Introduction to Convolutional Networks

CIFAR Summer School 2016

Rob Fergus

Facebook AI Research
New York University
Overview

• Look at some of the recent progress with Convolutional Network models
  – Assume familiarity with basic neural nets

• Non-exhaustive coverage
  – Huge number of recent papers

• Review some computer vision applications
Convolutional Neural Networks

- LeCun et al. 1989
- Neural network with specialized connectivity structure
Multistage Hubel-Wiesel Architecture

- Stack multiple stages of simple cells / complex cells layers
- Higher stages compute more global, more invariant features
- Classification layer on top

History:
- Neocognitron [Fukushima 1971-1982]
- Convolutional Nets [LeCun 1988-2007]
- HMAX [Poggio 2002-2006]
- Many others…. 
Overview of Convnets

- Feed-forward:
  - Convolve input
  - Non-linearity (rectified linear)
  - Pooling (local max)
- Supervised
- Train convolutional filters by back-propagating classification error
Convnet Successes

• Handwritten text/digits
  – MNIST (0.17% error [Ciresan et al. 2011])
  – Arabic & Chinese [Ciresan et al. 2012]

• Simpler recognition benchmarks
  – CIFAR-10 (9.3% error [Wan et al. 2013])
  – Traffic sign recognition
    • 0.56% error vs 1.16% for humans [Ciresan et al. 2011]

• But less good at more complex datasets
  – E.g. Caltech-101/256 (few training examples)
Application to ImageNet

• ~14 million labeled images, 20k classes
• Images gathered from Internet
• Human labels via Amazon Turk

[Deng et al. CVPR 2009]
Goal

- Image Recognition
  - Pixels $\rightarrow$ Class Label

[Krizhevsky et al. NIPS 2012]
Krizhevsky et al. [NIPS2012] 

- Same model as LeCun’98 but:
  - Bigger model (8 layers)
  - More data  \( (10^6 \text{ vs } 10^3 \text{ images}) \)
  - GPU implementation (50x speedup over CPU)
  - Better regularization (DropOut)

- 7 hidden layers, 650,000 neurons, 60,000,000 parameters
- Trained on 2 GPUs for a week
ImageNet Classification (2010 – 2015)

Top-5 Classification Error (%)
Examples

• From Clarifai.com

Predicted Tags:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>food</td>
<td>(16.00%)</td>
</tr>
<tr>
<td>dinner</td>
<td>(3.10%)</td>
</tr>
<tr>
<td>bbq</td>
<td>(2.90%)</td>
</tr>
<tr>
<td>market</td>
<td>(2.50%)</td>
</tr>
<tr>
<td>meal</td>
<td>(1.40%)</td>
</tr>
<tr>
<td>turkey</td>
<td>(1.40%)</td>
</tr>
<tr>
<td>grill</td>
<td>(1.30%)</td>
</tr>
<tr>
<td>pizza</td>
<td>(1.30%)</td>
</tr>
<tr>
<td>eat</td>
<td>(1.10%)</td>
</tr>
<tr>
<td>holiday</td>
<td>(1.00%)</td>
</tr>
</tbody>
</table>

Stats:

Size: 247.24 KB
Time: 110 ms
Examples

• From Clarifai.com
Examples

• From Clarifai.com

Predicted Tags:

- barcelona (6.50%)
- street (3.00%)
- cave (2.20%)
- sagrada (1.90%)
- old (1.80%)
- night (1.40%)
- familia (1.40%)
- jerusalem (1.40%)
- guanajuato (1.10%)
- alley (1.00%)

Stats:
Size: 278.96 KB
Time: 113 ms
Using Features on Other Datasets

• Train model on ImageNet 2012 training set

• Re-train classifier on new dataset
  – Just the top layer (softmax)

• Classify test set of new dataset
The Details

• Operations in each layer

• Architecture

• Training

• Results
Components of Each Layer

Pixels / Features

Filter with learned dictionary

Non-linearity

Spatial local max pooling

Output Features
Filtering

- Convolution
  - Filter is learned during training
  - Same filter at each location
Filtering

- Local
  - Each unit layer above look at local window
  - But no weight tying

- E.g. face recognition
Filtering

- **Tiled**
  - Filters repeat every n
  - More filters than convolution for given # features
Non-Linearity

- Rectified linear function
  - Applied per-pixel
  - output = max(0,input)

Input feature map

Output feature map

Black = negative; white = positive values

Only non-negative values
**Non-Linearity**

- **Other choices:**
  - **Tanh**
  - **Sigmoid:** $1/(1+\exp(-x))$
  - **PReLU**

Pooling

- **Spatial Pooling**
  - Non-overlapping / overlapping regions
  - Sum or max
  - Boureau et al. ICML’10 for theoretical analysis
• Pooling across feature groups
  • Additional form of inter-feature competition
  • MaxOut Networks [Goodfellow et al. ICML 2013]
Role of Pooling

