CS 6501: Deep Learning for Computer Graphics

Convolutional and Recurrent Neural Networks

Connelly Barnes
Outline

- Convolutional Neural Networks ("CNNs", "ConvNets")
  - Useful for images
- Recurrent Neural Networks ("RNNs")
  - Useful for processing sequential data (e.g. text)
Outline

• Convolutional Neural Networks
  • History
  • Convolutional layers
  • Downsampling: stride and pooling layers
  • Fully connected layers
  • Residual networks
  • Data augmentation
• Recurrent Neural Networks
• Deep learning libraries
History

A bit of history:

Hubel & Wiesel, 1959
RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX

1962
RECEPTIVE FIELDS, BINOCULAR INTERACTION AND FUNCTIONAL ARCHITECTURE IN THE CAT'S VISUAL CORTEX

1968...
History

Topographical mapping in the cortex: nearby cells in cortex represented nearby regions in the visual field
History

Hierarchical organization

Hubel & Weisel
- topographical mapping

featural hierarchy
- high level
- mid level
- low level
History

Neurocognitron [Fukushima 1980]

“sandwich” architecture (SCSCSC…)
simple cells: modifiable parameters
complex cells: perform pooling
History

Gradient-based learning applied to document recognition
[LeCun, Bottou, Bengio, Haffner 1998]
History

ImageNet Classification with Deep Convolutional Neural Networks
[Krizhevsky, Sutskever, Hinton, 2012]

“AlexNet”
Today: CNNs Widely Used

• Self-driving cars
Today: CNNs Widely Used

- Image Classification
Convolutional Neural Networks

- Similar to multilayer neural network, but weight matrices now have a special structure (Toeplitz or block Toeplitz) due to convolutions.
- The convolutions typically sum over all color channels.
**Convolutional Neural Network Neuron Layout**

- Input layer: RGB image
  - Centered, i.e. subtract mean over training set
  - Usually crop to fixed size (square) input image
Convolutional Neural Network Neuron Layout

- Hidden layer
Receptive Field

Receptive Field: Input Region

Weights (Shared)

Hidden Layer Neuron

Image from Wikipedia
Mathematically...

\[ L_i = \varphi (L_{i-1} \ast [W_1, W_2, \ldots W_n] + [b_1, \ldots, b_n]) \]
Convolutional / Filtering

Learn multiple filters.

E.g.: 1000x1000 image
100 Filters
Filter size: 10x10
10K parameters

Input

Feature Map

Feature maps

Pooling

Non-linearity

Convolution (Learned)

Input Image
Outline

• Convolutional Neural Networks
  • History
  • Convolutional layers
  • **Downsampling: stride and pooling layers**
  • Fully connected layers
  • Residual networks
  • Data augmentation
• Recurrent neural networks
• Deep learning libraries
Stride

- Stride $m$ indicates that instead of computing every pixel in the convolution, compute only every $m$th pixel.
Max/average pooling

- “Downsampling” using max() operator
- Downsampling factor $f$ could differ from neighborhood size $N$ that is pooled over.
Max/average pooling

• For max pooling, backpropagation just propagates error back to whichever neuron had the maximum value.
• For average pooling, backpropagation splits error equally among all the input neurons.
Fully connected layers

• Connect every neuron to every other neuron, as with multilayer perceptron.

[LeCun et al., 1998]
Outline

• Convolutional Neural Networks
  • History
  • Convolutional layers
  • Downsampling: stride and pooling layers
  • Fully connected layers
  • Residual networks
  • Data augmentation
• Recurrent Neural Networks
• Deep learning libraries
Residual networks

- Make it easy to learn the identity function:
  - Network with all zero weights gives identity function.
  - Helps with vanishing/exploding gradients.
Outline

• Convolutional Neural Networks
  • History
  • Convolutional layers
  • Downsampling: stride and pooling layers
  • Fully connected layers
  • Residual networks
  • Data augmentation
• Recurrent neural networks
• Deep learning libraries
Data Augmentation

