Object Classification

Lecture 4
Deep Residual Networks
Deep Learning Gets Way Deeper

8:30-10:30am, June 19
ICML 2016 tutorial

Kaiming He
Facebook AI Research*

*as of July 2016. Formerly affiliated with Microsoft Research Asia
Introduction
Introduction

Deep Residual Networks (ResNets)

• “Deep Residual Learning for Image Recognition”. CVPR 2016 (next week)

• A simple and clean framework of training “very” deep nets

• State-of-the-art performance for
  • Image classification
  • Object detection
  • Semantic segmentation
  • and more...

ResNets @ ILSVRC & COCO 2015 Competitions

• **1st places in all five main tracks**
  • ImageNet Classification: “Ultra-deep” 152-layer nets
  • ImageNet Detection: 16% better than 2nd
  • ImageNet Localization: 27% better than 2nd
  • COCO Detection: 11% better than 2nd
  • COCO Segmentation: 12% better than 2nd

*improvements are relative numbers

ImageNet Classification top-5 error (%)
Revolution of Depth

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

Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

GoogleNet, 22 layers
(ILSVRC 2014)

Revolution of Depth

AlexNet, 8 layers (ILSVRC 2012)  VGG, 19 layers (ILSVRC 2014)  ResNet, 152 layers (ILSVRC 2015)

Revolution of Depth

ResNet, 152 layers

Revolution of Depth

Engines of visual recognition

HOG, DPM 34 
AlexNet (RCNN) 58 8 layers 
VGG (RCNN) 66 16 layers 
ResNet (Faster RCNN)* 86 101 layers

PASCAL VOC 2007 Object Detection mAP (%)

ResNet’s object detection result on COCO

*the original image is from the COCO dataset

Very simple, easy to follow

- Many third-party implementations (list in https://github.com/KaimingHe/deep-residual-networks)
  - Facebook AI Research’s Torch ResNet:
  - Torch, CIFAR-10, with ResNet-20 to ResNet-110, training code, and curves: code
  - Lasagne, CIFAR-10, with ResNet-32 and ResNet-56 and training code: code
  - Neon, CIFAR-10, with pre-trained ResNet-32 to ResNet-110 models, training code, and curves: code
  - Torch, MNIST, 100 layers: blog, code
  - A winning entry in Kaggle’s right whale recognition challenge: blog, code
  - Neon, Place2 (mini), 40 layers: blog, code
  - …

- Easily reproduced results (e.g. Torch ResNet: https://github.com/facebook/fb.resnet.torch)

- A series of extensions and follow-ups
  - > 200 citations in 6 months after posted on arXiv (Dec. 2015)

Background

From shallow to deep
Traditional recognition

But what’s next?

Shallower

Deeper

Traditional recognition:
- Pixels → Classifier → “bus”?  
- Edges → Classifier → “bus”?  
- Edges → Histogram → Classifier → “bus”?  
- Edges → Histogram → K-means/Sparse Code → Classifier → “bus”?  

Diagram:
- Traditional recognition processes images by first extracting edges and then using a classifier to determine if an object is a bus.  
- More advanced methods include using SIFT/HOG features, which can be used in shallower or deeper classification models.
Deep Learning

Specialized components, domain knowledge required

Generic components ("layers"), less domain knowledge

Repeat elementary layers => Going deeper

• End-to-end learning
• Richer solution space
Spectrum of Depth

5 layers: easy

>10 layers: initialization, Batch Normalization

>30 layers: skip connections

>100 layers: identity skip connections

>1000 layers: ?

shallow

deep
Initialization

If:
• Linear activation
• $x, y, w$: independent
Then:

1-layer:
$$Var[y] = (n^{in} Var[w]) Var[x]$$

Multi-layer:
$$Var[y] = \left( \prod_d n^{in}_d Var[w_d] \right) Var[x]$$

LeCun et al 1998 “Efficient Backprop”
Glorot & Bengio 2010 “Understanding the difficulty of training deep feedforward neural networks”
Initialization

Both forward (response) and backward (gradient) signal can vanish/explode

Forward:

$$Var[y] = \prod_d n_d^{in} Var[w_d])Var[x]$$

Backward:

$$Var\left[ \frac{\partial}{\partial x} \right] = \prod_d n_d^{out} Var[w_d])Var[\frac{\partial}{\partial y}]$$

