Recurrent Neural Networks*

- Basic concept of RNNs
- LSTM networks
- GRU networks
- Example application in tomography

*Slides “barrowed” from Prof. Greg Buzzard
Basic Concept of RNN

- State machine viewpoint
  - $x_t, y_t$ - input and output
  - $s_t$ - state
  - $\theta$ – parameter

- “Unrolling the loop”
  - Parameters $\theta$ are shared
  - Time dependent processes such as speech
RNN Problems and Solutions

- **Problem:**
  - Back propagation now iterates in time
  - Gradient tends to vanish over long time scales
  - Difficult to model long time dependencies in data

- **Solution:**
  - Use skip connection, batch normalization (BM) and tricky methods for gating information from the past.
  - Results in long-short time memory (LSTM) RNN
Long-Short Term Memory (LSTM)

- **LSTM architecture**
  - State has two components $s_t = [C_t, h_t]$
  - $C_t$ - cell state
    - Store information that flows from one time to the next
    - Reduces vanishing gradient problem much like the skipped connection
  - $h_t$ - hidden state
    - This is usually the output of the LSTM
    - It typically needs to be further processed to produce the desired output

Useful reference: [http://colah.github.io/posts/2015-08-Understanding-LSTMs/](http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
For example, if we have

\[ y = h \circ x \]

For rank 1 case:

\[ y_i = h_i \, x_i \]

For rank 2 case:

\[ y_{i,j} = h_{i,j} \, x_{i,j} \]

Etc., ...
A Look Inside LSTM block: L1

Forget old information that is not useful. “Forget gate”

Add new information that is of value

\[ C_t = f_t \circ C_{t-1} + a_t \]
A Look Inside an LSTM block: L2

\[ f_t = \sigma(W_{fx} \cdot x_t + W_{fh} \cdot h_{t-1} + b_f) \]
\[ i_t = \sigma(W_{ix} \cdot x_t + W_{ih} \cdot h_{t-1} + b_i) \]
\[ \tilde{C}_t = \tanh(W_{cx} \cdot x_t + W_{ch} \cdot h_{t-1} + b_c) \]
\[ C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \]

Hadamard product: \( \odot \)
Matrix multiplication: \( \cdot \)
Sigmoid: \( \sigma(z) = \frac{1}{1+e^{-z}} \)
A Look Inside an LSTM block: L3

\[
f_t = \sigma(W_{fx} \cdot x_t + W_{fh} \cdot h_{t-1} + b_f)
\]
\[
i_t = \sigma(W_{ix} \cdot x_t + W_{ih} \cdot h_{t-1} + b_i)
\]
\[
\tilde{C}_t = \tanh(W_{cx} \cdot x_t + W_{ch} \cdot h_{t-1} + b_c)
\]
\[
C_t = f_t \circ C_{t-1} + i_t \circ \tilde{C}_t
\]
\[
o_t = \sigma(W_{ox} \cdot x_t + W_{oh} \cdot h_{t-1} + b_o)
\]
\[
h_t = o_t \circ \tanh(C_t)
\]

Hadamard product: \( \circ \)
Matrix multiplication: \( \cdot \)
Sigmoid: \( \sigma(z) = \frac{1}{1+e^{-z}} \)
LSTM Intuition

- **The backbone**
  - Carries state forward and gradients backward.
  - Gating (Hadamard product) modulate information flow.

- **“Forgetting” gate:**
  - Use \( C, h, \) and \( x \) to determine how much to suppress.
  - If \( C \) encodes that we need a verb. Forget that when verb is found.

- **Input gate:**
  - \( i \) determines which values of \( C \) to update.
  - \( \tanh(\cdot) \) generates new state to add to \( C \).

- **Output gate:**
  - \( o \) is the output gate: modulates what part of the state \( C \) gets passed (via \( \tanh \)) to current output \( h \).
  - Could encode whether a noun is singular or plural to prepare for a verb.
Convolutional LSTM block*†

\[
\begin{align*}
&f_t = \sigma(W_{fx} * x_t + W_{fh} * h_{t-1} + W_{fc} \circ C_{t-1} + b_f) \\
&i_t = \sigma(W_{ix} * x_t + W_{ih} * h_{t-1} + W_{ic} \circ C_{t-1} + b_i) \\
&\tilde{C}_t = \tanh(W_{cx} * x_t + W_{ch} * h_{t-1} + b_c) \\
&C_t = f_t \circ C_{t-1} + i_t \circ \tilde{C}_t \\
o_t = \sigma(W_{ox} * x_t + W_{oh} * h_{t-1} + W_{oc} \circ C_{t-1} + b_o) \\
h_t = o_t \circ \tanh(C_t)
\end{align*}
\]

ConvLSTM2D available in TensorFlow

Why use ConvLSTM to SBP

- **Advantages:**
  - Allows LSTM RNN to be used with images
  - Dramatically reduces number of parameters as compared to LSTM implementation.

