Normalization

- Contrast normalization

The same response for different contrasts is desired.
• **Contrast normalization** (between/across feature maps)
  – Local mean = 0, local std. = 1
  – "Local" $\xrightarrow{7 \times 7 \text{Gaussian}}$
  – Equalizes the feature maps

Credit: R. Fergus
Improving Generalization: DropOut

[Hinton et al. NIPS’12]

Motivation:

• Random Forests generalize well due to averaging of many models
• Decision Trees are fast - ConvNets are slow – many models are not feasible

• Similar to random forest bagging [Breiman’94], but differs in that parameters are shared
• For fully connected layers only:
  – In training: Independently set each hidden unit activity to zero with 0.5 probability
  – In testing multiply neuron output by 0.5

• Corresponds to averaging over exponentially many samples from different model architectures
Further pre-processing tricks

- Mean removal
  Centered (0-mean) RGB values.

- Data augmentation
  Train on 224x224 patches extracted randomly from images, and also their horizontal reflections

[Chatfield et al. BMVC’14]

[Krizhevsky et al. NIPS?’12]
ImageNet Classification 2013/2014 Results


VGG Team ILSVRC Progress

- Pre-2012: 26.2% error \( \rightarrow \) 2012: 16.5% error \( \rightarrow \) 2013: 11.2% error

Credit: R. Fergus
ImageNet Sample classifications

[Krizhevsky et al. NIPS’12]
ImageNet Sample classifications

[Krizhevsky et al. NIPS’12]
## ImageNet Classification Progress from ‘12-’14

<table>
<thead>
<tr>
<th>2012 Teams</th>
<th>%error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervision (Toronto)</td>
<td>15.3</td>
</tr>
<tr>
<td>ISI (Tokyo)</td>
<td>26.1</td>
</tr>
<tr>
<td>VGG (Oxford)</td>
<td>26.9</td>
</tr>
<tr>
<td>XRCE/INRIA</td>
<td>27.0</td>
</tr>
<tr>
<td>UvA (Amsterdam)</td>
<td>29.6</td>
</tr>
<tr>
<td>INRIA/LEAR</td>
<td>33.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2013 Teams</th>
<th>%error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clarifai (NYU spinoff)</td>
<td>11.7</td>
</tr>
<tr>
<td>NUS (Singapore)</td>
<td>12.9</td>
</tr>
<tr>
<td>Zeiler-Fergus (NYU)</td>
<td>13.5</td>
</tr>
<tr>
<td>A. Howard</td>
<td>13.5</td>
</tr>
<tr>
<td>OverFeat (NYU)</td>
<td>14.1</td>
</tr>
<tr>
<td>UvA (Amsterdam)</td>
<td>14.2</td>
</tr>
<tr>
<td>Adobe</td>
<td>15.2</td>
</tr>
<tr>
<td>VGG (Oxford)</td>
<td>15.2</td>
</tr>
<tr>
<td>VGG (Oxford)</td>
<td>23.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2014 Teams</th>
<th>%error</th>
</tr>
</thead>
<tbody>
<tr>
<td>GoogLeNet</td>
<td>6.6</td>
</tr>
<tr>
<td>VGG (Oxford)</td>
<td>7.3</td>
</tr>
<tr>
<td>MSRA</td>
<td>8.0</td>
</tr>
<tr>
<td>A. Howard</td>
<td>8.1</td>
</tr>
<tr>
<td>DeeperVision</td>
<td>9.5</td>
</tr>
<tr>
<td>NUS-BST</td>
<td>9.7</td>
</tr>
<tr>
<td>TTIC-ECP</td>
<td>10.2</td>
</tr>
<tr>
<td>XYZ</td>
<td>11.2</td>
</tr>
<tr>
<td>UvA</td>
<td>12.1</td>
</tr>
</tbody>
</table>

Credit: Y. LeCun
Deeper is Better

• Each weight layer performs a linear operation, followed by non-linearity
  – a single layer can be seen as a linear classifier itself
• More layers – more non-linearities
  – leads to a more discriminative model
• What limits the number of layers?
  – many models use pooling after each conv. layer
    • input image resolution sets the limit: \( \log(s) \) for \( s \times s \) input
  – computational complexity
Better (≈deeper) architectures exist now

