CSC321 Lecture 17: ResNets and Attention

Roger Grosse
Overview

Two topics for today:

- **Topic 1: Deep Residual Networks (ResNets)**
  - This is the state-of-the-art approach to object recognition.
  - It applies the insights of avoiding exploding/vanishing gradients to train really deep conv nets.

- **Topic 2: Attention**
  - Machine translation: it’s hard to summarize long sentences in a single vector, so let’s let the decoder peek at the input.
  - Vision: have a network glance at one part of an image at a time, so that we can understand what information it’s using.
  - We can use attention to build differentiable computers (e.g. Neural Turing Machines)
Deep Residual Networks

I promised you I’d explain the best ImageNet object recognizer from 2015, but that it required another idea.

<table>
<thead>
<tr>
<th>Year</th>
<th>Model</th>
<th>Top-5 error</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>Hand-designed descriptors + SVM</td>
<td>28.2%</td>
</tr>
<tr>
<td>2011</td>
<td>Compressed Fisher Vectors + SVM</td>
<td>25.8%</td>
</tr>
<tr>
<td>2012</td>
<td>AlexNet</td>
<td>16.4%</td>
</tr>
<tr>
<td>2013</td>
<td>a variant of AlexNet</td>
<td>11.7%</td>
</tr>
<tr>
<td>2014</td>
<td>GoogLeNet</td>
<td>6.6%</td>
</tr>
<tr>
<td>2015</td>
<td>deep residual nets</td>
<td>4.5%</td>
</tr>
</tbody>
</table>

That idea is exploding and vanishing gradients, and dealing with them by making it easy to pass information directly through a network.
Deep Residual Networks

- Recall: the Jacobian $\frac{\partial h(T)}{\partial h(1)}$ is the product of the individual Jacobians:

$$
\frac{\partial h(T)}{\partial h(1)} = \frac{\partial h(T)}{\partial h(T-1)} \cdots \frac{\partial h(2)}{\partial h(1)}
$$

- But this applies to multilayer perceptrons and conv nets as well! (Let $t$ index the layers rather than time.)

- Then how come we didn’t have to worry about exploding/vanishing gradients until we talked about RNNs?
  - MLPs and conv nets were at most 10s of layers deep.
  - RNNs would be run over hundreds of time steps.
  - This means if we want to train a really deep conv net, we need to worry about exploding/vanishing gradients!
Deep Residual Networks

- Remember Homework 3? You derived backprop for this architecture:

  \[ z = W^{(1)}x + b^{(1)} \]
  \[ h = \phi(z) \]
  \[ y = x + W^{(2)}h \]

- This is called a residual block, and it’s actually pretty useful.

- Each layer adds something (i.e. a residual) to the previous value, rather than producing an entirely new value.

- Note: the network for \( \mathcal{F} \) can have multiple layers, be convolutional, etc.
Deep Residual Networks

- We can string together a bunch of residual blocks.

- What happens if we set the parameters such that $F(x^{(\ell)}) = 0$ in every layer?
  - Then it passes $x^{(1)}$ straight through unmodified!
  - This means it’s easy for the network to represent the identity function.

- Backprop:

$$
\bar{x}^{(\ell)} = x^{(\ell+1)} + x^{(\ell+1)} \frac{\partial F}{\partial x} \\
= x^{(\ell+1)} \left( I + \frac{\partial F}{\partial x} \right)
$$

- As long as the Jacobian $\partial F / \partial x$ is small, the derivatives are stable.
Deep Residual Networks

- Deep Residual Networks (ResNets) consist of many layers of residual blocks.
- For vision tasks, the $\mathcal{F}$ functions are usually 2- or 3-layer conv nets.
- Performance on CIFAR-10, a small object recognition dataset:

For a regular convnet (left), performance declines with depth, but for a ResNet (right), it keeps improving.
Deep Residual Networks

- A 152-layer ResNet achieved 4.49% top-5 error on Image Net. An ensemble of them achieved 3.57%.
- Previous state-of-the-art: 6.6% (GoogLeNet)
- Humans: 5.1%
- They were able to train ResNets with more than 1000 layers, but classification performance leveled off by 150.
- What are all these layers doing? We don’t have a clear answer, but the idea that they’re computing increasingly abstract features is starting to sound fishy...
Next topic: attention-based models.

Remember the encoder/decoder architecture for machine translation:

The network reads a sentence and stores all the information in its hidden units.

Some sentences can be really long. Can we really store all the information in a vector of hidden units?

Let’s make things easier by letting the decoder refer to the input sentence.
Attention-Based Machine Translation

- We’ll look at the translation model from the classic paper: Bahdanau et al., *Neural machine translation by jointly learning to align and translate*. ICLR, 2015.

