Dynamic Neural Networks for Efficient Image and Video Classification

Rogerio Schmidt Feris
MIT-IBM Watson AI Lab
http://rogerioferis.com

LatinX in AI Research at ICML 2020
ImageNet Classification (top-5 accuracy)

- **2010**: 72%
- **2011**: 74%
- **2012**: 84.6%
- **2013**: 86.8%
- **2014**: 92%
- **2015**: 94.4%
- **2016**: 95.8%
- **2017**: 96.2%
- **2018**: 97.6%
- **2019**: 98.2%

Numbers from [paperswithcode.com](http://paperswithcode.com)
Better Results $\Rightarrow$ More Complexity

- **2010**: 72%
- **2011**: 74%
- **2012**: 84.6%
- **2013**: 86.8%
- **2014**: 92%
- **2015**: 94.4%
- **2016**: 95.8%
- **2017**: 96.2%
- **2018**: 97.6%
- **2019**: 98.2%

- **AlexNet**: 60M param
- **Noisy Student**: 480M param
- **FixResNeXt**: 829M param

Shallow 10 Layers 20 Layers 100+ Layers
Many applications require real-time inferencing.
Model Compression and Acceleration

- Low-rank factorization, Knowledge Distillation, Pruning, Quantization, Neural Architecture Search, etc.

EfficientNet [Tan & Le, 2019]

MobileNet V3 [Howard et al, 2019]

ProxylessNAS [Cai et al, 2019]
Most methods rely on **one-size-fits-all networks** that require the same fixed set of features to be extracted for all inputs, no matter their complexity.
This talk: Dynamic (Adaptive) Neural Networks for Efficient Image and Video Classification

- Networks models that are dynamically reconfigured depending on the input

- Conditional Computation [Bengio et al, 2013/2016]
Feed-Forward Convolutional Neural Networks

Adapted from Veit et al
Feed-Forward Convolutional Neural Networks

What happens when we delete a step?

Adapted from Veit et al
Feed-Forward Convolutional Neural Networks

Adapted from Veit et al
What happens if we delete a layer at test time?

Adapted from Veit et al.
What happens if we delete a layer at test time?

Adapted from Veit et al.
Why does this happen?

VGG

ResNet

Adapted from Veit et al
Why does this happen?

The unraveled view is equivalent and showcases the many paths in ResNet.
Deletion of a Layer
Deletion of a Layer

All paths are affected

VGG

Only half of the paths are affected

ResNet

Adapted from Veit et al
Can we delete a sequence of layers without performance drop?

This experiment [Veit et al, 2016]:
- Layers were dropped randomly
- Global dropping strategy for all images
BlockDrop: Dynamic Inference Paths in Residual Networks

Zuxuan Wu*, Tushar Nagarajan*, Abhishek Kumar, Steven Rennie, Larry S. Davis, Kristen Grauman, Rogerio Feris

CVPR 2018

* Authors contributed equally
Do we really need to run 100+ layers / residual blocks of a neural network if we have an “easy” input image?
“Dropping some blocks during testing doesn’t hurt performance much”

(Veit et al., NIPS 16)
How to determine which blocks to drop depending on the input image?

[Wu & Nagarajan et al, CVPR 2018]
Our Idea: BlockDrop

“Predict which blocks to drop conditioned on the input image, in one shot, without compromising accuracy”

[Wu & Nagarajan et al, CVPR 2018]
BlockDrop: Dynamic Inference Paths in Residual Networks

[Wu & Nagarajan et al, CVPR 2018]
BlockDrop: Dynamic Inference Paths in Residual Networks [CVPR 2018]

Policy Network Training through Reinforcement Learning

[Wu & Nagarajan et al, CVPR 2018]
BlockDrop: Dynamic Inference Paths in Residual Networks

Results on ImageNet:

20% - 36% computational savings (FLOPs)