ILSVRC14 Winners: ~7.3% Top-5 error
- VGG: 16 layers of stacked 3x3 convolution with stride 1

Extra layers injected into deeper stacks
- first layers capture lower-level primitives, don’t need to be very discriminative
- spatial resolution is higher in the first layers, adding extra layers there is computationally prohibitive
Better (≈deeper) architectures exist now

ILSVRC14 Winners: \(\sim 7.3\%\) Top-5 errors

- VGG: 16 layers of stacked 3x3 convolution with stride 1

Other details:
- Rectification (ReLU) non-linearity
- 5 max-pool layers (x2 reduction)
- no normalisation
- 3 fully-connected (FC) layers
Better (≈deeper) architectures exist now

ILSVRC14 Winners: \(~6.6\% \text{ Top-5 error}\)
- GoogLeNet: composition of multi-scale dimension-reduced modules:
  - 1x1 convolutions serve as dimensionality reduction

Zeiler-Fergus Architecture (1 tower)

Credit: C. Szegedy
In images, correlations tend to be local
Cover very local clusters by 1x1 convolutions
Less spread out correlations

number of filters

1x1
Cover more spread out clusters by 3x3 convolutions
Cover more spread out clusters by 5x5 convolutions
Cover more spread out clusters by 5x5 convolutions

Credit: C. Szegedy
A heterogeneous set of convolutions

Credit: C. Szegedy
Schematic view (naive version)

number of filters

1x1

3x3

5x5

Credit: C. Szegedy
Naive idea

1x1 convolutions

3x3 convolutions

5x5 convolutions

Filter concatenation

Previous layer

Credit: C. Szegedy
Naive idea (**does not work!**)
- Too many parameters

Credit: C. Szegedy
• **1x1 Convolution**
  – Only operates across channels
  – Performs a linear projections of a single pixel’s features
  – Can be used to reduce dimensionality
  – Increases depth
Inception

9 Inception modules

Network in a network in a network...

Credit: C. Szegedy
Classification failure cases

**Groundtruth:** coffee mug

**GoogLeNet:**
- table lamp
- lamp shade
- printer
- projector
- desktop computer

Credit: C. Szegedy
Classification failure cases

Groundtruth: hay

GoogLeNet:
- sorrel (horse)
- hartebeest
- Arabian camel
- warthog
- gazelle

Credit: C. Szegedy
Classification failure cases

**Groundtruth**: Police car

**GoogLeNet**:
- laptop
- hair drier
- binocular
- ATM machine
- seat belt
Parametric ReLU

- Activation function:
  \[ f(y_i) = \begin{cases} 
  y_i, & \text{if } y_i > 0 \\
  a_i y_i, & \text{if } y_i \leq 0 
\end{cases} \]

- \(a_i\) is learnable with back-prop
  - per-channel or per-layer
  - learnable activation function!

- Generalises
  - ReLU (\(a_i=0\))
  - leaky ReLU (\(a_i=0.01\))

- 0.5%/0.2% top-1/top-5 error reduction

[He et al., 2015]
Batch Normalisation

• The distribution of activations changes during training, making training harder
  ○ Eliminating these changes speeds up training

• Whitening of neural net inputs is a standard pre-processing technique

• Batch normalisation [Ioffe & Szegedy, 2015] performs normalisation of outputs of each layer to zero mean and unit variance
  – can be seen as diagonal whitening
  – performed after each weight layer before ReLU

Credit: K. Simonyan
Batch Normalisation (2)

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

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

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

- scale and shift parameters are learnt
- doing backprop through batchnorm is important
- nets with batchnorm need less regularisation
  - smaller/zero dropout & weight decay

Credit: K. Simonyan
Better (≈deeper) architectures exist now

ILSVRC15 Winners: \(~3.57\%\ \textbf{Top-5 error}\)

