CSE5194: ResNet and ResNeXt

The Ohio State University
Zhengqi Dong
E-mail: dong.760@osu.edu
Date: 09/20/2019
Part 1: (ResNet) Deep Residual Learning for Image Recognition
1. Introduction: Background and Key Contribution

• Background
  o Published in 2015 by Kaiming He, and etc, with 56517 citation so far
  o Won 1st place on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation at the ILSVRC & COCO 2015 competition.
  o Achieved 3.57% error on the ImageNet test dataset
  o A 28% improvement on the COCO object detection dataset

• Key contribution:
  o Solved the **degradation problem** -- with the network depth increasing, accuracy gets saturated and degraded rapidly.
  o The residual networks are easier to optimize and can gain higher accuracy as increased depth.
1. Introduction: Residual Neural Network

• Question: Shouldn’t building better neural networks as easy as stacking more layers?

• Problem: Vanishing/Exploding gradients, degradation problem.
• Old Solution: Normalized initialization and normalized the intermediate layers
• New Solution: Residual Learning Block

1. Introduction: Problem Definition

• Vanishing Gradient
  • Your gradient/derivative can get very very very small
  • Even exponentially small
  • Make the training difficult to converge, or not converge.

• Exploding Gradients
  • Your gradient/derivative can get very very very larger
  • Make the gradient exploded/diverge…

• Degradation problem:
  • With the network depth increasing, accuracy get saturated

Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.
2. Network Design: Residual Learning Block

• Better solution: ResNet
• Inspired by VGG nets: stacking building block of the same shape
• Residual Block
  • Insert “identity shortcuts”, aka shortcut connection, or skip connection
  • Allow the information directly pass to deeper layer.
  • Add neither extra parameter nor computational complexity
• ResNet = a stack of Residual Block
2. Network Design: Plain vs ResNet vs VGG

- ResNet = Plain Network + Short Connection
  - Residual network can gain accuracy from considerably increased depth.
- Top: a ResNet with 34 parameter layers (3.6 billion FLOPs).
- Middle: a plain network with 34 parameter layer (3.6 billion FLOPs).
- Bottom: VGG-19 model (19.6 billion FLOPs).
2. Network Design: Shortcut Connections

- Identity Mapping and Projection $W_s$:
  - If input and output has same dimensions (denoted by solid line):
    \[ y = F(x, \{W_i\}) + x. \]  
  - If input and output has different dimensions (denoted by dotted line):
    - Option A: Zero padded for extra dimension
    - Option B: Perform the projection shortcut to match the dimension (done by 1*1 Conv).
2. Network Design: Comparison of three shortcut strategies

- **Comparison**
  - Type A: zero-padding for increasing dimensions, and rest are identity shortcut (parameter free)
  - Type B: Projection for increasing dimensions only
  - Type C: Projection for all shortcut

- **Conclusion**:
  - Type C is marginally better as extra parameters introduced, but time complexity and model size are doubled
  - Type A is used for rest of paper

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-1 Err</th>
<th>Top-5 Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-16 [41]</td>
<td>28.07</td>
<td>9.33</td>
</tr>
<tr>
<td>GoogleLeNet [44]</td>
<td>-</td>
<td>9.15</td>
</tr>
<tr>
<td>Plain-34</td>
<td>28.54</td>
<td>10.02</td>
</tr>
<tr>
<td>ResNet-34 A</td>
<td>25.03</td>
<td>7.76</td>
</tr>
<tr>
<td>ResNet-34 B</td>
<td>24.52</td>
<td>7.46</td>
</tr>
<tr>
<td>ResNet-34 C</td>
<td>24.19</td>
<td>7.40</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>22.85</td>
<td>6.71</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>21.75</td>
<td>6.05</td>
</tr>
<tr>
<td>ResNet-152</td>
<td>21.43</td>
<td>5.71</td>
</tr>
</tbody>
</table>

Table 3. Error rates (% \textit{10-crop testing}) on ImageNet validation. VGG-16 is based on our test. ResNet-50/101/152 are option B that only uses projections for increasing dimensions.