- Spatial pooling
  - Invariance to small transformations
  - Larger receptive fields
    (see more of input)

Visualization technique from [Le et al. NIPS’10]:

Videos from: http://ai.stanford.edu/~quocle/TCNNweb
Components of Each Layer

- Pixels / Features
- Filter with learned dictionary
- Non-linearity
- Spatial local max pooling
- [Optional] Normalization across data/features
- Output Features
Normalization

- Contrast normalization across features
  - See Divisive Normalization in Neuroscience
• Contrast normalization (across feature maps)
  – Local mean = 0, local std. = 1, “Local” \(\rightarrow\) 7x7 Gaussian
  – Equalizes the features maps
Role of Feature Normalization

- Introduces local competition between features
  - “Explaining away” in graphical models
  - Just like top-down models
  - But more local mechanism

- Also helps to scale activations at each layer better for learning
  - Makes energy surface more isotropic
  - So each gradient step makes more progress

- Empirically, seems to help a bit (1-2%) on ImageNet
- Most recent models don’t seem to have use though
Normalizing across Data

• Batch Normalization


Algorithm 1: Batch Normalizing Transform, applied to activation $x$ over a mini-batch.

Input: Values of $x$ over a mini-batch: $\mathcal{B} = \{x_1...m\}$; Parameters to be learned: $\gamma, \beta$

Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i$, $\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2$, $\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$, $y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i)$

Figure 2: Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.
Overview of Convnets

- **Feed-forward:**
  - Convolve input
  - Non-linearity (rectified linear)
  - Pooling (local max)

- **Supervised**

- **Train convolutional filters by back-propagating classification error**

---

Input Image → Convolution (Learned) → Non-linearity → Pooling → Feature maps
Architecture

• Big issue: how to select
  – Manual tuning of features ➔ manual tuning of architecture

• Depth
• Width
• Parameter count
How to Choose Architecture

• Many hyper-parameters:
  – # layers, # feature maps

• Cross-validation

• Grid search (need lots of GPUs)

• Smarter strategies:
  – Random [Bergstra & Bengio JMLR 2012]
  – Gaussian processes [Hinton??]
How important is Depth

- “Deep” in Deep Learning
- Ablation study
- Tap off features
Architecture of Krizhevsky et al.

- 8 layers total
- Trained on Imagenet dataset [Deng et al. CVPR’09]
- 18.2% top-5 error
- Our reimplementation: 18.1% top-5 error
Architecture of Krizhevsky et al.

- Remove top fully connected layer
  - Layer 7

- Drop 16 million parameters

- Only 1.1% drop in performance!
Architecture of Krizhevsky et al.

- Remove both fully connected layers
  - Layer 6 & 7
- Drop ~50 million parameters
- 5.7% drop in performance
Architecture of Krizhevsky et al.

- Now try removing upper feature extractor layers:
  - Layers 3 & 4
- Drop ~1 million parameters
- 3.0% drop in performance
Architecture of Krizhevsky et al.

- Now try removing upper feature extractor layers & fully connected:
  - Layers 3, 4, 6, 7
- Now only 4 layers
- 33.5% drop in performance

→ Depth of network is key
## Tapping off Features at each Layer

Plug features from each layer into linear SVM or soft-max

<table>
<thead>
<tr>
<th></th>
<th>Cal-101 (30/class)</th>
<th>Cal-256 (60/class)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (1)</td>
<td>44.8 ± 0.7</td>
<td>24.6 ± 0.4</td>
</tr>
<tr>
<td>SVM (2)</td>
<td>66.2 ± 0.5</td>
<td>39.6 ± 0.3</td>
</tr>
<tr>
<td>SVM (3)</td>
<td>72.3 ± 0.4</td>
<td>46.0 ± 0.3</td>
</tr>
<tr>
<td>SVM (4)</td>
<td>76.6 ± 0.4</td>
<td>51.3 ± 0.1</td>
</tr>
<tr>
<td>SVM (5)</td>
<td>86.2 ± 0.8</td>
<td>65.6 ± 0.3</td>
</tr>
<tr>
<td>SVM (7)</td>
<td>85.5 ± 0.4</td>
<td>71.7 ± 0.2</td>
</tr>
<tr>
<td>Softmax (5)</td>
<td>82.9 ± 0.4</td>
<td>65.7 ± 0.5</td>
</tr>
<tr>
<td>Softmax (7)</td>
<td>85.4 ± 0.4</td>
<td>72.6 ± 0.1</td>
</tr>
</tbody>
</table>
Translation (Vertical)

Layer 1

Layer 7

Output

Lawn Mower
Shih-Tzu
African Crocodile
African Grey
Entertrainment Center

Vertical Translation (Pixels)