• Many weights to train
  • Often would be helpful to have more training data
• Fake having more training data
  • Random rotations
  • Random flips
  • Random shifts
  • Recolorings
  • etc

Figure from BaiduVision
Outline

• Convolutional Neural Networks
• Recurrent Neural Networks
  • Simple RNNs
  • LSTM, GRU
  • Applications
• Deep learning libraries
Recurrent Neural Networks

• Feedforward neural networks have no memory: cannot remember the state of the world between one instant of time and the next
  • Cannot remember important events and recall them in the future
  • Cannot perform loops
  • Cannot implement arbitrary algorithms
• Recurrent networks help by adding memory to the computation
Recurrent Neural Networks

Outputs

Hidden State (Memory)

Inputs

Vanilla Neural Networks

Slide from Stanford CS231N
Recurrent Neural Networks

e.g. Image Captioning
image -> sequence of words

Slide from Stanford CS231N
Recurrent Neural Networks

- one to one
- one to many
- many to one
- many to many
- many to many

e.g. Sentiment Classification
sequence of words -> sentiment
Recurrent Neural Networks

- **One to one**
- **One to many**
- **Many to one**
- **Many to many**

E.g. **Machine Translation**

seq of words -> seq of words

Slide from Stanford CS231N
Recurrent Neural Networks

One to one

One to many

Many to one

Many to many

E.g. Video classification on frame level

Slide from Stanford CS231N
We can process a sequence of vectors $x$ by applying a recurrence formula at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

- $h_t$: new state
- $h_{t-1}$: old state
- $x_t$: input vector at some time step
- $f_W$: some function with parameters $W$
Simple Recurrent Neural Network

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

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

Output:
\[ y_t = W_{hy} h_t \]
How to Train?

- **Backpropagation through time**
How to Train?

- More efficient: **Truncated backpropagation through time**

1. Only run backpropagation every $k_1$ time steps

2. Limit number of times ($k_2$) unfolded

Image from Wikipedia
Application of Recurrent Neural Network

- **Text synthesis** (student presentation)

PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.
Vanishing Gradients Revisited

• Suppose we want to “remember” an event at time 0 and use this in our model of the world at some later time $t$.
• Error gradients vanish exponentially quickly in the size of the time lag between these events.
• Fancier RNN models help with this problem:
  • Long Short Term Memory (LSTM)
  • Gated Recurrent Units (GRU)
Long Short Term Memory

• How to better remember hidden state for a long time?
• Idea: use *gates* to create cells that can remember for a long time.

Transistor diagram (from Wikipedia)

• Rough analogy: ternary logic gates used in transistors, AND, OR, …
• But use sigmoid activations so we have continuous values in [0, 1]
Simple Recurrent Neural Network

Diagrams from Christopher Olah
Long Short Term Memory

Gates: forget, input, output

Diagrams from Christopher Olah
Long Short Term Memory

- Easy to have hidden state $C_t$ just flow through time, unchanged.
Long Short Term Memory

- Gate: pointwise multiplication.
- Multiply by zero: let nothing through.
- Multiply by one: let everything through.

Diagrams from Christopher Olah
Long Short Term Memory

- Forget gate: conditionally discard previously remembered information.

\[ f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f) \]
Long Short Term Memory

• Input gate: conditionally remember new information.

\[
i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i) \]
\[
\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)
\]
Long Short Term Memory

• Output gate: conditionally output a relevant part of our memory.

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

\[ h_t = o_t \times \tanh \left( C_t \right) \]
Gated Recurrent Units (GRUs)

- Merge input / forget units into a single “update unit.”
- Merge hidden states.

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

Diagrams from Christopher Olah
Outline

• Convolutional Neural Networks
• Recurrent Neural Networks
• Deep learning libraries
Libraries

• Deep learning:
  • **Caffe** (C++ with Python bindings),
  • **Torch** (Lua)
  • **TensorFlow** (C++ with Python bindings)
  • Python: **Keras**, built on **Theano**

• Recurrent networks (search for your framework + LSTM):
  • **Caffe**
  • **Torch**
  • **TensorFlow**
  • **Keras**