Lecture 3:
Some Advanced Topics in Deep Learning

Instructor: Hao Su
Jan 16, 2018
Agenda

• Network Visualization

• Matrix Calculus and Batch Normalization

• Optimization for Networks

• Theories behind Network Generalizability
What’s going on inside ConvNets?

Input Image: 3 x 224 x 224

What are the intermediate features looking for?

Class Scores: 1000 numbers

Krizhevsky et al., “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS 2012. Figure reproduced with permission.
First Layer: Visualize Filters

AlexNet:
64 x 3 x 11 x 11

ResNet-18:
64 x 3 x 7 x 7

ResNet-101:
64 x 3 x 7 x 7

DenseNet-121:
64 x 3 x 7 x 7
Last Layer

4096-dimensional feature vector for an image (layer immediately before the classifier)

Run the network on many images, collect the feature vectors
Last Layer: Nearest Neighbors

4096-dim vector

Test image  L2 Nearest neighbors in feature space

Recall: Nearest neighbors in pixel space

Krizhevsky et al., "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012. Figures reproduced with permission.
Last Layer: Dimensionality Reduction

Visualize the “space” of FC7 feature vectors by reducing dimensionality of vectors from 4096 to 2 dimensions

Simple algorithm: Principle Component Analysis (PCA)

More complex: t-SNE

Van der Maaten and Hinton. “Visualizing Data using t-SNE”, JMLR 2008
Figure copyright Laurens van der Maaten and Geoff Hinton, 2008. Reproduced with permission.
Visualizing Activations

conv5 feature map is 128x13x13; visualize as 128 13x13 grayscale images

Maximally Activating Patches

Pick a layer and a channel; e.g. conv5 is 128 x 13 x 13, pick channel 17/128

Run many images through the network, record values of chosen channel

Visualize image patches that correspond to maximal activations

Figure copyright Jost Tobias Springenberg, Alexey Dosovitsky, Thomas Brox, Martin Riedmiller, 2015, reproduced with permission.

[Stanford CS231n]
Saliency Maps

How to tell which pixels matter for classification?

Dog
Saliency Maps

How to tell which pixels matter for classification?

Compute gradient of (unnormalized) class score with respect to image pixels, take absolute value and max over RGB channels


Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

[Stanford CS231n]
Saliency Maps


[Stanford CS231n]
Intermediate Features via (guided) backprop

Pick a single intermediate neuron, e.g. one value in 128 x 13 x 13 conv5 feature map

Compute gradient of neuron value with respect to image pixels

Zeller and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014
Visualizing CNN features: Gradient Ascent

(Guided) backprop:
Find the part of an image that a neuron responds to

Gradient ascent:
Generate a synthetic image that maximally activates a neuron

\[ I^* = \text{arg max}_I \left[ f(I) + R(I) \right] \]

Neuron value
Natural image regularizer

[Stanford CS231n]
Visualizing CNN features: Gradient Ascent

1. Initialize image to zeros

   zero image

2. Forward image to compute current scores

3. Backprop to get gradient of neuron value with respect to image pixels

4. Make a small update to the image

\[
\arg\max_I \left[ S_c(I) - \lambda \| I \|^2 \right]
\]

score for class c (before Softmax)
Visualizing CNN features: Gradient Ascent

\[ \arg \max_I S_c(I) - \lambda \| I \|_2^2 \]

Simple regularizer: Penalize L2 norm of generated image

- washing machine
- computer keyboard
- kit fox
- goose
- ostrich
- limousine

Yosinski et al., "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014. Figure copyright Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, 2014. Reproduced with permission.
Network Visualization

• Still an open area!