\textbf{Revolution of Depth}

\begin{itemize}
  \item ILSVRC'15 ResNet: 3.57\%
  \item ILSVRC'14 GoogleNet: 6.7\%
  \item ILSVRC'14 VGG: 7.3\%
  \item ILSVRC'13: 8 layers, 11.7\%
  \item ILSVRC'12 AlexNet: 8 layers, 16.4\%
  \item ILSVRC'11: shallow, 25.8\%
  \item ILSVRC'10: 28.2\%
\end{itemize}

ImageNet Classification top-5 error (%)
Better (≈deeper) architectures exist now

AlexNet, 8 layers (ILSVRC 2012)
- 11x11 conv, 96, /4, pool/2
- 5x5 conv, 256, pool/2
- 3x3 conv, 384
- 3x3 conv, 384
- 3x3 conv, 256, pool/2
- fc, 4096
- fc, 4096
- fc, 1000

VGG, 19 layers (ILSVRC 2014)
- 3x3 conv, 64
- 3x3 conv, 64, pool/2
- 3x3 conv, 128
- 3x3 conv, 128, pool/2
- 3x3 conv, 256
- 3x3 conv, 256
- 3x3 conv, 256
- 3x3 conv, 256, pool/2
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512, pool/2
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- fc, 4096
- fc, 4096
- fc, 1000

ResNet, 152 layers (ILSVRC 2015)
- “Ultra deep”
- all 3x3 conv filters (except first: 7x7)
- spatial size /2 → # filters x2
- Uses Batch Normalisation during training
- Standard hyperparameters & data augmentation
- No hidden fc layer
- No max pooling (except after first layer)
- No dropout

But: Simply stacking layers is not enough -→ residual connections
Why simply stacking layers does not work

- “Overly deep” plain nets have **higher training error**
- A general phenomenon, observed in many datasets
  - Network performs multiplicative transformations on feature maps
  - Simply stacking layers leads to bad gradient properties (vanishing/exploding)
Why simply stacking layers does not work

- A deeper model should not have higher training error
- A solution by construction:
  - original layers: copied from a learned shallower model
  - extra layers: set as identity
  - at least the same training error
- Optimization difficulties: solvers cannot find the solution when going deeper...

Credit: K. He
Solution: Residual learning

- Plain net
  \[ x \rightarrow \text{weight layer} \rightarrow \text{relu} \rightarrow \text{weight layer} \rightarrow \text{relu} \rightarrow H(x) \]

- Residual net
  \[ F(x) \rightarrow \text{weight layer} \rightarrow \text{relu} \rightarrow \text{weight layer} \]
  \[ H(x) = F(x) + x \rightarrow \text{relu} \]

- \( F(x) \) is a residual mapping w.r.t. identity

- If identity were optimal, easy to set weights as 0

- If optimal mapping is closer to identity, easier to find small fluctuations

- Deep ResNets can be trained without difficulties
- Deeper ResNets have lower training error, and also lower test error

Feature Generalization
ImageNet pre-training is general for various tasks.
ImageNet pre-training

- **Labeled data is rare for detection**: leverage large classification labeled datasets for pre-training.
- **ImageNet Classification pretraining + fine-tuning on a different task** has been shown to work very well
  - e.g. [Razavian’14] took the off-the-shelf ConvNet OverFeat + SVM classifier on top and obtained many state-of-the-art or competitive results on 10+ datasets and visual tasks

---


Credit: P. Sermanet
Classifier re-training on Caltech 256

State of the art accuracy with only 6 training samples/class

Credit: R. Fergus

Zeiler & Fergus, *Visualizing and Understanding Convolutional Networks*, ECCV’14
CNNs have set a new state-of-the-art for many tasks

- classification
- localization
- detection
- segmentation

Credit: P. Sermanet
Object Detection

Example detection results of Faster R-CNN


Credit: K. He
Better (≈deeper) architectures exist now

ILSVRC15 Winners: **Object detection revolution**

**Revolution of Depth**

Engines of visual recognition

PASCAL VOC 2007 **Object Detection** mAP (%)
Object Detection (In Brief)

• Common approach:
  – generate a large number of bounding box proposals
  – classify them using visual features

• ConvNet features work very well!
  – R-CNN [Girshick et al., 2013]