- Basic idea: each output word comes from one word, or a handful of words, from the input. Maybe we can learn to attend to only the relevant ones as we produce the output.
Attention-Based Machine Translation

- The model has both an encoder and a decoder. The encoder computes an **annotation** of each word in the input.
- It takes the form of a **bidirectional RNN**. This just means we have an RNN that runs forwards and an RNN that runs backwards, and we concatenate their hidden vectors.
  - The idea: information earlier or later in the sentence can help disambiguate a word, so we need both directions.
  - The RNN uses an LSTM-like architecture called gated recurrent units.
Attention-Based Machine Translation

- The decoder network is also an RNN. Like the encoder/decoder translation model, it makes predictions one word at a time, and its predictions are fed back in as inputs.
- The difference is that it also receives a context vector $c^{(t)}$ at each time step, which is computed by attending to the inputs.
Attention-Based Machine Translation

- The context vector is computed as a weighted average of the encoder's annotations.
  \[ c^{(i)} = \sum_j \alpha_{ij} h^{(j)} \]

- The attention weights are computed as a softmax, where the inputs depend on the annotation and the decoder's state:
  \[ \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})} \]
  \[ e_{ij} = a(s^{(i-1)}, h^{(j)}) \]

- Note that the attention function depends on the annotation vector, rather than the position in the sentence. This means it's a form of content-based addressing.
  - My language model tells me the next word should be an adjective. Find me an adjective in the input.
Attention-Based Machine Translation

- Here’s a visualization of the attention maps at each time step.

- Nothing forces the model to go linearly through the input sentence, but somehow it learns to do it.
  - It’s not perfectly linear — e.g., French adjectives can come after the nouns.
Attention-Based Machine Translation

- The attention-based translation model does much better than the encoder/decoder model on long sentences.
Attention-Based Caption Generation

- Attention can also be used to understand images.
- We humans can’t process a whole visual scene at once.
  - The fovea of the eye gives us high-acuity vision in only a tiny region of our field of view.
  - Instead, we must integrate information from a series of glimpses.
- The next few slides are based on this paper from the UofT machine learning group:

The caption generation task: take an image as input, and produce a sentence describing the image.

**Encoder:** a classification conv net (VGGNet, similar to AlexNet). This computes a bunch of feature maps over the image.

**Decoder:** an attention-based RNN, analogous to the decoder in the translation model

- In each time step, the decoder computes an attention map over the entire image, effectively deciding which regions to focus on.
- It receives a context vector, which is the weighted average of the conv net features.
Attention-Based Caption Generation

This lets us understand where the network is looking as it generates a sentence.

A bird flying over a body of water.

A woman is throwing a frisbee in a park.
A dog is standing on a hardwood floor.
A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.
A group of people sitting on a boat in the water.
A giraffe standing in a forest with trees in the background.
Attention-Based Caption Generation

This can also help us understand the network’s mistakes.

- A large white bird standing in a forest.
- A woman holding a clock in her hand.
- A man wearing a hat and a hat on a skateboard.
- A person is standing on a beach with a surfboard.
- A woman is sitting at a table with a large pizza.
- A man is talking on his cell phone while another man watches.
Neural Turing Machines (optional)

- We said earlier that multilayer perceptrons are like differentiable circuits.
- Using an attention model, we can build differentiable computers.
- We’ve seen hints that sparsity of memory accesses can be useful:

  Computers have a huge memory, but they only access a handful of locations at a time. Can we make neural nets more computer-like?
Neural Turing Machines (optional)

- Recall Turing machines:

  You have an infinite tape, and a head, which transitions between various states, and reads and writes to the tape.

  “If in state A and the current symbol is 0, write a 0, transition to state B, and move right.”

  These simple machines are universal — they’re capable of doing any computation that ordinary computers can.
Neural Turing Machines (optional)

- **Neural Turing Machines** are an analogue of Turing machines where all of the computations are differentiable.
  - This means we can train the parameters by doing backprop through the entire computation.
- Each memory location stores a vector.
- The read and write heads interact with a weighted average of memory locations, just as in the attention models.
- The controller is an RNN (in particular, an LSTM) which can issue commands to the read/write heads.
Neural Turing Machines (optional)

- Repeat copy task: receives a sequence of binary vectors, and has to output several repetitions of the sequence.
- Pattern of memory accesses for the read and write heads:
Neural Turing Machines (optional)

- Priority sort: receives a sequence of (key, value) pairs, and has to output the values in sorted order by key.

- Sequence of memory accesses:

  ![Diagram of memory accesses]

  **Write Weightings**  **Read Weightings**

  ![Time](#)  ![Time](#)