TRAINING DEEP ARCHITECTURES

Deeply-Supervised Nets

Chen-Yu Lee *
Dept. of EECS, UCSD
chenyu@ucsd.edu

Sanming Xu *
Dept. of EECS, UCSD
sanhx@ucsd.edu

Patrick Gallagher
Microsoft Research
gallagher@msn.com

Zhuangyou Zhang
Microsoft Research
zhang@min.com

Zhiyuan Yu
Dept. of EECS, UCSD
zhiyu@ucsd.edu

- Pre-dates resnet / densenet
- Basic idea, have a separate classifier path which goes from intermediate layers directly to final label.
- Initial classifiers won’t do well, but will provide gradients that make initial layers informative for final task.

GENERAL

Reminder

- PSET 5 due Friday 11:59pm.
- Slides PDFs Due Sunday 11:59pm.
- Presentation Schedule Posted Monday
  - Everyone is expected to attend both days
- Project Reports Due Thursday 11:59pm.
**TRAINING DEEP ARCHITECTURES**

**Do Deep Nets Really Need to be Deep?**

Lei Junmei Bu  
University of Toronto  
jiexi@cs.toronto.edu

Rich Caruana  
Microsoft Research  
caruana@microsoft.com

**Abstract**

Currently, deep neural networks are the state-of-the-art on problems such as speech recognition and computer vision. In this paper we empirically demonstrate that shallow feed-forward nets can learn the complex functions previously learned by convolutional nets or RNNs through a novel training technique. Moreover, in some cases the shallow nets can learn these deep functions using the same number of parameters as the original deep models. On the TIMIT phoneme recognition and CIFAR-10 image recognition tasks, shallow nets can be trained that perform similarly to complex, well-regularized, deep convolutional models.

- Shows that more "depth" helps not just to represent more complex function, but also in training.

**TRAINING DEEP ARCHITECTURES**

**Basic Idea**

- Train a very deep network.
- Now train a much shallower network to mimic the output of the deep network.
- On the same training set, don’t set your target to be cross-entropy wrt true label.
- Instead, train on squared-error to prediction of deep network.
- This is more supervision than the label. Teacher tells student:
  - What should be its confidence in its prediction
  - Which of the wrong answers are more likely than others (Geoff Hinton’s "dark knowledge")
  - Sometimes, to predict the wrong answer

**TRAINING DEEP ARCHITECTURES**

**Deep Network Models are Large**

- But can be compressed post-training
  - But you can’t train a smaller model as well from scratch
- Train deep network with lots of weights
- Set smallest weights to 0
- Now retrain remaining weights, fixing these to 0
- Do this repeatedly (removing a small number of weights at a time)
- Earlier works had shown this can even help in generalization (Yann LeCun called it "optimal brain damage")

**TRAINING DEEP ARCHITECTURES**

<table>
<thead>
<tr>
<th>Architecture</th>
<th># Parms.</th>
<th># Hidden units</th>
<th>Err.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNN</td>
<td>300k-300k+ dropout</td>
<td>~10M</td>
<td>4k</td>
</tr>
<tr>
<td>DSNN-10k</td>
<td>13k-100k-10k+ dropout capability</td>
<td>~7M</td>
<td>~100k</td>
</tr>
<tr>
<td>Single layer</td>
<td>followed by SVM</td>
<td>~12M</td>
<td>~2k</td>
</tr>
<tr>
<td>CNN1 (no</td>
<td>512-1024-512-512</td>
<td>~10k</td>
<td>~10k</td>
</tr>
<tr>
<td>convolution)</td>
<td>no augmentation</td>
<td>~500k</td>
<td>~120k</td>
</tr>
<tr>
<td>Teacher CNN</td>
<td>128-128-128-128-128-128</td>
<td>~500k</td>
<td>~20k</td>
</tr>
<tr>
<td>(no augmentation)</td>
<td>ensemble of 4 CNNs</td>
<td>~140k</td>
<td>~80k</td>
</tr>
<tr>
<td>SNN/CNN-MD-12k</td>
<td>4k-512-128-128 with re-regularization</td>
<td>~50k</td>
<td>~11k</td>
</tr>
<tr>
<td>SNN4/CNN-MD-12k</td>
<td>trained on a single CNN with no re-regularization</td>
<td>~700k</td>
<td>~100k</td>
</tr>
<tr>
<td>SNN/SNN-MD-12k</td>
<td>128-130k-128 with 0 regularization</td>
<td>~700k</td>
<td>~100k</td>
</tr>
</tbody>
</table>