Canonical Distance

P(true class)
Scale Invariance

Layer 1

Output

Layer 7
Rotation Invariance

Layer 1

Layer 7

Output

Rotation Degrees

Canonical Distance

Lawn Mower
Shih−Tzu
African Crocodile
African Grey
Entertrainment Center

Rotation Degrees

Canonical Distance

Lawn Mower
Shih−Tzu
African Crocodile
African Grey
Entertrainment Center
Very Deep Models (1)


ConvNet Configuration

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A-LRN</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>input (224 × 224 RGB image)</td>
<td>conv3-64</td>
<td>LRN</td>
<td>conv3-64</td>
<td>conv3-64</td>
<td>conv3-64</td>
<td>conv3-64</td>
</tr>
<tr>
<td></td>
<td>maxpool</td>
<td></td>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
</tr>
<tr>
<td></td>
<td>conv3-256</td>
<td></td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
</tr>
<tr>
<td></td>
<td>maxpool</td>
<td></td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
</tr>
<tr>
<td></td>
<td>conv3-512</td>
<td></td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
</tr>
<tr>
<td></td>
<td>maxpool</td>
<td></td>
<td>FC-4096</td>
<td>FC-4096</td>
<td>FC-1000</td>
<td>soft-max</td>
</tr>
</tbody>
</table>

- Lots of 3x3 conv layers: more non-linearity than single 7x7 layer
- Close to SOA results on Imagenet: 6.8% top-5 val
- Can be hard to train

**Table 1:** ConvNet configurations

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A-LRN</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 weight layers</td>
<td>11 weight layers</td>
<td>13 weight layers</td>
<td>16 weight layers</td>
<td>16 weight layers</td>
<td>19 weight layers</td>
<td></td>
</tr>
</tbody>
</table>
| Table 2: **Number of parameters** (in millions).

<table>
<thead>
<tr>
<th>Network</th>
<th>A</th>
<th>A-LRN</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of parameters</td>
<td>133</td>
<td>133</td>
<td>134</td>
<td>138</td>
<td>144</td>
<td></td>
</tr>
</tbody>
</table>

**Table 3:** ConvNet performance at a single test scale.

<table>
<thead>
<tr>
<th>ConvNet config. (Table 1)</th>
<th>smallest image side</th>
<th>top-1 val. error (%)</th>
<th>top-5 val. error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>256</td>
<td>29.6</td>
<td>10.4</td>
</tr>
<tr>
<td>A-LRN</td>
<td>256</td>
<td>29.7</td>
<td>10.5</td>
</tr>
<tr>
<td>B</td>
<td>256</td>
<td>28.7</td>
<td>9.9</td>
</tr>
<tr>
<td>C</td>
<td>384</td>
<td>28.1</td>
<td>9.3</td>
</tr>
<tr>
<td>D</td>
<td>256</td>
<td>27.0</td>
<td>8.8</td>
</tr>
<tr>
<td>E</td>
<td>384</td>
<td>26.8</td>
<td>8.7</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>25.6</td>
<td>8.1</td>
</tr>
<tr>
<td></td>
<td>[256;512]</td>
<td>25.6</td>
<td>9.0</td>
</tr>
<tr>
<td></td>
<td>384</td>
<td>26.9</td>
<td>8.7</td>
</tr>
<tr>
<td></td>
<td>[256;512]</td>
<td>25.5</td>
<td>8.0</td>
</tr>
</tbody>
</table>
GoogLeNet inception module:

1. Multiple filter scales at each layer
2. Dimensionality reduction to keep computational requirements down

GoogLeNet vs Previous Models


Width of inception modules ranges from 256 filters (in early modules) to 1024 in top inception modules.

Can remove fully connected layers on top completely

Number of parameters is reduced to 5 million

6.7% top-5 validation error on Imagnet

Computational cost is increased by less than 2X compared to Krizhevsky’s network. (<1.5Bn operations/evaluation)

[From http://image-net.org/challenges/LSVRC/2014/slides/Go]
Residual Networks

[He, Zhang, Ren, Sun, CVPR 2016]

Really, really deep convnets don’t train well, E.g. CIFAR10:

Key idea: introduce “pass through” into each layer

Thus only residual now needs to be learned

\[ F(x) \]

weight layer

relu

\[ F(x) + x \]

weight layer

identity

Table 4. Error rates (%) of single-model results on the ImageNet validation set (except \(^1\) reported on the test set).