LeCun et al 1998 “Efficient Backprop”
Glorot & Bengio 2010 “Understanding the difficulty of training deep feedforward neural networks”
Initialization

- Initialization under **linear** assumption

\[ \prod_d n_d^{in} Var[w_d] = const_{fw} \text{ (healthy forward)} \]
and
\[ \prod_d n_d^{out} Var[w_d] = const_{bw} \text{ (healthy backward)} \]

\[ n_d^{in} Var[w_d] = 1 \]

or*

\[ n_d^{out} Var[w_d] = 1 \]

*: \[ n_d^{out} = n_d^{in} \], so \[ \frac{const_{bw}}{const_{fw}} = \frac{n_{out}}{n_{in}} \frac{n_{last}}{n_{first}} < \infty \].

It is sufficient to use either form.

"Xavier" init in Caffe

LeCun et al 1998 “Efficient Backprop”
Glorot & Bengio 2010 “Understanding the difficulty of training deep feedforward neural networks”
Initialization

• Initialization under ReLU

\[ \prod_d \frac{1}{2} n^\text{in}_d \text{Var}[w_d] = const_{\text{fw}} \text{ (healthy forward)} \]

and

\[ \prod_d \frac{1}{2} n^\text{out}_d \text{Var}[w_d] = const_{\text{bw}} \text{ (healthy backward)} \]

With \( D \) layers, a factor of 2 per layer has exponential impact of \( 2^D \)

\[ \frac{1}{2} n^\text{in}_d \text{Var}[w_d] = 1 \]

or

\[ \frac{1}{2} n^\text{out}_d \text{Var}[w_d] = 1 \]

“MSRA” init in Caffe

Initialization

22-layer ReLU net:
good init converges faster

30-layer ReLU net:
good init is able to converge

*Figures show the beginning of training

Batch Normalization (BN)

• Normalizing input (LeCun et al 1998 “Efficient Backprop”)

• BN: normalizing each layer, for each mini-batch

• Greatly accelerate training

• Less sensitive to initialization

• Improve regularization

Batch Normalization (BN)

\[
layer \rightarrow x \rightarrow \hat{x} = \frac{x - \mu}{\sigma} \rightarrow y = \gamma \hat{x} + \beta
\]

- $\mu$: mean of $x$ in mini-batch
- $\sigma$: std of $x$ in mini-batch
- $\gamma$: scale
- $\beta$: shift

- $\mu$, $\sigma$: functions of $x$, analogous to responses
- $\gamma$, $\beta$: parameters to be learned, analogous to weights

Deep Residual Networks

From 10 layers to 100 layers
Going Deeper

• Initialization algorithms ✓
• Batch Normalization ✓

• Is learning better networks as simple as stacking more layers?

 Simply stacking layers?

- *Plain* nets: stacking 3x3 conv layers...
- 56-layer net has **higher training error** and test error than 20-layer net

• “Overly deep” plain nets have higher training error
• A general phenomenon, observed in many datasets

A shallower model (18 layers) vs a deeper counterpart (34 layers)

- Richer solution space
- 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...

Deep Residual Learning

• Plain net

\[ H(x) \text{ is any desired mapping, hope the 2 weight layers fit } H(x) \]

Deep Residual Learning

- Residual net

\[ H(x) = F(x) + x \]

Where:

- \( H(x) \) is any desired mapping,
- hope the 2 weight layers fit \( H(x) \)
- hope the 2 weight layers fit \( F(x) \)
- let \( H(x) = F(x) + x \)

Deep Residual Learning

• $F(x)$ is a residual mapping w.r.t. identity

\[ H(x) = F(x) + x \]

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

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

Related Works – Residual Representations

• **VLAD & Fisher Vector** [Jegou et al 2010], [Perronnin et al 2007]
  • Encoding residual vectors; powerful shallower representations.

• **Product Quantization (IVF-ADC)** [Jegou et al 2011]
  • Quantizing residual vectors; efficient nearest-neighbor search.

• **MultiGrid & Hierarchical Precondition** [Briggs, et al 2000], [Szeliski 1990, 2006]
  • Solving residual sub-problems; efficient PDE solvers.

Network “Design”

- Keep it simple

- Our basic design (VGG-style)
  - all 3x3 conv (almost)
  - spatial size /2 => # filters x2 (~same complexity per layer)
  - Simple design; just deep!