Table: Comparison of shallow and deep models: classification error rate on CIFAR-10. "c" convolutional layer, "p" pooling layer, "k" locally connected layer, "fc" fully connected layer.
Typically when you train with SGD, you’ll want to begin with a higher learning rate, and then drop the learning rate a few times during training.

**Beyond SGD and Momentum**

- Methods like AdaGrad and AdaDelta apply a different effective learning rate to each weight.
  - Each element of each convolutional kernel, fully connected layer weight, bias vector.
  - Basic idea: keep track of gradient magnitudes, and divide the learning rate by that
    - If certain layers get higher magnitude gradients, use a lower learning rate to aid stability.
    - If certain layers get lower magnitude / vanishing gradients, use a higher learning rate to “boost” them

Most popular in use today: **Adam**

- Keep running average of average of gradient values and magnitudes
- Correct for “bias” in their values.

**OTHER ARCHITECTURES**

- For object detection, RCNN takes all potential bounding boxes, resizes, and sends them through the network.

  - With overlapping boxes, there’s intuitively some computation that’s redundant.
  - But can’t apply the fully convolutional trick directly.
**OTHER ARCHITECTURES**

**One Solution**: Faster R-CNN (by Ross Girshick)

- Generate a feature map: \( W \times H \times C \)
- Then given a bunch of bounding boxes, (max / average) pool within each box
- Gives you a C dimensional vector for each box.
- Then have fully connected layers for each vector.
- Use the fact that pooling automatically adapts to arbitrary shapes.

**ROI pooling**

- Basic idea: convolution with up-sampling
- Sometimes called deconvolution (but this terminology is discouraged, because that means something else)
- Implementation wise: if you implement conv with downsampling, the ‘backward’ step is conv-transpose.
- You also have “unpooling” layers.

**Other Architectures**

**Transposed Convolutions**

- Take global information to a single vector, and then spread it back to the image.
- Modern versions of this will have skip or residual connections between initial and final layers.
OTHER ARCHITECTURES

- What this solves is the fact that in a feedforward neural network, reduction in scale happens jointly with increasing complexity.
- But you might need some combination of global/semantic and local/image-level information for a task.

Another way to do this: Hypercolumns

OTHER APPLICATIONS

- Deep neural networks now used in nearly all applications across vision
- Not just semantic tasks, but tasks that can benefit from semantic reasoning
- Or even tasks where neural nets are a better way of encoding complex models about texture, materials, etc.
OTHER APPLICATIONS

Stereo

Replace Census Transform + Hamming Distance with an NN

Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches

Jure Žiganter
Faculty of Computer and Information Science
University of Ljubljana
Vihova pot 125, SI-1000 Ljubljana, Slovenia
JURE.ZIGNAT@FRI.UJI.LJ.SI

Yann LeCun
Courant Institute of Mathematical Sciences
New York University
212 Broadway, New York, NY 10003, USA
YANN@CAMS.YU.EDU

OTHER APPLICATIONS

Deblurring with Deep Learning

A Neural Approach to Blind Motion Deblurring

Ayan Chakrabarti

Depth Estimation without Geometric Cues

Depth from a Single Image by Harmonizing Overcomplete Local Network Predictions

Ayan Chakrabarti, Jingyu Shao, Gregory Shakhnarovich
NEURAL STYLE TRANSFER

Image Style Transfer Using Convolutional Neural Networks

Leon A. Guyer
Centre for Integrative Neuroscience, University of Tübingen, Germany
Graduate School of Neural Information Processing, University of Tübingen, Germany

Alexander S. Fikar
Centre for Integrative Neuroscience, University of Tübingen, Germany
Bernstein Center for Computational Neuroscience, Tübingen, Germany
Max Planck Institute for Biological Cybernetics, Tübingen, Germany
Bayler College of Medicine, Houston, TX, USA