<table>
<thead>
<tr>
<th>method</th>
<th>top-1 err.</th>
<th>top-5 err.</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG [41] (ILSVRC’14)</td>
<td>-</td>
<td>8.43(^1)</td>
</tr>
<tr>
<td>GoogLeNet [44] (ILSVRC’14)</td>
<td>-</td>
<td>7.89</td>
</tr>
<tr>
<td>VGG [41] (v5)</td>
<td>24.4</td>
<td>7.1</td>
</tr>
<tr>
<td>BN-inception [16]</td>
<td>21.99</td>
<td>5.81</td>
</tr>
<tr>
<td>ResNet-34 B</td>
<td>21.84</td>
<td>5.71</td>
</tr>
<tr>
<td>ResNet-34 C</td>
<td>21.53</td>
<td>5.60</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>20.74</td>
<td>5.25</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>19.87</td>
<td>4.60</td>
</tr>
<tr>
<td>ResNet-152</td>
<td><strong>19.38</strong></td>
<td><strong>4.49</strong></td>
</tr>
</tbody>
</table>

With ensembling, 3.57% top-5 test error on ImageNet
Visualizing Convnets

- Want to know what they are learning

- Raw coefficients of learned filters in higher layers difficult to interpret

- Two classes of method:
  1. Project activations back to pixel space
  2. Optimize input image to maximize a particular feature map or class
Visualizing Convnets

- Projection from higher layers back to input
  - Several similar approaches:
    - Visualizing and Understanding Convolutional Networks, Matt Zeiler & Rob Fergus, ECCV 2014
Projection from Higher Layers

[Zeiler et al. ECCV14]
Details of Operation

Deconvnet layer

- Layer Above Reconstruction
- Max Unpooling
- Unpooled Maps
- Rectified Linear Function
- Rectified Unpooled Maps
- Convolutional Filtering \( \{F\} \)
- Reconstruction

Switches

Convnet layer

- Pooled Maps
- Max Pooling
- Rectified Feature Maps
- Rectified Linear Function
- Feature Maps
- Convolutional Filtering \( \{F\} \)
- Layer Below Pooled Maps
Unpooling Operation

Layer Above Reconstruction

Unpooling

Max Locations “Switches”

Unpooled Maps

Rectified Feature Maps

Pooled Maps

Pooling
Layer 1 Filters
Visualizations of Higher Layers

- Use ImageNet 2012 validation set
- Push each image through network

- Take max activation from feature map associated with each filter
- Use Deconvnet to project back to pixel space
- Use pooling “switches” peculiar to that activation
Layer 1: Top-9 Patches
Layer 2: Top-1
Layer 2: Top-9

- **NOT SAMPLES FROM MODEL**
- Just parts of input image that give strong activation of this feature map
- Non-parametric view on invariances learned by model
Layer 2: Top-9 Patches

- Patches from validation images that give maximal activation of a given feature map
Visualizing Convnets

• Optimize input to maximize particular output
  – Lots of approaches, e.g. Erhan et al. [Tech Report 2009], Le et al. [NIPS 2010].
  – Depend on initialization

• Google DeepDream
  [http://googleresearch.blogspot.ch/2015/06/inceptionism-going-deeper-into-neural.html]
  – Maximize “banana” output
Google DeepDream

https://photos.google.com/share/F1QipPX0SCL7OzWilt9LnuQliattX4OUCj_8EP65_cTVnBmS1jnYgsGQAieQUc1VQWdgQ/photo/AF1QipMYTXpt0TvZ0Q5kubkGw8VAq2isxBuL02wKZafB?key=aVBxWjhwSzhg2RjJWLRuVFBBZEN1d205bUdEMnhB
Training Big ConvNets

• **Stochastic Gradient Descent**
  – Compute (noisy estimate of) gradient on small batch of data & make step
  – Take as many steps as possible (even if they are noisy)
  – Large initial learning rate
  – Anneal learning rate

• **Momentum**
  – Variants [Sutskever ICML 2012]
Annealing of Learning Rate

- Start large, slowly reduce
- Explore different scales of energy surface
Evolution of Features During Training
Evolution of Features During Training
Normalisation across Data

- **Batch Normalization**


**Algorithm 1**: Batch Normalizing Transform, applied to activation $x$ over a mini-batch.

- **Input**: Values of $x$ over a mini-batch: $B = \{x_1...m\}$;
- **Parameters to be learned**: $\gamma, \beta$
- **Output**: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

\[
\begin{align*}
\mu_B &\leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i && \text{// mini-batch mean} \\
\sigma_B^2 &\leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2 && \text{// mini-batch variance} \\
\hat{x}_i &\leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} && \text{// normalize} \\
y_i &\leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) && \text{// scale and shift}
\end{align*}
\]

Figure 2: Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.
Automatic Tuning of Learning Rate?

- **ADAGRAD**

- **ADADELTA**

- **No more pesky learning rates**

\[
\Delta x_t = -\frac{\eta}{\sqrt{\sum_{\tau=1}^{t} g_{\tau}^2}} g_t
\]

\[
\Delta x_t = -\frac{\text{RMS}[\Delta x]_{t-1}}{\text{RMS}[g]_t} g_t
\]

\[
\Delta x_t = -\frac{1}{|\text{diag}(H_t)|} \frac{E[g_{t-w:t}]^2}{E[g_{t-w:t}^2]} g_t
\]
Local Minima?