- Other remarks:
  - no hidden fc
  - no dropout

Training

• All plain/residual nets are trained from scratch

• All plain/residual nets use Batch Normalization

• Standard hyper-parameters & augmentation

**CIFAR-10 experiments**

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

---

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

ImageNet experiments

• A practical design of going deeper

ImageNet experiments

- Deeper ResNets have lower error

<table>
<thead>
<tr>
<th>Model</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-152</td>
<td>5.7</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>6.1</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>6.7</td>
</tr>
<tr>
<td>ResNet-34</td>
<td>7.4</td>
</tr>
</tbody>
</table>

This model has lower time complexity than VGG-16/19

10-crop testing, top-5 val error (%)

ImageNet experiments

ImageNet Classification top-5 error (%)

- **ILSVRC'15**
  - ResNet
  - 3.57%
- **ILSVRC'14**
  - GoogleNet
  - 6.7%
  - VGG
  - 7.3%
- **ILSVRC'13**
  - 11.7%
- **ILSVRC'12**
  - AlexNet
  - 16.4%
- **ILSVRC'11**
  - Shallow
  - 25.8%
- **ILSVRC'10**
  - 28.2%

Discussions
Representation, Optimization, Generalization
### Issues on learning deep models

| **Representation ability** | • Ability of model to fit training data, if optimum could be found  
  • If model A’s solution space is a superset of B’s, A should be better. |
|---------------------------|---------------------------------------------------------------------|
| **Optimization ability**  | • Feasibility of finding an optimum  
  • Not all models are equally easy to optimize |
| **Generalization ability**| • Once training data is fit, how good is the test performance |
How do ResNets address these issues?

- **Representation ability**
  - No explicit advantage on representation (only re-parameterization), but
  - Allow models to go deeper

- **Optimization ability**
  - Enable very smooth forward/backward prop
  - Greatly ease optimizing deeper models

- **Generalization ability**
  - Not explicitly address generalization, but
  - Deeper+thinner is good generalization
On the Importance of Identity Mapping

From 100 layers to 1000 layers
On identity mappings for optimization

- shortcut mapping: $h = \text{identity}$
- after-add mapping: $f = \text{ReLU}$

$$x_{l+1} = f(h(x_l) + F(x_l))$$
On identity mappings for optimization

- shortcut mapping: $h = \text{identity}$
- after add mapping: $f = \text{ReLU}$
- What if $f = \text{identity}$?

$$x_{l+1} = f(h(x_l) + F(x_l))$$
On identity mappings for optimization

- shortcut mapping: $h = \text{identity}$
- after add mapping: $f = \text{ReLU}$
- What if $f = \text{identity}$?

$F(x_l) \xrightarrow{\text{layer}} h(x_l) \xrightarrow{\text{layer}} x_{l+1} = f(h(x_l) + F(x_l))$
Very smooth forward propagation

\[ x_{l+1} = x_l + F(x_l) \]

\[ x_{l+2} = x_{l+1} + F(x_{l+1}) \]

Very smooth forward propagation

\[ x_{l+1} = x_l + F(x_l) \]

\[ x_{l+2} = x_{l+1} + F(x_{l+1}) \]

\[ x_{l+2} = x_l + F(x_l) + F(x_{l+1}) \]

Very smooth forward propagation

\[ x_{l+1} = x_l + F(x_l) \]

\[ x_{l+2} = x_{l+1} + F(x_{l+1}) \]

\[ x_{l+2} = x_l + F(x_l) + F(x_{l+1}) \]

\[ x_L = x_l + \sum_{i=l}^{L-1} F(x_i) \]
Very smooth forward propagation

\[ x_L = x_l + \sum_{i=l}^{L-1} F(x_i) \]

- Any \( x_l \) is directly forward-prop to any \( x_L \), plus residual.
- Any \( x_L \) is an additive outcome.
  - in contrast to multiplicative: \( x_L = \prod_{i=l}^{L-1} W_i x_l \)

Very smooth backward propagation

\[ x_L = x_l + \sum_{i=l}^{L-1} F(x_i) \]

\[ \frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L} \frac{\partial x_L}{\partial x_l} = \frac{\partial E}{\partial x_L} (1 + \frac{\partial}{\partial x_l} \sum_{i=1}^{L-1} F(x_i)) \]

Very smooth backward propagation

\[ \frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L} (1 + \frac{\partial}{\partial x_l} \sum_{i=1}^{L-1} F(x_i)) \]