- **Disadvantage:**
  - Training a deep LSTM requires that all the states be stored during training.
  - This requires a large amount of GPU memory.
GRU: Gated Recurrent Unit*  

**Hadamard product:** $\circ$  
**Matrix multiplication:** $\cdot$  
**Sigmoid:** $\sigma(z) = \frac{1}{1+e^{-z_i}}$  

$$r_t = \sigma(W_{rx} \cdot x_t + W_{rh} \cdot h_{t-1})$$  
$$z_t = \sigma(W_{zx} \cdot x_t + W_{zh} \cdot h_{t-1})$$  
$$\tilde{h}_t = \tanh(W_{hx} \cdot x_t + W_{hh} \cdot (r_t \circ h_{t-1}))$$  
$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h}_t$$  

Why use a GRU?

- **Advantages of GRU:**
  - Simpler and typically just as effective
  - Combine $C$ and $h$ into a single state/output.
  - Combine forget and input gates into update gate, $z$.

- **Disadvantages of GRU**
  - Convolutional GRU exists* but is not already implemented in PyTorch or TensorFlow.

---

*link to paper* Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville, “DELVING DEEPER INTO CONVOLUTIONAL NETWORKS FOR LEARNING VIDEO REPRESENTATIONS,” ICLR, 2016.
Computed Tomography (CT)

- Reconstruct object from projections
  - Used in medical, scientific, and industrial imaging
  - Collect views of object from different angles
  - Reconstruct 3D object
Sparse View CT Reconstruction

- Sparse view: Reduce acquisition time and dosage
- Filtered Back Projection (FBP) reconstruction
  - FBP requires 256 views for a $256 \times 256$ reconstruction.
  - 16-view (i.e., sparse view) reconstruction looks very bad

Solution: Deep Neural Net reconstruction
Stacked Back Projections (SBP)*

- Measure projections
- Back project each projection
- Stack them up in order

SBP contains
- All information from the sinogram.
- Sequential information.

**RSBP Network Architecture**

- **Recurrent Stacked Back Projection (RSBP)**
  - Uses convolutional LSTM\(^1\) processing of SBP

---

Reconstruction on Simulated data

Ground Truth

8 Views

MBIR NRMSE: 0.5517

FBP-CNN NRMSE: 0.4968

SBP-CNN NRMSE: 0.4460

MBIR NRMSE: 0.4782

FBP-CNN NRMSE: 0.4022

SBP-CNN NRMSE: 0.3646

RSBP-CNN NRMSE: 0.3529

Proposed Method

Typical 16-View FBP Reconstruction
Reconstruction on Real data

8 Views

MBIR
NRMSE: 0.5416

FBP-CNN
NRMSE: 0.5057

SBP-CNN
NRMSE: 0.4840

RSBP-CNN
NRMSE: 0.4652

16 Views

MBIR
NRMSE: 0.4266

FBP-CNN
NRMSE: 0.3620

SBP-CNN
NRMSE: 0.3569

RSBP-CNN
NRMSE: 0.3195

Proposed Method

Typical 16-View FBP Reconstruction
Unsupervised Learning

- The Concept of Unsupervised Training
- Autoencoders
- Decoders as Generators
Autoencoder

- Two stages:
  - Encoder generates a latent vector, \( z \), then encodes the object
  - Decoder generates an approximation to the original input
Training an Autoencoders

- An example of unsupervised learning

\[ \text{Loss}(\theta) = \sum_{k=0}^{K-1} \| y_k - g_{\theta_2}(f_{\theta_1}(y_k)) \|^2 \]

- Encoder Network \( f_{\theta_1} \)
- Latent vector, \( z \)
- Decoder Network \( g_{\theta_2} \)
- MSE Loss
- \( L(\theta) \)
## Probability Distribution of Natural Images

- An image can be thought of as a point in an $N$-dimensional space.
  - $x \in \mathbb{R}^N$ is a point in a space
  - Natural images have some distribution, $X \sim p(x)$

- Let $W$ be a white noise image, then
  - $W_i \sim \text{Uniform}[0,1]$
  - $W$ fills the space
  - A typical sample of $W$ has zero chance of looking like a natural image
  - Conclusion: $p(x)$ is very sparse

- Thought experiment:
  - Number of images ever seen by people:
    - $N \leq (10^{14} \text{ people}) \times \left(10^{14} \frac{\text{image}}{\text{people}}\right) = 10^{28}$
  - Bits required to encode all images:
    - $\log_2 \{N\} \leq \log_2 \{10^{28}\} = 28 \log_2 \{10\} = 93$ bits
  - All images can be encoded in less than 128 bits = 16 bytes
Application of Encoder

- The encoder as a preprocessor:
  - Dimensionality reduction
  - Pretrained

\[
y \xrightarrow{f_{\theta_1}} \text{Encoder Network} \xrightarrow{z} \text{latent vector, } z \xrightarrow{\text{DNN Postprocessing}} x
\]
The decoder can be used as a generator:

- Produces instantiations of \( y \)
- Variational autoencoders