• For only for eyes, but to induce useful characteristics for network diagnosis
Fooling Images / Adversarial Examples

(1) Start from an arbitrary image
(2) Pick an arbitrary class
(3) Modify the image to maximize the class
(4) Repeat until network is fooled
Fooling Images / Adversarial Examples

African elephant

koala

Difference

10x Difference

schooner

iPod

Difference

10x Difference

[Stanford CS231n]
Agenda

• Network Visualization

• Batch Normalization and Matrix Calculus

• Optimization for Networks

• Theories behind Network Generalizability
Batch Normalization

Algorithm 2 Batch normalization [Ioffe and Szegedy, 2015]

Input: Values of $x$ over minibatch $x_1 \ldots x_B$, where $x$ is a certain channel in a certain feature vector

Output: Normalized, scaled and shifted values $y_1 \ldots y_B$

1: $\mu = \frac{1}{B} \sum_{b=1}^{B} x_b$
2: $\sigma^2 = \frac{1}{B} \sum_{b=1}^{B} (x_b - \mu)^2$
3: $\hat{x}_b = \frac{x_b - \mu}{\sqrt{\sigma^2 + \epsilon}}$
4: $y_b = \gamma \hat{x}_b + \beta$

- Accelerates training and makes initialization less sensitive
- Zero mean and unit variance feature vectors

[Stanford STATS385]
A simple example:

\[ y = W_3 W_2 W_1 x \]

\[ W_3 \in \mathbb{R}^{1 \times 100}, \quad W_2 \in \mathbb{R}^{100 \times 100}, \quad W_1 \in \mathbb{R}^{100 \times 10}, \quad x \in \mathbb{R}^{10}, \| x \| \approx 1 \]

All elements sampled i.i.d from \( N(0,1) \)

\[
\frac{\partial y}{\partial W_1} = ? \quad \frac{\partial y}{\partial W_2} = ? \quad \frac{\partial y}{\partial W_3} = ?
\]

Hint:

\[
tr(ABC) = tr(CAB) = tr(BCA)
\]

\[
\nabla_A tr AB = B^T
\]
Calculus Example

• A simple example:

\[ y = W_3 W_2 W_1 x \]

\[ W_3 \in \mathbb{R}^{1 \times 100}, \quad W_2 \in \mathbb{R}^{100 \times 100}, \quad W_1 \in \mathbb{R}^{100 \times 10}, \quad x \in \mathbb{R}^{10}, \| x \| \approx 1 \]

All elements sampled i.i.d from \( N(0,1) \)

\[ \frac{\partial y}{\partial W_1} = ? \quad \frac{\partial y}{\partial W_2} = ? \quad \frac{\partial y}{\partial W_3} = ? \]

\[ \mathbb{E} \| W_1 \|^2_F = ? \quad \mathbb{E} \| W_2 \|^2_F = ? \quad \mathbb{E} \| W_3 \|^2_F = ? \]

Hint: \( \text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y) \)
Calculus Example

• A simple example:

\[ y = W_3 W_2 W_1 x \]

\[ W_3 \in \mathbb{R}^{1 \times 100}, \quad W_2 \in \mathbb{R}^{100 \times 100}, \quad W_1 \in \mathbb{R}^{100 \times 10}, \quad x \in \mathbb{R}^{10}, \| x \| \approx 1 \]

All elements sampled i.i.d from \( N(0,1) \)

\[ \frac{\partial y}{\partial W_1} = ? \quad \frac{\partial y}{\partial W_2} = ? \quad \frac{\partial y}{\partial W_3} = ? \]

\[ \mathbb{E} \| W_1 \|_F^2 = ? \quad \mathbb{E} \| W_2 \|_F^2 = ? \quad \mathbb{E} \| W_3 \|_F^2 = ? \]

Let \( y_1 = W_1 x, \quad y_2 = W_2 W_1 x \)

\[ \mathbb{E} \| y_1 \|_2^2 \leq ? \quad \mathbb{E} \| y_2 \|_2^2 \leq ? \quad \mathbb{E} \| y \|_2^2 \leq ? \quad \text{Hint: } \| A \|_2 \leq \| A \|_F \]
Why Batch Normalization is Effective?

