Lecture: Deep Networks Intro

• Topics: 1st lecture wrap-up, difficulty training deep networks, image classification problem, using convolutions, tricks to train deep networks

• Resources: http://www.cs.utah.edu/~rajeev/cs7960/notes/
• Canvas/registration
Early Bust

Commercial NN chips fell out of favor in the early 2000s

- SVMs overtook ANNs
- General-purpose processors were getting faster every year and quickly overtaking ASICs
- Limited market for machine learning

Note that none of the above is applicable today
First Lecture Recap

1. Hardware trends are pushing for accelerators.
2. Machine learning is taking over.
3. Déjà vu – but things will be different this time.
4. Machine learning primitives have stabilized.
5. Taxonomy and ANN vs. SNN discussion.
Google TPU

- Version 1: 15-month effort, basic design, only for inference, 92 TOPs peak, 15x faster than GPU, 40 W 28nm 300 mm$^2$ chip
- Version 2: designed for training, a pod is a collection of v2 chips connected with a torus topology
- Version 3: 8x higher throughput, liquid cooled

Ref: Google
Deep vs. Shallow

• Any combinational circuit can be expressed as a sum of products – but a two-layer circuit is impractical because of high fan-in and fan-out – hence, we design modular deep circuits

• Similarly, a deep neural network may be able to learn simple features in early layers and more complex features in subsequent layers
Learning in Deep Networks

- When training deep networks, the early layers may have a very slow learning rate (vanishing gradient problem)

\[ \frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4} \]
The Unstable Gradient Problem

- In a deep network, the learning rates of different layers tend to be wildly different (learning rates in early layers will either vanish or explode)

\[
\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}
\]
Deep Learning Challenges

- Vanishing/exploding gradients
- Picking the correct activation function
- Good initialization of weights
- Choice of network architecture, hyper-parameters, etc.
Deep Convolutional Neural Networks

Three Major Changes: local receptive fields, pooling, shared weights

Each neuron only sees a “local receptive field” (5x5 grid of neurons in this example).

All the neurons use the same set of weights for their local receptive field, i.e., they are all looking for the same pattern in different parts of the image.

What we’ve shown here is 1 “filter” or 1 “feature map”.
Convolutional Layer

A given layer can have many parallel filters
Intuitively, think of an early filter as detecting small localized patterns (edge detection). In practice, they look like this:
Pooling

Simplify/condense information: once a feature has been found, it’s exact location is not as important as its relative location – especially if it helps us reduce the parameters. Commonly used: max, L2.
A Shallow CNN for MNIST

How many weights/biases does this network need?

\[(25+1) \times 3 + ((3 \times 12 \times 12 + 1) \times 10 ) = 4408\]
When a layer has many input feature maps, the filter is 3-dimensional. The CNN ends with a few fully-connected classifier layers (MLP-like).
Krizhevsky et al., 2012

Deep network (5 conv+maxpool and 2 fully-connected) that uses 2 GPUs and enforces locality in some layers
Explaining the Rise of ML

1. Layers with few shared weights made training easier

2. Access to GPUs made training faster

3. Access to large labeled datasets made training tractable/accurate

4. Many tricks have been introduced in the past decade+
   - better activation functions (e.g., ReLU)
   - dropout (injects noise to avoid over-fitting)
   - better weight initialization, regularization (helps convergence)
   - expanded inputs (to grow training data and avoid over-fitting)
Deep Networks for Image Classification

- MNIST: 784-pixel images of hand-written digits; 50K training images; 10K testing images

- ILSVRC: e.g., 1000 categories, 1.2 million training images, 150K test images, top-5 criterion
Deep Learning Success

- Modern deep learning is better than humans on both MNIST (>99%) and ILSVRC top-5 (>95%)
Accuracies

- Baseline: single hidden layer with 100 neurons: 97.8%
- CNN: added 20 5x5 filters and a 2x2 max-pool: 98.78%
- CNN: added 40 20x5x5 filters and 2x2 max-pool: 99.06%
- CNN: substitute sigmoid with ReLU: 99.23%
- CNN: expand the training data: 99.37%
- CNN: adding two fully-connected layers: 99.43%
- CNN: 1000 neurons in fully-connected layers: 99.47%
- CNN: adding dropout: 99.60%
- CNN: voting among an ensemble of 5 nets: 99.67%

(ILSVRC winners have 152 layers)

Many ways to avoid over-fitting in fully-connected networks (conv layers don’t need these because the weights are shared): L2 regularization, dropout, expanded inputs.
Glossary

- **Sigmoid activation function**: \( f(x) = 1/(1+e^{-x}) \) – provides a smooth step function; tanh is similar.
- **ReLU activation function**: \( f(x) = \max(0,x) \) – it allows the output to grow larger than 1, and has a larger \( \sigma' \) of 0 or 1.
- **Softmax activation function**: \( f(z_j) = e^{z_j}/\sum_k e^{z_k} \) -- it’s normalizing the neuron outputs in a layer so they sum to 1 and the largest neuron sticks out.
- **Feature map and filter**: a filter or kernel is the grid of weights; a feature map is the resulting set of values when a filter is applied to a set of inputs.
- **Max pooling**: extracts the largest value in a 2D input grid.
- **L2 pooling**: computes the square root of the sum of squares of the values in a 2D input grid.
- **L2 regularization**: includes the weights in the cost function during training so we’re trying to not only reduce the error, but also the values of the weights.
- **Expanded inputs**: to avoid over-fitting, the (say) 50K training images are expanded to 250K images. Each image is shifted slightly to the left/right/top/bottom.
- **Dropout**: some activation functions are randomly dropped during training (to avoid overfitting).
- **DNN/CNN**: CNNs use shared kernels for all neurons in a feature map, while DNNs use private kernels for each neuron in a feature map.
- **SVM**: a mathematical approach to incrementally define hyperplanes that separate clusters – an alternative way to classify inputs into different categories.
LSTMs

- The neurons in MLPs and CNNs do not have state – every input image results in brand new computations with no memory of previous images.

- An LSTM is better suited for speech/text processing where interpreting a new syllable or pronoun may depend on past inputs.

- Recurrent neural networks (RNNs, where a layer’s output feeds back as input for the next computation) have evolved into LSTMs
LSTMs

1. Forget gate
2. New info to be retained
3. Strength of new info
4. Convert cell states to (-1, 1)
5. What part of the cell must be output

Image credit: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
References

• Chapters 5 and 6 of Nielsen’s book:
  http://neuralnetworksanddeeplearning.com