[The Loss Surfaces of Multilayer Networks

Distribution of test losses
What about 2nd order methods?

- Newton’s method: \[ \Delta x_t = H_t^{-1} g_t \]
- Full Hessian impractical to compute
- Approximations:
  - Diagonal [Becker & Lecun ‘88]
  - Truncated CG [Martens, ICML’10]
  - Per-batch low-rank [Sohl-Dickstien et al., ICML’14]
  - Saddle free (|H|) [Dauphin et al. NIPS’14]

- Generally, extra computation needed seems not worth it: take more (dumb) steps instead!
Saddle Point Perspective

[Identifying and attacking the saddle point problem in high-dimensional non-convex optimization, Dauphin et al., NIPS 2014]

- During optimization Hessian has both +ve and –ve eigenvalues
  - and maybe some zeros too (flat directions)
  - At minimum, all are +ve

- Cause problems for SGD

- Saddle Free Newton (SFN)
  - Use |H| (matrix where take absolute value of each eigenvalue of H)
Improving Generalization

- Data Augmentation (jitter, perturb)
- Weight decay (L1/2 penalty on weights)
- Weight sharing (reduces # parameters)
- Multi-task learning
- Inject Noise into network
  - DropOut [Hinton et al. 2012]
  - DropConnect [Wan et al. ICML 2012]
  - Stochastic Pooling [Zeiler & Fergus ICLR’13]
Big Model + Regularize vs Small Model

Small model

Big model

Big model + Regularize
Fooling Convnets

- Search for images that are misclassified by the network
- Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images, Anh Nguyen, Jason Yosinski, Jeff Clune, arXiv 1412.1897.

- Problem common to any discriminative method

Figure 1. Evolved images that are unrecognizable to humans, but that state-of-the-art DNNs trained on ImageNet believe with \( \geq 99.6\% \) certainty to be a familiar object. This result highlights differences between how DNNs and humans recognize objects.
DropOut


- Fully connected layers only
- Randomly set activations in layer to zero
- Gives ensemble of models
- Similar to bagging [Breiman’94], but differs in that parameters are shared.
DropConnect

- Wan et al. ICML 2013
- Fully-connected layers only
- Random binary mask on weights

**Mixture Model Interpretation**

DropConnect Network is a mixture model of 2 classifiers:

\[ o = \mathbb{E}_{M} [f(x; \mathbf{\check{M}})] = \sum_{M} p(M) f(x; \mathbf{\check{M}}) \]

**Empirical Study of DropConnect Network**

- Understand DropConnect Network with MNIST data set
- 200
- 400
- 800
- 1600
- 1.1
- 1.2
- 1.3
- 1.4
- 1.5
- 1.6
- 1.7
- 1.8
- 1.9
- 2
- 2.1
- 2.2
- 2.4
- 100
- 200
- 300
- 400
- 500
- 600
- 700
- 800
- 900
- 10
- 3
- 10
- 2
- 10
- 3
- Epoch
- Cross Entropy
- MNIST

**Input**

\[ x \]

**Softmax layer**

\[ s(r; W_s) \]

**Predictions**

\[ o(k x 1) \]

**Feature extractor**

\[ g(x; W_g) \]

**Activation function**

\[ a(u) \]

**Outputs**

\[ r(d x 1) \]

**Features**

\[ v(n x 1) \]

**DropConnect mask**

\[ M \]

**Previous layer mask**

**Current layer output mask**

\[ M' \]

**Effective Dropout mask**

\[ M' \]

- Wan et al. ICML 2013
- Fully-connected layers only
- Random binary mask on weights

**Evaluating DropConnect Network**

1. Testing error by varying the size of the network
2. Testing error by varying the drop-rate in a 400-400 network
3. Convergence properties of the train/test sets.
Stochastic Pooling

- For conv layers
- Compute activations $a_i: (\geq 0)$
- Normalize to sum to 1 $\Rightarrow p_i = \frac{a_i}{\sum_{k \in R_j} a_k}$
- Sample location, $l$, from multinomial
- Use activation from the location: $s = a_l$

[Zeiler and Fergus, ICLR 2013]
OTHER THINGS GOOD TO KNOW

- Check gradients numerically by finite differences
- Visualize features (feature maps need to be uncorrelated) and have high variance.

Good training: hidden units are sparse across samples and across features.
OTHER THINGS GOOD TO KNOW

- Check gradients numerically by finite differences
- Visualize features (feature maps need to be uncorrelated) and have high variance.

Bad training: many hidden units ignore the input and/or exhibit strong correlations.
OTHER THINGS GOOD TO KNOW

- Check gradients numerically by finite differences
- Visualize features (feature maps need to be uncorrelated) and have high variance.
- Visualize parameters

**GOOD**

**BAD**

- too noisy
- too correlated
- lack structure

**Good training:** learned filters exhibit structure and are uncorrelated.
OTHER THINGS GOOD TO KNOW

- Check gradients numerically by finite differences
- Visualize features (feature maps need to be uncorrelated) and have high variance.
- Visualize parameters
- Measure error on both training and validation set.
- Test on a small subset of the data and check the error $\to 0$. 
WHAT IF IT DOES NOT WORK?

