + U_h (r_t \odot h_{t-1}) + b_h) \]
\[ h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \]

- \( x_t \): input vector
- \( h_t \): output vector
- \( \hat{h}_t \): candidate activation vector
- \( z_t \): update gate vector
- \( r_t \): reset gate vector
- \( W, U \) and \( b \): parameter matrices and vector

https://en.wikipedia.org/wiki/Gated_recurrent_unit
GRUs vs. LSTMs

• Both have a forget gate

• GRU has fewer parameters, no output gate

• GRUs have similar performance compared to LSTMs, have shown better performance on certain datasets
Recurrent Neural Networks

E.g., action recognition on video frames

E.g., image captioning, image -> sequences of words

E.g., action prediction, sequences of frames -> action class

E.g., Video Captioning
Sequence of video frames -> caption
Recurrent Units on CNN Features

RGB Image
Time t
Depth Image

RGB Image
Time t+1
Depth Image

data association

Convolution + ReLU
Max Pooling
Concatenation
Deconvolution
Addition
Recurrent Units

Labels

DA-RNN. Xiang & Fox, RSS’17
Summary

• RNNs can be used for sequential data to capture dependencies in time

• LSTMs and GRUs are better than vanilla RNNs

• It is difficult to capture long-term dependencies in RNNs

• Use transformers (next lecture)
Further Reading

• Stanford CS231n, lecture 10, Recurrent Neural Networks 
  http://cs231n.stanford.edu/

• Long Short Term Memory
  https://www.researchgate.net/publication/13853244_Long_Short-term_Memory