Complementary to other model compression techniques

[Wu & Nagarajan et al, CVPR 2018]
SpotTune: Transfer Learning through Adaptive Fine-Tuning

Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, Rogerio Feris

CVPR 2019
Data Efficiency: Transfer Learning

- Fine-tuning is arguably the most widely used approach for transfer learning

- Existing methods are ad-hoc in terms of determining where to fine-tune in a deep neural network (e.g., fine-tuning last k layers)

- We propose SpotTune, a method that automatically decides, per training example, which layers of a pre-trained model should have their parameters frozen (shared with the source domain) or fine-tuned (adapted to the target domain)

[Guo et al, CVPR 2019]
Which layers to freeze and which layers to fine-tune?

[Guo et al, CVPR 2019]
SpotTune: Transfer Learning through Adaptive Fine-Tuning

[Guo et al, CVPR 2019]
SpotTune: Transfer Learning through Adaptive Fine-Tuning

SpotTune automatically identifies the right fine-tuning policy for each dataset, for each training example.

[Guo et al, CVPR 2019]
SpotTune: Transfer Learning through Adaptive Fine-Tuning

<table>
<thead>
<tr>
<th>Method</th>
<th>#par</th>
<th>ImNet</th>
<th>Airc.</th>
<th>C100</th>
<th>DPed</th>
<th>DTD</th>
<th>GTSR</th>
<th>Flwr</th>
<th>OGl</th>
<th>SVHN</th>
<th>UCF</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scratch</td>
<td>10x</td>
<td>59.87</td>
<td>57.10</td>
<td>75.73</td>
<td>91.20</td>
<td>37.77</td>
<td>96.55</td>
<td>56.30</td>
<td>88.74</td>
<td>96.63</td>
<td>43.27</td>
<td>1625</td>
</tr>
<tr>
<td>Scratch+ [37]</td>
<td>11x</td>
<td>59.67</td>
<td>59.59</td>
<td>76.08</td>
<td>92.45</td>
<td>39.63</td>
<td>96.90</td>
<td>56.66</td>
<td>88.74</td>
<td>96.78</td>
<td>44.17</td>
<td>1826</td>
</tr>
<tr>
<td>Feature Extractor</td>
<td>1x</td>
<td>59.67</td>
<td>23.31</td>
<td>63.11</td>
<td>80.33</td>
<td>55.53</td>
<td>68.18</td>
<td>73.69</td>
<td>58.79</td>
<td>43.54</td>
<td>26.80</td>
<td>544</td>
</tr>
<tr>
<td>Fine-tuning [38]</td>
<td>10x</td>
<td>60.32</td>
<td>61.87</td>
<td>82.12</td>
<td>92.82</td>
<td>55.53</td>
<td>99.42</td>
<td>81.41</td>
<td>89.12</td>
<td>96.55</td>
<td>51.20</td>
<td>3096</td>