- Any \( \frac{\partial E}{\partial x_L} \) is directly back-prop to any \( \frac{\partial E}{\partial x_l} \), plus residual.
- Any \( \frac{\partial E}{\partial x_l} \) is additive; unlikely to vanish
  - in contrast to multiplicative: \( \frac{\partial E}{\partial x_l} = \prod_{i=l}^{L-1} W_i \frac{\partial E}{\partial x_L} \)

Residual for every layer

forward: \( x_L = x_l + \sum_{i=l}^{L-1} F(x_i) \)

backward: \( \frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L} (1 + \frac{\partial}{\partial x_l} \sum_{i=1}^{L-1} F(x_i)) \)

Enabled by:
- shortcut mapping: \( h = \text{id} \)
- after-add mapping: \( f = \text{id} \)

Experiments

• Set 1: what if shortcut mapping $h \neq \text{identity}$

• Set 2: what if after-add mapping $f$ is identity

• Experiments on ResNets with more than 100 layers
  • deeper models suffer more from optimization difficulty
Experiment Set 1:  
what if shortcut mapping $h \neq \text{identity}$?
\[ h(x) = x \]
error: 6.6%

(a) original

*ResNet-110 on CIFAR-10


\[ h(x) = x \]
error: 6.6%

\[ h(x) = 0.5x \]
error: 12.4%

\[ h(x) = \text{gate} \cdot x \]
error: 8.7%

\[ h(x) = \text{conv}(x) \]
error: 12.2%

\[ h(x) = \text{dropout}(x) \]
error: > 20%
If $h$ is multiplicative, e.g. $h(x) = \lambda x$

Forward: $x_L = \lambda^{L-1} x_l + \sum_{i=l}^{L-1} \hat{F}(x_i)$

- if $h$ is multiplicative, shortcuts are blocked
- direct propagation is decayed

Backward: $\frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L} (\lambda^{L-1} + \sum_{i=1}^{L-1} \frac{\partial}{\partial x_l} \hat{F}(x_i))$

*assuming $f = \text{identity}*

- gating should have better representation ability (identity is a special case), but
- optimization difficulty dominates results

Experiment Set 2: what if after-add mapping $f$ is identity
$f$ is ReLU (original ResNet)

$f$ is BN+ReLU

$f$ is identity (pre-activation ResNet)

\[ f = \text{ReLU} \]

- BN could also block prop
- Keep the shortest pass as smooth as possible

1001-layer ResNets on CIFAR-10

- $f = \text{ReLU}$
- $f = \text{identity}$

- ReLU could also block prop when there are 1000 layers
- pre-activation design eases optimization (and improves generalization; see paper)

### Comparisons on CIFAR-10/100

#### CIFAR-10

<table>
<thead>
<tr>
<th>method</th>
<th>error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIN</td>
<td>8.81</td>
</tr>
<tr>
<td>DSN</td>
<td>8.22</td>
</tr>
<tr>
<td>FitNet</td>
<td>8.39</td>
</tr>
<tr>
<td>Highway</td>
<td>7.72</td>
</tr>
<tr>
<td>ResNet-110 (1.7M)</td>
<td>6.61</td>
</tr>
<tr>
<td>ResNet-1202 (19.4M)</td>
<td>7.93</td>
</tr>
<tr>
<td>ResNet-164, pre-activation (1.7M)</td>
<td>5.46</td>
</tr>
<tr>
<td><strong>ResNet-1001, pre-activation (10.2M)</strong></td>
<td><strong>4.92 (4.89±0.14)</strong></td>
</tr>
</tbody>
</table>

#### CIFAR-100

<table>
<thead>
<tr>
<th>method</th>
<th>error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIN</td>
<td>35.68</td>
</tr>
<tr>
<td>DSN</td>
<td>34.57</td>
</tr>
<tr>
<td>FitNet</td>
<td>35.04</td>
</tr>
<tr>
<td>Highway</td>
<td>32.39</td>
</tr>
<tr>
<td>ResNet-164 (1.7M)</td>
<td>25.16</td>
</tr>
<tr>
<td>ResNet-1001 (10.2M)</td>
<td>27.82</td>
</tr>
<tr>
<td>ResNet-164, pre-activation (1.7M)</td>
<td>24.33</td>
</tr>
<tr>
<td><strong>ResNet-1001, pre-activation (10.2M)</strong></td>
<td><strong>22.71 (22.68±0.22)</strong></td>
</tr>
</tbody>
</table>