Identity Shortcut:

\[ y = f(x, \{W_i\}) + x. \]  \hspace{1cm} (1)

Projection Shortcut:

\[ y = f(x, \{W_i\}) + W_s x. \]  \hspace{1cm} (2)
3. ImageNet Classification: Bottleneck Building Block

- **Dataset: ImageNet 2012 classification dataset**
  - Training dataset: 1.28M train images
  - Validation dataset: evaluated on 50K validation images
  - Testing dataset: final resulted (top1 and top5 error rate) tested on 100K test images.
  - Classes: consist of 1000 classes

- **Larger network architectures evaluated in ImageNet: “Bottleneck” building block**

![Image of Bottleneck Building Block](image-url)
4. ImageNet Classification: Performance analysis

- Plain network vs ResNet
  - Obvious degradation problem
  - Plain net has higher training error throughout the whole training procedure
  - Situation reversed with ResNets
5. Implementation Tricks (read offline)

• Augmentation follow the practice [8, 9]
  o Resize the images with its shorter side randomly sampled in between [256; 480] for scale augmentation. [9]
  o Standard color augmentation. [8]
  o A 224x224 crop is randomly sampled from an image or its horizontal flip, with the per-pixel mean subtracted [8]

• Other techniques follow [5, 6, 7]
  o Batch normalization right after each Conv and before activation. [7]
  o Initialized the weight as in [5] and train all plain/residual nets from scratch.
  o Use SGF with a mini-batch size of 256.
  o The learning rate starts from 0.1 and is divided by 10 when the error stagnated, and the models are trained for up to 60*10^4 iterations.
  o Use a weight decay of 0.0001 and a momentum of 0.9.
  o No dropout. [6] [7]

• Code is available: https://github.com/KaimingHe/deep-residual-networks
Part2: (ResNeXt) Aggregated Residual Transformations for Deep Neural Networks
1. Introduction: Background and Key Contribution

• Background:
  o Won second place in the 2016 ILSVRC image classification task
  o A simpler design: a 101-layer ResNeXt achieved better accuracy than ResNet-200 but has only 50% complexity.
  o The transition from “Feature Engineering” to “Network Engineering”: In contrast to traditional hand-designed features (e.g. SIFT and HOG), human effort has been shifted to designing better neural network architecture for learning representation.

• Main Contribution:
  o Adopted similar strategy inherited from VGG/ResNets: stack modules of same topology.
  o Exploited the split-transform-merge (aka multi-path) strategy in an easy and extensible way.
  o Introduces a new dimension for gaining the accuracy: Cardinality

Reference: https://arxiv.org/abs/1611.05431
Question: Shouldn’t building better neural networks as easy as stacking more layers?

- Old approach: going deeper (increase # layers) and wider (increase bottleneck width)
- New approach: increase cardinality C
- **Cardinality**: the size of the set of transformation (or # of branches/paths/groups)
2. Related work: Grouped Convolutions

- Grouped Convolutions:
  - a process of applying multiple kernels/filters per layer on same images
  - Allow the training of network across multiple GPUs, and thus results more efficient parallelized training.
  - Learned better representations, [https://blog.yani.io/filter-group-tutorial/](https://blog.yani.io/filter-group-tutorial/)


3. Method: Two Template rules

• Two simple rules:
  1. If producing spatial maps of the same size, the blocks share the same hyper-parameters (width and filter size)
  2. Each time when the spatial map is downsampled by a factor of 2, the width of the blocks is multiplied by a factor of 2.