</tr>
<tr>
<td>BN Adapt. [5]</td>
<td>1x</td>
<td>59.87</td>
<td>43.05</td>
<td>78.62</td>
<td>92.07</td>
<td>51.60</td>
<td>95.82</td>
<td>74.14</td>
<td>84.83</td>
<td>94.10</td>
<td>43.51</td>
<td>1353</td>
</tr>
<tr>
<td>LwF [26]</td>
<td>10x</td>
<td>59.87</td>
<td>61.15</td>
<td>82.23</td>
<td>92.34</td>
<td>58.83</td>
<td>97.57</td>
<td>83.05</td>
<td>88.08</td>
<td>96.10</td>
<td>50.04</td>
<td>2515</td>
</tr>
<tr>
<td>Series Res. adapt. [37]</td>
<td>2x</td>
<td>60.32</td>
<td>61.87</td>
<td>81.22</td>
<td>93.88</td>
<td>57.13</td>
<td>99.27</td>
<td>81.67</td>
<td>89.62</td>
<td>96.57</td>
<td>50.12</td>
<td>3159</td>
</tr>
<tr>
<td>Parallel Res. adapt. [38]</td>
<td>2x</td>
<td>60.32</td>
<td>64.21</td>
<td>81.92</td>
<td>94.73</td>
<td>58.83</td>
<td>99.38</td>
<td>84.68</td>
<td>89.21</td>
<td>96.54</td>
<td>50.94</td>
<td>3412</td>
</tr>
<tr>
<td>Res. adapt. (large) [37]</td>
<td>12x</td>
<td>67.00</td>
<td>67.69</td>
<td>84.69</td>
<td>94.28</td>
<td>59.41</td>
<td>97.43</td>
<td>84.86</td>
<td>89.92</td>
<td>96.59</td>
<td>52.39</td>
<td>3131</td>
</tr>
<tr>
<td>Res. adapt. decay [37]</td>
<td>2x</td>
<td>59.67</td>
<td>61.87</td>
<td>81.20</td>
<td>93.88</td>
<td>57.13</td>
<td>97.57</td>
<td>81.67</td>
<td>89.62</td>
<td>96.13</td>
<td>50.12</td>
<td>2621</td>
</tr>
<tr>
<td>Res. adapt. finetune all [37]</td>
<td>2x</td>
<td>59.23</td>
<td>63.73</td>
<td>81.31</td>
<td>93.30</td>
<td>57.02</td>
<td>97.47</td>
<td>83.43</td>
<td>89.82</td>
<td>96.17</td>
<td>50.28</td>
<td>2643</td>
</tr>
<tr>
<td>DAN [39]</td>
<td>2x</td>
<td>57.74</td>
<td>64.12</td>
<td>80.07</td>
<td>91.30</td>
<td>56.54</td>
<td>98.46</td>
<td>86.05</td>
<td>89.67</td>
<td>96.77</td>
<td>49.48</td>
<td>2851</td>
</tr>
<tr>
<td>PiggyBack [31]</td>
<td>1.28x</td>
<td>57.69</td>
<td>65.29</td>
<td>79.87</td>
<td>96.99</td>
<td>57.45</td>
<td>97.27</td>
<td>79.09</td>
<td>87.63</td>
<td>97.24</td>
<td>47.48</td>
<td>2838</td>
</tr>
<tr>
<td>SpotTune</td>
<td>11x</td>
<td>60.32</td>
<td>63.91</td>
<td>80.48</td>
<td>96.49</td>
<td>57.13</td>
<td>99.52</td>
<td>85.22</td>
<td>88.84</td>
<td>96.72</td>
<td>52.34</td>
<td><strong>3612</strong></td>
</tr>
</tbody>
</table>