*all based on moderate augmentation

## ImageNet Experiments

<table>
<thead>
<tr>
<th>method</th>
<th>data augmentation</th>
<th>top-1 error (%)</th>
<th>top-5 error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-152, original</td>
<td>scale</td>
<td>21.3</td>
<td>5.5</td>
</tr>
<tr>
<td>ResNet-152, pre-activation</td>
<td>scale</td>
<td>21.1</td>
<td>5.5</td>
</tr>
<tr>
<td>ResNet-200, original</td>
<td>scale</td>
<td>21.8</td>
<td>6.0</td>
</tr>
<tr>
<td><strong>ResNet-200, pre-activation</strong></td>
<td>scale</td>
<td><strong>20.7</strong></td>
<td><strong>5.3</strong></td>
</tr>
<tr>
<td>ResNet-200, pre-activation</td>
<td>scale + aspect ratio</td>
<td><strong>20.1</strong>*</td>
<td><strong>4.8</strong>*</td>
</tr>
</tbody>
</table>

*independently reproduced by: [https://github.com/facebook/fb.resnet.torch/tree/master/pretrained#notes](https://github.com/facebook/fb.resnet.torch/tree/master/pretrained#notes)

*training code and models available.*

Summary of observations

• Keep the shortest path as smooth as possible
  • by making $h$ and $f$ identity
  • forward/backward signals directly flow through this path

• Features of any layers are additive outcomes

• 1000-layer ResNets can be easily trained and have better accuracy

Future Works

• **Representation**
  • 1-layer block vs. multi-layer block?
  • Flat vs. Bottleneck?
  • Inception-ResNet [Szegedy et al 2016]
  • ResNet in ResNet [Targ et al 2016]
  • Width vs. Depth [Zagoruyko & Komodakis 2016]

• **Generalization**
  • DropOut, MaxOut, DropConnect, ...
  • Drop Layer (Stochastic Depth) [Huang et al 2016]

• **Optimization**
  • Without residual?

More Visual Recognition Tasks

ResNet-based methods lead on these benchmarks *(incomplete list)*:  

- ImageNet classification, detection, localization  
- MS COCO detection, segmentation  
- PASCAL VOC detection, segmentation  
- MPII Human pose estimation [Newell et al 2016]  
- Depth estimation [Laina et al 2016]  
- Segment proposal [Pinheiro et al 2016]  
- …
Potential Applications

ResNets have shown outstanding or promising results on:

- Visual Recognition
- Image Generation (Pixel RNN, Neural Art, etc.)
- Natural Language Processing (Very deep CNN)
- Speech Recognition (preliminary results)
- Advertising, user prediction (preliminary results)
Conclusions of the Tutorial

• Deep Residual Learning:
  • Ultra deep networks can be easy to train
  • Ultra deep networks can gain accuracy from depth
  • Ultra deep representations are well transferrable
  • Now 200 layers on ImageNet and 1000 layers on CIFAR!

Resources

• Models and Code
  • Our ImageNet models in Caffe: https://github.com/KaimingHe/deep-residual-networks

• Many available implementation
  (see https://github.com/KaimingHe/deep-residual-networks)
  • Facebook AI Research’s Torch ResNet:
    https://github.com/facebook/fb.resnet.torch
    • Torch, CIFAR-10, with ResNet-20 to ResNet-110, training code, and curves: code
    • Lasagne, CIFAR-10, with ResNet-32 and ResNet-56 and training code: code
    • Neon, CIFAR-10, with pre-trained ResNet-32 to ResNet-110 models, training code, and curves: code
    • Torch, MNIST, 100 layers: blog, code
    • A winning entry in Kaggle’s right whale recognition challenge: blog, code
    • Neon, Place2 (mini), 40 layers: blog, code
    • ……

Self-Supervised Learning Project Resource

• [https://github.com/facebookresearch/fair-sslime](https://github.com/facebookresearch/fair-sslime)
Exploring the Limits of Weakly Supervised Pretraining

Laurens van der Maaten

ECCV 2018

First, train a model on a large "source" dataset (say, ImageNet)
Pretraining Vision Models

- First, train a model on a large "source" dataset (say, ImageNet)

- Finetune on a small "target" dataset

- Measure accuracy on target task
Research question

Can we use large amounts of weakly supervised images for pretraining?