Training diverges:
- Learning rate may be too large → decrease learning rate
- BPROP is buggy → numerical gradient checking

Parameters collapse / loss is minimized but accuracy is low
- Check loss function:
  - Is it appropriate for the task you want to solve?
  - Does it have degenerate solutions? Check “pull-up” term.

Network is underperforming
- Compute flops and nr. params. → if too small, make net larger
- Visualize hidden units/params → fix optimization

Network is too slow
- Compute flops and nr. params. → GPU,distrib. framework, make net smaller
Industry Deployment

• Used in Facebook, Google, Microsoft
• Face recognition, image search, photo organization…. 
• Very fast at test time (~100 images/sec/GPU)

[Taigman et al. DeepFace: Closing the Gap to Human-Level Performance in Face Verification, CVPR’14]
Labeled Faces in Wild Dataset

- Task: given pair of images, same person or not?

![ROC curve](image)

- Human cropped (97.5%)
- DeepFace-ensemble (97.25%)
- DeepFace-single (97.00%)
- TL Joint Baysian (96.33%)
- High-dimensional LBP (95.17%)
- Tom-vs-Pete + Attribute (93.30%)
- combined Joint Baysian (92.42%)

[Tagman et al. CVPR’14]
Detection with ConvNets

- So far, all about classification

- What about localizing objects within the scene?
Two General Approaches

1. Examine very position / scale
   - E.g. Overfeat: Integrated recognition, localization and detection using convolutional networks, Sermanet et al., ICLR 2014

2. Use some kind of proposal mechanism to attend to a set of possible regions
   - E.g. Region-CNN [Rich feature hierarchies for accurate object detection and semantic segmentation, Girshick et al., CVPR 2014]
Sliding Window with ConvNet

Input Image

- Image size: 224
- Filter size: 7
- Stride: 2

Layer 1
- Conv: 110
- 3x3 max pool: 55
- Stride: 2
- Norm.

Layer 2
- Conv: 26
- 3x3 max pool: 55
- Stride: 2
- Norm.

Layer 3
- Conv: 13
- 3x3 max pool: 13
- Stride: 2
- Norm.

Layer 4
- Conv: 13
- 3x3 max pool: 13
- Stride: 2
- Norm.

Layer 5
- Full: 4096 units
- 4096 units
- Softmax

Output
Sliding Window with ConvNet

Input Window: 224x224

Feature Extractor:
- Conv 1: 110x110, 3x3 max pool, stride 2
  - Layer 1: 55x55, 3x3 conv, stride 2
  - Layer 2: 26x26, 3x3 conv, stride 2
  - Layer 3: 13x13, 3x3 conv, stride 2
  - Layer 4: 13x13, 3x3 conv, stride 2

Classifier:
- Conv 5: 13x13, 3x3 max pool, stride 2
- Full 6: 4096 units
- Full 7: 4096 units
- Class softmax

Output: C classes
Sliding Window with ConvNet

No need to compute two separate windows --- Just one big input window
Multi-Scale Sliding Window ConvNet

Feature Maps

Class Maps

Feature Extractor

Classifier

C=1000

256
Multi-Scale Sliding Window ConvNet

Feature Extractor

Feature Maps

Bounding Box Maps

Regression Network
OverFeat – Output before NMS
Overfeat Detection Results

[Sermanet et al. ICLR 2014]
**R-CNN Approach**

Rich feature hierarchies for accurate object detection and semantic segmentation, Girshick et al., CVPR 2014

- **Bottom-up proposal mechanism**

- **Scored by classifier**

- **Current best detection approach on PASCAL VOC**

  - **Current best detection approach on PASCAL VOC**

  Further work combines proposal mechanism with classification network:

**Figure 1: Object detection system overview.** Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. R-CNN achieves a mean average precision (mAP) of **53.7% on PASCAL VOC 2010**. For comparison, [34] reports 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. The popular deformable part models perform at 33.4%.
Video Classification

• Want to capture temporal structure
• 3D convolutions & 3D max-pooling
• E.g. C3D model

8 convolution, 5 pool, 2 fully-connected layers
3x3x3 convolution kernels
2x2x2 pooling kernels

[Learning Spatiotemporal Features with 3D Convolutional Networks, Tran et al., arXiv:1412.0767, 2014]