• A simple example:

\[ y = W_3 s_3 W_2 s_2 W_1 s_1 x \]

\[ W_3 \in \mathbb{R}^{1 \times 100}, \quad W_2 \in \mathbb{R}^{100 \times 100}, \quad W_1 \in \mathbb{R}^{100 \times 10}, \quad x \in \mathbb{R}^{10}, \|x\| \approx 1 \]

All elements sampled i.i.d from \( N(0,1) \)

\[ \frac{\partial y}{\partial W_1} = ? \quad \frac{\partial y}{\partial W_2} = ? \quad \frac{\partial y}{\partial W_3} = ? \]
Why Batch Normalization is Effective?

• A simple example:

\[ y = W_3 s_3 W_2 s_2 W_1 s_1 x \]

\[ W_3 \in \mathbb{R}^{1 \times 100}, \quad W_2 \in \mathbb{R}^{100 \times 100}, \quad W_1 \in \mathbb{R}^{100 \times 10}, \quad x \in \mathbb{R}^{10}, \|x\| \approx 1 \]

All elements sampled i.i.d from \( N(0,1) \)

\[ \frac{\partial y}{\partial W_1} = ? \quad \frac{\partial y}{\partial W_2} = ? \quad \frac{\partial y}{\partial W_3} = ? \]

Update rule:

\[ \Delta w^{t+1} = w^t + \eta \nabla E(w^t) \]

\( \eta \) : step size, e.g., stage-wise constant
Agenda

• Network Visualization

• Matrix Calculus and Batch Normalization

• Optimization for Networks

• Theories behind Network Generalizability
Reminder: The error surface for a linear neuron

- The error surface lies in a space with a horizontal axis for each weight and one vertical axis for the error.
  - For a linear neuron with a squared error, it is a quadratic bowl.
  - Vertical cross-sections are parabolas.
  - Horizontal cross-sections are ellipses.
- For multi-layer, non-linear nets the error surface is much more complicated.
  - But locally, a piece of a quadratic bowl is usually a very good approximation.
Convergence speed of full batch learning when the error surface is a quadratic bowl

- Going downhill reduces the error, but the direction of steepest descent does not point at the minimum unless the ellipse is a circle.
  - The gradient is big in the direction in which we only want to travel a small distance.
  - The gradient is small in the direction in which we want to travel a large distance.

Even for non-linear multi-layer nets, the error surface is locally quadratic, so the same speed issues apply.
How the learning goes wrong

• If the learning rate is big, the weights slosh to and fro across the ravine.
  – If the learning rate is too big, this oscillation diverges.
• What we would like to achieve:
  – Move quickly in directions with small but consistent gradients.
  – Move slowly in directions with big but inconsistent gradients.
Stochastic gradient descent

• If the dataset is highly redundant, the gradient on the first half is almost identical to the gradient on the second half.
  – So instead of computing the full gradient, update the weights using the gradient on the first half and then get a gradient for the new weights on the second half.
  – The extreme version of this approach updates weights after each case. It's called “online”.

• Mini-batches are usually better than online.
  – Less computation is used updating the weights.
  – Computing the gradient for many cases simultaneously uses matrix-matrix multiplies which are very efficient, especially on GPUs.

• Mini-batches need to be balanced for classes
Momentum Method

The intuition behind the momentum method

Imagine a ball on the error surface. The location of the ball in the horizontal plane represents the weight vector.

- The ball starts off by following the gradient, but once it has velocity, it no longer does steepest descent.
- Its momentum makes it keep going in the previous direction.

- It damps oscillations in directions of high curvature by combining gradients with opposite signs.
- It builds up speed in directions with a gentle but consistent gradient.