Highlights

- We pretrain models by predicting relevant hashtags for images
- We pretrain models to predict 17.5K hashtags for 3.5B images
- After finetuning, we beat the state-of-the-art on, e.g., ImageNet
Hashtag Supervision

- It is easy to get billions of public images and hashtags

- Hashtags are more structured than captions

- Hashtags were often assigned to make images “searchable”
Hashtag Supervision

• But hashtags are not perfect supervision

#cat #travel #thailand #family
Hashtag Supervision

- But hashtags are not perfect supervision
- Some hashtags are not visually relevant
Hashtag Supervision

- But hashtags are not perfect supervision
- Some hashtags are not visually relevant
- Other hashtags are not in the photo

#cat #travel #thailand #family
Hashtag Supervision

- But hashtags are not perfect supervision
- Some hashtags are not visually relevant
- Other hashtags are not in the photo
- And there are many false negatives
Hashtag Supervision

- But hashtags are not perfect supervision
- Some hashtags are not visually relevant
- Other hashtags are not in the photo
- And there are many false negatives
- Is this noise bias or variance? Is scaling up sufficient to reduce the variance?

#cat #travel #thailand #family
#building #fence #...
Experiments

- Select a set of hashtags
- Download all public Instagram images that has at least one of these hashtags
- Use WordNet synsets to merge hashtags into canonical form (merge #brownbear and #ursusarctos)
Experiments

- Select a set of hashtags

- Download all public Instagram images that has at least one of these hashtags

- Use WordNet synsets to merge hashtags into canonical form (merge #brownbear and #ursusarctos)

- The final list has 17,517 hashtags
Experiments

- Select a set of hashtags
- Download all public Instagram images that has at least one of these hashtags
- Use WordNet synsets to merge hashtags into canonical form (merge #brownbear and #ursusarctos)
- Final dataset has ~3.5 billion images
Experiments

- Select a set of hashtags
- Download all public Instagram images that has at least one of these hashtags
- Use WordNet synsets to merge hashtags into canonical form (merge #brownbear and #ursusarctos)
- Final dataset has ~3.5 billion images
Experiments

- Select a set of hashtags
- Download all public Instagram images that has at least one of these hashtags
- Use WordNet synsets to merge hashtags into canonical form (merge #brownbear and #ursusarctos)
- Final dataset has ~3.5 billion images
Experiments

- Select a set of hashtags
- Download all public Instagram images that has at least one of these hashtags
- Use WordNet synsets to merge hashtags into canonical form (merge #brownbear and #ursusarctos)
- De-duplicate test sets against Instagram!

We developed strong near-duplicate detector:

We found that <0.3% of ImageNet images are in our 3.5B Instagram sample.

(This is actually a lower percentage than in most prior papers on “transfer” learning.)
Experiments

- Train ResNeXt-32xCd convolutional networks
- Use c-of-K vector to represent multiple labels
- Train to minimize multi-class logistic loss

most experiments use ResNeXt-101 32x16d
Experiments

- Train ResNeXt-32xCd convolutional networks
- Use c-of-K vector to represent multiple labels
- Train to minimize multi-class logistic loss
- Distribute training batches across 336 GPUs
- Scale learning rate by batch size \((N=8,064)\) after learning rate “warm-up” (Goyal et al., 2017)
Results
Fix Model; Vary Data

- Pretrain model on ImageNet or Instagram
- Finetune on ImageNet

Target task: ImageNet

![Graph showing ImageNet top-1 accuracy across different numbers of classes in target task.](image)
Fix Model; Vary Data

- Pretrain model on ImageNet or Instagram
- Finetune on ImageNet

"standard" ImageNet training
Fix Model; Vary Data

- Pretrain model on ImageNet or Instagram
- Finetune on ImageNet

pre-training on 1B Instagram images, selected to match ImageNet classes
Fix Model; Vary Data

- Pretrain model on ImageNet or Instagram
- Finetune on ImageNet

pretraining on 1-3.5B Instagram images, without selection
Fix Model;
Vary Data

- Pretrain model on ImageNet or Instagram
- Finetune on ImageNet
- Similar results on larger versions of ImageNet

Target task: ImageNet

![Graph showing ImageNet top-1 accuracy for different number of classes in the target task and source task sizes.](image)
Fix Model;
Vary Data

- We observe similar results on the CUB-2011 Birds dataset and Places-365
Fix Data; Vary Model

- Increasing model capacity has a larger positive effect
- Even lower error rates may be possible?
Fix Data; Vary Model