SpotTune sets the new state of the art on the Visual Decathlon Challenge

[Guo et al, CVPR 2019]
AdaShare: Learning What to Share for Efficient Multi-Task Learning

Ximeng Sun, Rameswar Panda, Rogerio Feris, Kate Saenko

NeurIPS 2020
Hard Parameter Sharing

- Hand-designed architectures composed of base layers that are shared across tasks and specialized branches that learn task-specific features.

- Performance depends on “where to branch” in the network [Misra et al, 2016]

- The space of possible branching architectures is combinatorially large!
Soft Parameter Sharing

- Network column for each task and a mechanism for feature sharing between columns.

Number of parameters grow linearly with the number of tasks!
Can we determine which layers in the network should be shared across which tasks and which layers should be task-specific to achieve the best accuracy/memory footprint trade-off for scalable and efficient multi-task learning?
Proposed Approach: AdaShare

- Single network that supports separate execution paths for different tasks
AdaShare: Learning what to Share in Multi-Task Learning
AdaShare: Learning what to Share in Multi-Task Learning
AdaShare: Experimental Results

- **CityScapes [2 tasks].** *AdaShare* achieves the best performance on 5 out of 7 metrics using less than 1/2 parameters of most baselines.

<table>
<thead>
<tr>
<th>Model</th>
<th># Params ↓</th>
<th>Semantic Seg.</th>
<th>Depth Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>mIoU ↑</td>
<td>Pixel Acc ↑</td>
</tr>
<tr>
<td>Single-Task</td>
<td>2</td>
<td>40.2</td>
<td>74.7</td>
</tr>
<tr>
<td>Multi-Task</td>
<td>1</td>
<td>37.7</td>
<td>73.8</td>
</tr>
<tr>
<td>Cross-Stitch</td>
<td>2</td>
<td>40.3</td>
<td>74.3</td>
</tr>
<tr>
<td>Sluice</td>
<td>2</td>
<td>39.8</td>
<td>74.2</td>
</tr>
<tr>
<td>NDDR-CNN</td>
<td>2.07</td>
<td><strong>41.5</strong></td>
<td>74.2</td>
</tr>
<tr>
<td>MTAN</td>
<td>2.41</td>
<td>40.8</td>
<td>74.3</td>
</tr>
<tr>
<td><strong>AdaShare</strong></td>
<td>1</td>
<td><strong>41.5</strong></td>
<td><strong>74.9</strong></td>
</tr>
</tbody>
</table>
AdaShare: Experimental Results

- **NYU v2 [3 tasks].** AdaShare achieves the best performance on 10 out of 12 metrics using less than 1/3 parameters of most baselines.

<table>
<thead>
<tr>
<th>Model</th>
<th># Params ↓</th>
<th>Semantic Seg.</th>
<th></th>
<th>Surface Normal Prediction</th>
<th>Depth Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>mIoU ↑</td>
<td>Pixel Acc ↑</td>
<td>Error ↓</td>
<td>Error ↓</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Mean</td>
<td>Abs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Median</td>
<td>Rel</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11.25°</td>
<td>1.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>22.5°</td>
<td>1.25²</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30°</td>
<td>1.25³</td>
</tr>
<tr>
<td>Single-Task</td>
<td>3</td>
<td>27.5</td>
<td>58.9</td>
<td>17.5</td>
<td>0.62</td>
</tr>
<tr>
<td>Multi-Task</td>
<td>1</td>
<td>24.1</td>
<td>57.2</td>
<td>16.6</td>
<td>0.58</td>
</tr>
<tr>
<td>Cross-Stitch</td>
<td>3</td>
<td>25.4</td>
<td>57.6</td>
<td>17.2</td>
<td>0.58</td>
</tr>
<tr>
<td>Sluice</td>
<td>3</td>
<td>23.8</td>
<td>56.9</td>
<td>17.2</td>
<td>0.58</td>
</tr>
<tr>
<td>NDDR-CNN</td>
<td>3.15</td>
<td>21.6</td>
<td>53.9</td>
<td>17.1</td>
<td>0.66</td>
</tr>
<tr>
<td>MTAN</td>
<td>3.11</td>
<td>26.0</td>
<td>57.2</td>
<td>16.6</td>
<td>0.57</td>
</tr>
<tr>
<td><strong>AdaShare</strong></td>
<td><strong>1</strong></td>
<td><strong>30.2</strong></td>
<td><strong>62.4</strong></td>
<td><strong>16.6</strong></td>
<td><strong>0.55</strong></td>
</tr>
</tbody>
</table>

![Image of AdaShare diagram and table comparison]
AdaShare: Experimental Results

- **Tiny-Taskonomy [5 Tasks]**. AdaShare outperforms the baselines on 3 out of 5 tasks using less than 1/5 parameters of most baselines.