Matthias Bethge
Centre for Integrative Neuroscience, University of Tübingen, Germany
Bernstein Center for Computational Neuroscience, Tübingen, Germany
Max Planck Institute for Biological Cybernetics, Tübingen, Germany

- Take a pre-trained imagenet network (like VGG-16)
- Now given two images: one of content $p$ and one of style $q$
- Find an image $x$ such that the activations of different layers from passing $x$ through VGG
  - Are similar to those of $p$ for some layers
  - Have similar co-variances to those of $q$ for some other layers

- Minimize this by gradient descent to $x$
NEURAL STYLE TRANSFER

https://github.com/anishathalye/neural-style

GENERATIVE ADVERSARIAL NETWORKS

Ian Goodfellow et al., 2014.

- In traditional Training: Input, correct output, loss
  - What if there’s more than one output
  - What if you don’t know what the loss function should be
  - What if you need to generate more than one output (get many possible answers)

GANs will train a generator network: $y = G(z)$ (also, conditional GANs $y = G(x; z)$)

- $z$ is a “noise vector” from some known distribution
- You will call $G$ multiple times with different $z$
- And $G$ will produce samples from the distribution $p(y)$ or $p(y|x)$
- You can think of the function $G$ being the “inverse-cdf” of $p(y)$ wrt the distribution $z$ ...

(Some versions don’t include a $z$, not really GANs, but is called training with an adversarial loss)

GENERATIVE ADVERSARIAL NETWORKS

- But the generator isn’t trained with “direct” supervision
- Rather, it’s trained “adversarially” to “fool” a discriminator network $D$

$$
z \Rightarrow G \Rightarrow y_{fake} \Rightarrow D \leftarrow label=fake$$

$$y_{real} \Rightarrow D \leftarrow label=real$$

- Train $D$ that produces probability of real/fake to minimize cross-entropy loss
- Train $G$ to maximize this loss and fool $D$

$$\max_D \min_G \mathbb{E}_{y \sim P_{real}} \log D(y) + \mathbb{E}_{z \sim P_z} \log(1 - D(G(z)))$$

- $G$ gets gradients “through” $D$
- This is a min-max optimization. The stationary point of this game is when $D$ always outputs probability 0.5, and $G(z)$ are samples from $P_{real}$.
- In practice, optimize this by making alternating updates.
  - To update generator, instead of maximizing $-\log(1 - D(G(z)))$, minimize $\log D(G(z))$

GENERATIVE ADVERSARIAL NETWORKS

- Common architecture for generating images: DCGAN
  - Radford, Metz, Chintala (ICLR 2016)

- In general, GAN training is unstable. Typically, “discriminator” always wins.
  - But lots of methods are trying to address this now.
GENERATIVE ADVERSARIAL NETWORKS

- Neyshabur, Bhojanapalli, Chakrabarti, Stabilizing GAN Training with Multiple Random Projections.
  - Basic idea, train a single generator against multiple discriminators.

Some examples: $L^R G(\alpha z_1 + (1-\alpha)z_2)$

Current Uses

- "Unsupervised" Training
  - You have lots of unlabeled data, train GAN
  - Use Discriminator as a "pre-trained" network
  - Instead of imagenet, pre-trained on real/fake

Adversarial Loss: Conditional GANs

- When you don’t know what the right loss should be
- There’s no one single answer, but you want to produce something “plausible”

UN- / SELF-SUPERVISED LEARNING

Will be the next big challenge

- Try to pre-train on the data itself without labels
- Pre-train by predicting sound from video

Image-to-Image Translation with Conditional Adversarial Nets

UN- / SELF-SUPERVISED LEARNING

Visually Indicated Sounds

Examples of sounds from the Greatest Hits dataset. Click each image to play.
UN-/SELF-SUPERVISED LEARNING

- Pre-train by solving jigsaw puzzles

Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles

Mehdi Norouzi and Paolo Fua
Institute for Informatics
University of Bocconi

That’s all folks!

Larsson, Maire, Shakhnarovich, CVPR 2017.

UN-/SELF-SUPERVISED LEARNING

- Pre-train by learning to add color