[Slide: Manohar Paluri]
Action Recognition – UCF101 dataset
## Action Recognition Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Imagenet</td>
<td>68.8</td>
</tr>
<tr>
<td>iDT</td>
<td>76.2</td>
</tr>
<tr>
<td>Deep networks [19]</td>
<td>65.4</td>
</tr>
<tr>
<td>Spatial stream network [36]</td>
<td>72.6</td>
</tr>
<tr>
<td>LRCN [7]</td>
<td>71.1</td>
</tr>
<tr>
<td>LSTM composite model [39]</td>
<td>75.8</td>
</tr>
<tr>
<td><strong>C3D</strong> (1 net)</td>
<td>82.3</td>
</tr>
<tr>
<td><strong>C3D</strong> (3 nets)</td>
<td><strong>85.2</strong></td>
</tr>
<tr>
<td>iDT with Fisher vector [31]</td>
<td>87.9</td>
</tr>
<tr>
<td>Temporal stream network [36]</td>
<td>83.7</td>
</tr>
<tr>
<td>Two-stream networks [36]</td>
<td>88.0</td>
</tr>
<tr>
<td>LRCN [7]</td>
<td>82.9</td>
</tr>
<tr>
<td>LSTM composite model [39]</td>
<td>84.3</td>
</tr>
<tr>
<td>Multi-skip feature stacking [26]</td>
<td>89.1</td>
</tr>
<tr>
<td><strong>C3D</strong> (3 nets) + iDT</td>
<td><strong>90.4</strong></td>
</tr>
</tbody>
</table>

**Use raw pixel inputs**

**Use optical flows**

[Slide: Manohar Paluri]
2D vs 3D Convnets

- UCF101 training

[Graph showing clip accuracy over # epoch for different depths]

[t-SNE visualization]

[Slide: Manohar Paluri]
Sport Classification Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of Nets</th>
<th>Clip hit@1</th>
<th>Video hit@1</th>
<th>Video hit@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep Video’s Single-Frame + Multires [19]</td>
<td>3 nets</td>
<td>42.4</td>
<td>60.0</td>
<td>78.5</td>
</tr>
<tr>
<td>Deep Video’s Slow Fusion [19]</td>
<td>1 net</td>
<td>41.9</td>
<td>60.9</td>
<td>80.2</td>
</tr>
<tr>
<td>C3D (trained from scratch)</td>
<td>1 net</td>
<td>44.9</td>
<td>60.0</td>
<td>84.4</td>
</tr>
<tr>
<td>C3D (fine-tuned from I380K pre-trained model)</td>
<td>1 net</td>
<td><strong>46.1</strong></td>
<td><strong>61.1</strong></td>
<td><strong>85.2</strong></td>
</tr>
</tbody>
</table>

[Slide: Manohar Paluri]
Dense Scene Labeling

• Classification: pixels $\rightarrow$ label
• Detection: pixels $\rightarrow$ boxes

• Use Convnets to do pixels $\rightarrow$ pixels
  – Segmentation of image
  – Image processing tasks (denoising etc.)
  – Don’t want pooling
Dense Scene Labeling

- Convnet output is per-pixel label map
Dense Scene Labeling

- ConvNet output is per-pixel depth map
• Convnet output is per-pixel normal map
Input: 320x240

[Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture, Eigen et al., arXiv 1411.4734, 2014]
[Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture, Eigen et al., arXiv 1411.4734, 2014]
Multi-Scale Convnets

Input: 320x240

[Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture, Eigen et al., arXiv 1411.4734, 2014]
Use Appropriate Loss Functions

**Depth:** \( d = D - D^* \) \( D = \log \text{predicted depth}, \ D^* = \log \text{true depth} \)

\[
L_{\text{depth}}(D, D^*) = \frac{1}{n} \sum_i d_i^2 - \frac{1}{2n^2} \left( \sum_i d_i \right)^2 + \frac{1}{n} \sum_i [(\nabla_x d_i)^2 + (\nabla_y d_i)^2]
\]

[Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture, Eigen et al., arXiv 1411.4734, 2014]
Depths Comparison

Eigen NIPS’14 (2 scales)  Ours  Ground Truth

[Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture, Eigen et al., arXiv 1411.4734, 2014]
Surface Normals

|-----------|---------|--------------------|----------------|------------|--------------|

[Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture, Eigen et al., arXiv 1411.4734, 2014]
Scene Parsing

- Farabet et al. “Learning hierarchical features for scene labeling” PAMI 2013
Segmentation

- Ciresan et al. “DNN segment neuronal membranes...” NIPS 2012
- Turaga et al. “Maximin learning of image segmentation” NIPS 2009
Denoising with ConvNets

- Burger et al. “Can plain NNs compete with BM3D?” CVPR 2012
Deblurring with Convnets

• Blind deconvolution
  – Learning to Deblur, Schuler et al., arXiv 1406.7444, 2014

| Blurry image with ground truth kernel | Result of [Zho+13] PSNR 23.17 | Deblurring result w. noise agnostic training PSNR 23.29 | Deblurring result w. noise specific training PSNR 23.41 |
Inpainting with Convnets