- Increasing model capacity has a larger positive effect
- Even lower error rates may be possible?

best result: 85.4% top-1 accuracy
State-of-the-art

- Compared to prior SotA, +2.7% in top-1 accuracy
  (+1.4% top-5 accuracy)
State-of-the-art

- Compared to prior SotA, +2.7% in top-1 accuracy (+1.4% top-5 accuracy)

<table>
<thead>
<tr>
<th>Model</th>
<th>Image size</th>
<th>Parameters</th>
<th>Mult-adds</th>
<th>Top-1 Acc. (%)</th>
<th>Top-5 Acc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception V2 [27]</td>
<td>224</td>
<td>11.2M</td>
<td>1.94B</td>
<td>74.8</td>
<td>92.2</td>
</tr>
<tr>
<td>NASNet-A (5 @ 1538) [31]</td>
<td>299</td>
<td>10.9M</td>
<td>2.35B</td>
<td>78.6</td>
<td>94.2</td>
</tr>
<tr>
<td>Inception V3 [51]</td>
<td>299</td>
<td>23.8M</td>
<td>5.72B</td>
<td>78.0</td>
<td>93.9</td>
</tr>
<tr>
<td>Xception [52]</td>
<td>299</td>
<td>22.8M</td>
<td>8.38B</td>
<td>79.0</td>
<td>94.5</td>
</tr>
<tr>
<td>Inception ResNet V2 [53]</td>
<td>299</td>
<td>55.8M</td>
<td>13.2B</td>
<td>80.4</td>
<td>95.3</td>
</tr>
<tr>
<td>NASNet-A (7 @ 1920) [31]</td>
<td>299</td>
<td>22.6M</td>
<td>4.93B</td>
<td>80.8</td>
<td>95.3</td>
</tr>
<tr>
<td>ResNeXt-101 64×4 [15]</td>
<td>320</td>
<td>83.6M</td>
<td>31.5B</td>
<td>80.9</td>
<td>95.6</td>
</tr>
<tr>
<td>PolyNet [54]</td>
<td>331</td>
<td>92M</td>
<td>34.7B</td>
<td>81.3</td>
<td>95.8</td>
</tr>
<tr>
<td>DPN-131 [55]</td>
<td>320</td>
<td>79.5M</td>
<td>32.0B</td>
<td>81.5</td>
<td>95.8</td>
</tr>
<tr>
<td>SENet [56]</td>
<td>320</td>
<td>145.8M</td>
<td>42.3B</td>
<td>82.7</td>
<td>96.2</td>
</tr>
<tr>
<td>NASNet-A (6 @ 4032) [31]</td>
<td>331</td>
<td>88.9M</td>
<td>23.8B</td>
<td>82.7</td>
<td>96.2</td>
</tr>
</tbody>
</table>

*Our models:*

- IG-3.5B-17k ResNeXt-101 32×16d | 224 | 194M | 36B | 84.2 | 97.2 |
- IG-940M-1.5k ResNeXt-101 32×32d | 224 | 466M | 87B | 85.1 | 97.5 |
- **IG-940M-1.5k ResNeXt-101 32×48d** | 224 | 829M | 153B | **85.4** | **97.6** |
Learning Curves

- Accuracy on target task improves (almost) log-linearly with data size

- Matching hashtags to target task helps (1.5K tags)

- Positive effect of pre-training increases with difficulty of target task
Label Noise

- Add “noise” that changes a hashtag with probability $p$

- Models are surprisingly robust to label “noise” in source task
Detection

- Train Mask R-CNN with Uru “trunk” on COCO
- Box AP: Average APs over range of IoU values
Detection

- Train Mask R-CNN with Uru “trunk” on COCO
- Box AP: Average APs over range of IoU values

on largest model, +1.5% box AP
Visual Concreteness

- Predicting hashtags is easier for visually "concrete" hashtags?

* Brysbaert et al., 2014
Visual Concreteness

- Predicting hashtags is easier for visually "concrete" hashtags
- Correlation: $\rho = 0.43$

* Brysbaert et al., 2014
Conclusion

Billion-scale pretraining leads to >2.0% reduction in ImageNet top-1 error

Discussion

- Results suggest further improvements are possible
- Current networks are underfitting on datasets at this scale
- Hypothesis: hashtag-based pre-training particularly beneficial as target task involves recognition of larger visual variety