• Fast R-CNN [Girshick et al., 2015]
  – for each proposal, predicts its class and precise bbox location
  – re-uses conv. features, no need to re-compute

• Proposals
  – Selective search
  – Multi-Box
  – Faster R-CNN

Credit: K. Simonyan
Object Detection code online

- **R-CNN**
  (Caffe + MATLAB): https://github.com/rbgirshick/rcnn

- **Fast R-CNN**
  (Caffe + MATLAB): https://github.com/rbgirshick/fast-rcnn
  (MatConvNet + MATLAB):
  https://github.com/vlfeat/matconvnet/tree/master/examples/fast_rcnn

- **Faster R-CNN**
  (Caffe + MATLAB): https://github.com/ShaoqingRen/faster_rcnn
  (Caffe + Python): https://github.com/rbgirshick/py-faster-rcnn
  (TensorFlow + Python) https://github.com/smallcorgi/Faster-RCNN_TF

- **Faster R-CNN for Pedestrian Detection**
  (Caffe + MATLAB): https://github.com/zhangliliang/RPN_BF/tree/RPN-pedestrian
Feature sharing via transfer learning

- Pre-training allows to use big models for small datasets
  - For example: Pre-Train model on large ImageNet 2012 training set
    - Train CNN
    - Auxiliary task: ILSVRC 2012 classification (1.2 million images)
    - Fine-tune CNN
    - Target task: PASCAL VOC detection (~25k object labels)

- Re-train on new dataset (fine tuning or transfer learning)
  - Either: Just the classifier-layer or the whole network
    - For fine-tuning pre-trained layers, the learning rate has to be lowered to avoid unlearning the pre-trained weights
    - For fine tuning new layers (e.g. the new classifier layer) the learning rate has to be higher
    - Better: Two stage fine-tuning
      - Stage 1: First only learn new layers with the learning rate of pre-trained layers set to zero
      - Stage 2: Use default learning rate to fine-tune everything (optimize all parameters jointly)

- Classify test set of new dataset

[Girshick et al. CVPR’14]
(Fine tuned) CNNs for detection on the Pascal dataset

- Combines bottom-up region proposals with rich features computed by a CNN

**R-CNN: Regions with CNN features**

1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

- Previous state-of-the-art: 35.1% mean average precision
- scratch: Training on Pascal train+val data
- pre-train: Pre-training on ImageNet and just the classifier is trained on Pascal
- fine-tune: Two stage fine-tuning on Pascal

![ImageNet classification results](chart.png)
R-CNN

- R-CNN is slow because it needs to apply the CNN to each region proposal.
Exploits translational equivariance of convolutional layers to process many proposals for an image. Is trained end-to-end with a box-classification loss (log) and a bounding box offset loss ($L_1$).
Fast R-CNN: Efficient RoI sampling

Slow R-CNN and SPP-net use region-wise sampling to make mini-batches

- Sample 128 example RoIs uniformly at random
- Examples will come from different images with high probability

Fast R-CNN: Efficient RoI sampling

Solution: use hierarchical sampling to build mini-batches

- Sample a small number of images (2)
- Sample many examples from each image (64)

Fast R-CNN: Efficient RoI pooling

Use the fully convolutional trick during training

- Share computation between overlapping examples from the same image

Are the examples from just 2 images diverse enough?

- Concern: examples from the sample image may be too correlated
  - In practice this fact is ignored
Review: Region of Interest (RoI) pooling layer

Just a special case of the SPP layer with one pyramid level

Credit: K. He
Hierarchical Sampling

1. Sample a small number of images, **Randomly**

2. Sample many examples from each image

Fast R-CNN training with SGD

Hierarchical Sampling

1. Sample a small number of images, **Randomly**

2. Sample many examples from each image
Fast R-CNN training with SGD

Fast R-CNN training with SGD

RoI Network Forward Pass

Sample Image

ConvNet

Sample RoIs

> 10000

2

> 4000

128

ConvNet Forward Pass

Rol Network Forward Pass

Fast R-CNN training with SGD

Faster R-CNN

- Faster R-CNN = Fast R-CNN + Region Proposal Networks
  - Does not depend on an external region proposal algorithm
  - Does object detection in a single forward pass

Credit: K. He
Region Proposal from Feature Maps

• Object detection networks are fast (0.2s)...
• but what about region proposal?
  • Selective Search [Uijlings et al. ICCV 2011]: 2s per image
  • EdgeBoxes [Zitnick & Dollar. ECCV 2014]: 0.2s per image

• Can we do region proposal on the same set of feature maps?