![Table 1. (Left) ResNet-50. (Right) ResNeXt-50 with a 32×4d template (using the reformulation in Fig. 3(c)). Inside the brackets are the shape of a residual block, and outside the brackets is the number of stacked blocks on a stage. “C=32” suggests grouped convolutions [24] with 32 groups. The numbers of parameters and FLOPs are similar between these two models.](image)
3. Method: Aggregated Transformations

• Simple Neurons:

\[
D \sum_{i=1}^{w_i x_i}
\]

(1)

• Aggregated Transformation:

\[
y = x + \sum_{i=1}^{C} T_i(x)
\]

(3)

• C: cardinality, size of the set of transformations to be aggregated

• T_i(x): arbitrary transformation function, e.g. linear transformation
3. Method: Equivalent building block of ResNeXt

- Fig. 3 a): \[ y = x + \sum_{i=1}^{C} T_i(x), \] (3)
- Fig. 3 b): Similar to Inception-ResNet block, but the same topology shared amount the multiple paths.
- Fig. 3 c): applied grouped convolutions
3. Method: Inception-ResNet vs ResNeXt

- **Inception-ResNet**
  - Many hyper-parameters need to be tailored for each individual transformation
  - Hard to adapt to a new dataset/task

- **ResNeXt**:
  - Use the same topology among all paths
  - Proved a better accuracy over all Inception model


Figure: Inception-ResNet-v2 module

Figure: ResNeXt building block with 32 cardinality
For evaluating different cardinalities $C$, the complexity (# params) is preserved by adjusting the width of bottleneck.

- Calculate the #params for original network
  - ResNet-50 (1x64d) = $256 \times 64 + 3 \times 3 \times 64 \times 64 + 64 \times 256 \approx 70k$ params
- Calculate the #params for bottleneck width $d$:
  - ResNeXt-50 (32x4d) = $C \times (256 \times d + 3 \times 3 \times d \times d + d \times 256) \approx 70k$ params
4. ImageNet-1K: Increasing Cardinality Vs Deeper/Wider

- Original approach:
  - Going Deeper: 0.3% improvement
  - Going Wider: 0.7% improvement

- New approach:
  - Increasing Cardinality (C): 1.3% improvement

- Conclusion: Increasing cardinality C shows much better results than going deeper or wider

![Graph showing training curves for ImageNet-1K with increasing cardinality vs deeper/wider approaches.](image)

Table 4. Comparisons on ImageNet-1K when the number of FLOPs is increased to 2× of ResNet-101’s. The error rate is evaluated on the single crop of 224×224 pixels. The highlighted factors are the factors that increase complexity.
5. Implementation Details (read offline)

- A 224x224 crop is randomly cropped from a resized image using the scale and aspect ratio augmentation [13] [10]
- The shortcuts connection for different input-output dimension are project, type B in [12]
- Downsampling of conv3, 4, and 5 is done by stride-2 convolutions in the 3x3 layer of the first block in each stage, as suggested in [10]
- Use SGD with a mini-batch size of 256 on 8 GPUs (32 samples per GPUs for Data parallelism)
- The weight decay is 0.0001 and the momentum is 0.9
- Start from a learning rate of 0.1, and divide it by 10 for three times using the schedule in [10].
- Adopt the weight initialization of [12]
- evaluate the error on the single 224x224 center crop from an image whose shorter side is 256.
- Choose Fig.3 c) ResNeXt block, grouped convolutions.
- Batch normalization(BN) is performed right after the convolutions, and ReLU is performed right after BN, except the output of the block [12]
- Code is available of https://github.com/facebookresearch/ResNeXt
  - [PyTorch version]: https://pytorch.org/hub/pytorch_vision_resnext/
Conclusion

• ResNet
  o Vanishing gradient, Exploding gradient, and degradation problem
  o Residual building block, Bottleneck Building block
  o Shortcut connection, Projection shortcut
  o Deep residual network are easy to optimize and can gain a better accuracy as the increased of network depth.

• ResNeXt
  o multi-branch/path (split-transform-merge in Inception net) strategy
  o Two template rule, Aggregated transformation
  o Trade-off between Cardinality(C) and Bottleneck width(d)
  o Increasing cardinality is more effective than going deeper/wider.
Thank You!
Any Question?
[3] Deeplearning.ai, https://www.youtube.com/watch?v=ZILlbUvp5lk&list=PLpFsSf5Dm-pd5d3rjNtIXUHT-v7bdaE1e&index=113