- Image Denoising and Inpainting with Deep Neural Networks, Xie et al. NIPS 2012.
- Mask-specific inpainting with deep neural networks, Köhler et al., Pattern Recognition 2014

Original Schmid CVPR’10 Köhler et al.
Removing Local Corruption

Restoring An Image Taken Through a Window Covered with Dirt or Rain

Rain Sequence
Each frame processed independently

David Eigen, Dilip Krishnan and Rob Fergus
ICCV 2013
Removing Local Corruption

- Restoring An Image Taken Through a Window Covered with Dirt or Rain, Eigen et al., ICCV 2013.
Convnet + Structured Learning

Convnet + Structured Learning


• Lots more recently……
BODY TRACKING

• Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation

  J. Tompson, A. Jain, Y. LeCun, C. Bregler, NIPS 2014
BODY TRACKING: PART DETECTOR

Simplified multi-resolution efficient model:
BODY TRACKING: SPATIAL MODEL

Start with MRF formulation

“Convolutional priors”

Sum-product belief propagation

\[
\bar{p}_A = \frac{1}{Z} \prod_{v \in V} \left( p_{A|v} \ast p_v + b_{v \rightarrow A} \right)
\]
BODY TRACKING: SPATIAL MODEL

Implement it as a network (no longer MRF)!

\[
\tilde{p}_A = \frac{1}{Z} \prod_{v \in V} (p_{A|v} * p_v + b_{v \rightarrow A})
\]

\[
\tilde{e}_A = \exp \left( \sum_{v \in V} \left[ \log \left( \text{SoftPlus} \left( e_{A|v} \right) \right) \cdot \text{ReLU} \left( e_v \right) + \text{SoftPlus} \left( b_{v \rightarrow A} \right) \right] \right)
\]

where: \( \text{SoftPlus} \left( x \right) = \frac{1}{\beta} \log \left( 1 + \exp \left( \beta x \right) \right), \frac{1}{2} \leq \beta \leq 2 \)

\( \text{ReLU} \left( x \right) = \max \left( x, \epsilon \right), 0 < \epsilon \leq 0.01 \)
References

• [Slide 5]

• Zheng Song*, Qiang Chen*, Zhongyang Huang, Yang Hua, and Shuicheng Yan. Contextualizing Object Detection and Classification. In CVPR'11. (* indicates equal contribution) [No. 1 performance in VOC’10 classification task]

• [Slide 6]

• [Slide 7]
  Gehler and Nowozin, On Feature Combination for Multiclass Object Classification, ICCV’09

• [Slide 8]

• [Slide 10]
  Yoshua Bengio and Yann LeCun: Scaling learning algorithms towards AI, in Bottou, L. and Chapelle, O. and DeCoste, D. and Weston, J. (Eds), Large-Scale Kernel Machines, MIT Press, 2007
References

- [Slide 11]
- S. Lazebnik, C. Schmid, and J. Ponce, Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories, CVPR 2006
- [Slide 12]
- Christoph H. Lampert, Hannes Nickisch, Stefan Harmeling: "Learning To Detect Unseen Object Classes by Between-Class Attribute Transfer", IEEE Computer Vision and Pattern Recognition (CVPR), Miami, FL, 2009
- http://www.scholarpedia.org/article/Neocognitron
References

- [Slide 31] Q.V. Le, J. Ngiam, Z. Chen, D. Chia, P. Koh, A.Y. Ng, Tiled Convolutional Neural Networks. NIPS, 2010
- http://ai.stanford.edu/~quocle/TCNNweb
- Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus, Adaptive Deconvolutional Networks for Mid and High Level Feature Learning, International Conference on Computer Vision (November 6-13, 2011)
References

• [Slide 35]
• [Slide 36]
• [Slide 39]
• [Slide 41]
References

• [Slide 42]
• [Slide 44]
• P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, Object Detection with Discriminatively Trained Part Based Models, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, No. 9, September 2010
• [Slide 45]
• Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus, Adaptive Deconvolutional Networks for Mid and High Level Feature Learning, International Conference on Computer Vision (November 6-13, 2011)
References

• [Slide 48]

• [Slide 49]
  R. Girshick, P. Felzenszwalb, D. McAllester, Object Detection with Grammar Models, NIPS 2011

• [Slide 50]

• [Slide 51]


References

• [Slide 52]
• [Slide 53]
• Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus, Adaptive Deconvolutional Networks for Mid and High Level Feature Learning, International Conference on Computer Vision (November 6-13, 2011)
• [Slide 54]
• [Slide 55]
References

• [Slide 56]
• http://www.image-net.org/challenges/LSVRC/2010/
• Q.V. Le, W.Y. Zou, S.Y. Yeung, A.Y. Ng., Learning hierarchical spatio-temporal features for action recognition with independent subspace analysis, CVPR 2011