[UToronto CSC321]
Momentum Method

\[ v(t) = \alpha \, v(t-1) - \varepsilon \, \frac{\partial E}{\partial w}(t) \]

The effect of the gradient is to increment the previous velocity. The velocity also decays by \( \alpha \) which is slightly less than 1.

\[ \Delta w(t) = v(t) \]

The weight change is equal to the current velocity.

\[ = \alpha \, v(t-1) - \varepsilon \, \frac{\partial E}{\partial w}(t) \]

\[ = \alpha \, \Delta w(t-1) - \varepsilon \, \frac{\partial E}{\partial w}(t) \]

The weight change can be expressed in terms of the previous weight change and the current gradient.
Momentum Method

- It leads to faster and stable convergence.
- Reduced Oscillations

ADAM: An Improved Moment Method

First order and second order moment estimation

\[ m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \]
\[ v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \]

Bias correction

\[ \hat{m}_t = \frac{m_t}{1 - \beta_1^t} \]
\[ \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \]

Adam update rule:

\[ \theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \]

[UToronto CSC321]
Learning Curve

![Learning Curve Graph]

[Hao Su] [UToronto CSC321]
Agenda

• Network Visualization

• Matrix Calculus and Batch Normalization

• Optimization for Networks

• Analysis behind Network Generalizability
Model Selection

Underfitting

Overfitting

[Chiyuan Zhang, ICLR’17]
Deep Learning
Bias — Variance

[Chiyuan Zhang, ICLR'17]
### Parameter Count vs. Num Training Samples

- **MLP 1x512**
  - Parameter Count: 24
  - Num Training Samples: 12.5

- **Alexnet**
  - Parameter Count: 28
  - Num Training Samples: 37.5

- **Inception**
  - Parameter Count: 33
  - Num Training Samples: 25

- **Wide Resnet**
  - Parameter Count: 179
  - Num Training Samples: 50

---

**Test Error**

- **MLP 1x512**
  - Test Error: 37.5

- **Alexnet**
  - Test Error: 25

- **Inception**
  - Test Error: 12.5

- **Wide Resnet**
  - Test Error: 0

---

[Chiyuan Zhang, ICLR'17]
Random Label Dataset

[Chiyuan Zhang, ICLR'17]
Randomization Test

[Chiyuan Zhang, ICLR’17]

The graph illustrates the relationship between the ratio of random label noise and the accuracy of training and testing for different levels of label noise: No Label Noise and Full Label Noise. The x-axis represents the ratio of random label noise, ranging from 0 to 1, while the y-axis shows the accuracy, ranging from 0 to 100. The bars for train accuracy are in blue, and those for test accuracy are in green.
Randomization Test

[Chiyuan Zhang, ICLR’17]

Train Accuracy

Test Accuracy

Generalization Gap

No Label Noise 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Ratio of Random Label Noise

Full Label Noise
Randomization Test

- Deep Neural Networks Easily fit random labels

[Chiyuan Zhang, ICLR’17]
Regularizers in Deep Learning

- Data augmentation: domain-specific transformations
- Weight decay: l2-regularizer on weights
- Dropout*: randomly mask out responses

Fitting Random Label with Regularizers

[Chiyuan Zhang, ICLR’17]

<table>
<thead>
<tr>
<th>Regularizer</th>
<th>Model</th>
<th>Training Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight decay</td>
<td>Inception</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Alexnet</td>
<td>Failed to converge</td>
</tr>
<tr>
<td></td>
<td>MLP 1x512</td>
<td>99.21%</td>
</tr>
<tr>
<td>Crop Augmentation*</td>
<td>Inception</td>
<td>99.93%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Regularizer</th>
<th>Model</th>
<th>Training top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dropout</td>
<td>Inception V3</td>
<td>96.15%</td>
</tr>
<tr>
<td>Dropout + Weight decay</td>
<td>Inception V3</td>
<td>97.95%</td>
</tr>
</tbody>
</table>

*We need to tune the hyperparams a bit and run for more epochs for this to converge, see paper for details.
Implicit Regularization

[Chiyuan Zhang, ICLR’17]

A regularizer is a mechanism that constrains the model or empowers the data, hurting the training process.
Next Class

• The **optimization algorithm** and **landscape of the loss function** have to be taken into consideration

• Sketch of some latest theoretical investigation into deep learning