<table>
<thead>
<tr>
<th>Models</th>
<th># Params ↓</th>
<th>Seg ↓</th>
<th>SN ↑</th>
<th>Depth ↓</th>
<th>Keypoint ↓</th>
<th>Edge ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Task</td>
<td>5</td>
<td>0.575</td>
<td>0.707</td>
<td>0.022</td>
<td>0.197</td>
<td>0.212</td>
</tr>
<tr>
<td>Multi-Task</td>
<td>1</td>
<td>0.587</td>
<td>0.702</td>
<td>0.024</td>
<td>0.194</td>
<td>0.201</td>
</tr>
<tr>
<td>Cross-Stitch</td>
<td>5</td>
<td>0.560</td>
<td>0.684</td>
<td>0.022</td>
<td>0.202</td>
<td>0.219</td>
</tr>
<tr>
<td>Sluice</td>
<td>5</td>
<td>0.610</td>
<td>0.702</td>
<td>0.023</td>
<td><strong>0.192</strong></td>
<td>0.198</td>
</tr>
<tr>
<td>NDDR-CNN</td>
<td>5.41</td>
<td><strong>0.539</strong></td>
<td>0.705</td>
<td>0.024</td>
<td>0.194</td>
<td>0.206</td>
</tr>
<tr>
<td>MTAN</td>
<td>4.51</td>
<td>0.637</td>
<td>0.702</td>
<td>0.023</td>
<td>0.193</td>
<td>0.203</td>
</tr>
<tr>
<td><strong>AdaShare</strong></td>
<td><strong>1</strong></td>
<td>0.566</td>
<td><strong>0.707</strong></td>
<td>0.025</td>
<td><strong>0.192</strong></td>
<td><strong>0.193</strong></td>
</tr>
</tbody>
</table>
Dynamic Neural Networks for Video Classification

**MIT:** Bowen Pan, Camilo Fosco, Alex Andonian, Aude Oliva

**BU & IBM:** Ximeng Sun and Kate Saenko

**IBM:** Yue Meng, Rameswar Panda, Chung-Ching Lin, Richard Chen, Quanfu Fan, Prasanna Sattigeri, Leonid Karlinsky, Rogerio Feris
AR-Net: Adaptive frame resolution for efficient action recognition [ECCV 2020]

- Key idea is to select the resolution of each frame on-the-fly to achieve the best accuracy/efficiency trade-off in video classification
AR-Net: Experimental Results

ActivityNet dataset

- MultiAgent [58]
- LiteEval [60]
- AdaFrame10 [61]
- AdaFrame5 [61]
- ListenToLook(MN|R) [20]
- ListenToLook(IA|R) [20]
- ListenToLook(IA|IA) [20]
- SCSampler [33]
- Ours (ResNet)
- Ours (EfficientNet)
AdaMML: Adaptive multimodal learning for efficient video recognition

- Key idea is to select on-the-fly the optimal modalities for each video segment conditioned on the input for efficient video recognition.
AdaMML: Experimental Results

RGB+Flow+Audio Performance on Kinetics-Sounds dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc. (%)</th>
<th>Selection Rate (%)</th>
<th>GFLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB</td>
<td>82.85</td>
<td>100</td>
<td>141.36</td>
</tr>
<tr>
<td>Flow</td>
<td>75.73</td>
<td></td>
<td>163.39</td>
</tr>
<tr>
<td>Audio</td>
<td>65.49</td>
<td></td>
<td>3.82</td>
</tr>
<tr>
<td>Naïve</td>
<td>82.81</td>
<td>100</td>
<td>308.56</td>
</tr>
<tr>
<td>AdaMML-Flow</td>
<td>88.54</td>
<td>56.13 20.31</td>
<td>132.94</td>
</tr>
<tr>
<td>AdaMML-RGBDiff</td>
<td>89.06</td>
<td>55.06 26.82</td>
<td>141.97</td>
</tr>
</tbody>
</table>

Action: Playing Accordion
Summary

- Adaptive (dynamic) neural networks for efficient image and video classification

BlockDrop

SpotTune

Adashare

AdaMML

AR-Net
References

- Y. Lu, A. Kumar, S. Zhai, Y. Cheng, T. Javidi, R. S. Feris. “Fully-adaptive Feature Sharing in Multi-Task Networks with Applications in Person Attribute Classification” CVPR 2017

(* equal contribution)