Feature Maps = features and their locations

Convolutional: sliding-window operations

Feature:
encoding “what”
(and implicitly encoding “where”)

Map:
explicitly encoding “where”
Region Proposal from Feature Maps

- By decoding **one response** at a single pixel, we can still roughly see the object outline*
- Finer localization information has been encoded in the channels of a convolutional feature response
- Extract this information for better localization...

* Zeiler & Fergus’s method traces unpooling information so the visualization involves more than a single response. But other visualization methods reveal similar patterns.

Region Proposal from Feature Maps

- a feature vector (e.g., 256-d)

- The channels of this feature vector encodes finer localization information

- The spatial position of this feature provides coarse locations


Credit: K. He
RPN: Multiple Anchors

3x3 “sliding window”
- $K$ objectness classifiers
- $K$ box regressors

Anchor boxes: $K$ anchors per location with different scales and aspect ratios
Region Proposal Network

- Slide a small window on the feature map
- Build a small network for:
  - classifying object or not-object, and
  - regressing bbox locations
- Position of the sliding window provides localization information with reference to the image
- Box regression provides finer localization information with reference to this sliding window

Credit: K. He
Anchors as references

**Anchors**: pre-defined reference boxes
- Box regression is with reference to anchors: regressing an anchor box to a ground-truth box

- Object probability is with reference to anchors, e.g.:
  - anchors as positive samples: if IoU > 0.7 or IoU is max
  - anchors as negative samples: if IoU < 0.3


Credit: K. He
Anchors as references

- **Anchors**: pre-defined reference boxes

- **Translation-invariant** anchors:
  - the same set of anchors are used at each sliding position
  - the same prediction functions (with reference to the sliding window) are used
  - a translated object will have a translated prediction


Credit: K. He
Anchors as references

- **Anchors**: pre-defined reference boxes

- **Multi-scale/size** anchors:
  - multiple anchors are used at each position:
    - e.g., 3 scales (128², 256², 512²) and 3 aspect ratios
      (2:1, 1:1, 1:2) yield 9 anchors
  - each anchor has its own prediction function
  - **single-scale** features, multi-scale predictions


Credit: K. He
Anchors as references

- Comparisons of multi-scale strategies


Credit: K. He
Region Proposal Network

- RPN is **fully convolutional** [Long et al. 2015]
- RPN is trained end-to-end
- RPN shares convolutional feature maps with the detection network (covered in Ross’s section)


Credit: K. He
Faster R-CNN

<table>
<thead>
<tr>
<th>System</th>
<th>Time</th>
<th>07 Data</th>
<th>07+12 Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-CNN</td>
<td>~50s</td>
<td>66.0</td>
<td>-</td>
</tr>
<tr>
<td>Fast R-CNN</td>
<td>~2s</td>
<td>66.9</td>
<td>70.0</td>
</tr>
<tr>
<td>Faster R-CNN</td>
<td>198ms</td>
<td>69.9</td>
<td>73.2</td>
</tr>
</tbody>
</table>

detection mAP on PASCAL VOC 2007, with VGG-16 pre-trained on ImageNet

Training goal: Share features

CNN A + RPN
Region Proposal Network
proposals

Goal: share so
CNN A == CNN B

CNN B + detector

CNN B

proposals
from any algorithm

feature map

classifier
RoI pooling

Credit: R. Girshick
Faster R-CNN: Joint training

- Classification loss
- Bounding-box regression loss

Region Proposal Network

Proposal features map

RoI pooling

CNN

Classification loss

Bounding-box regression loss

Credit: R. Girshick
Faster R-CNN + ResNet

Examples
Faster R-CNN + ResNet Examples
Faster R-CNN + ResNet
Examples