Scaling-Up Deep Learning For Autonomous Vehicles

JOSE M. ALVAREZ | NVIDIA | GPU TECHNOLOGY CONFERENCE | San Jose 2019
NVIDIA
Al-Infra
Al-Infra Team

One of our top Goals

Industry grade Deep Learning to take AV Perception DNN into production, tested in multiple locations and conditions.
DL For Autonomous Vehicles

PBs of data, large-scale labeling, large-scale training, etc.

Inference optimized DNN (TensorRT)

POST /datasets/{id}

Datasets

Manually selected data

Deep Learning

Train/test data

Metrics

Simulation, verification results

Labeling

Inference optimized DNN (TensorRT)
AI-Infra Team

One of our top Goals

Industry grade Deep Learning to take AV Perception DNN into production, tested in multiple locations and conditions.

High-quality system → No failures in Millions of miles → Quality-driven AV Perception

The Challenge of Scale
Self-driving cars requires tremendously large datasets for training and testing.
DL for Autonomous Driving

The Challenge of Scale

Data Collection fleet => 100 cars
2000h of data collected per car, per year
Assuming 5 2MP cameras per car, radar data, etc. => 1 TB / h / car
Grand total of 200 PB collected per year!
Only 1/1000 likely to be used for training (curated, labeled data)
DL for Autonomous Vehicles

PBs of data, large-scale labeling, large-scale training, etc.

Mine highly confused / most informative data

Active Learning

Manually selected data

Inference optimized DNN (TensorRT)

POST /datasets/{id}

Dataset

Deep Learning

Metrics

Simulation, verification results

Train/test data

Trained Models

Labels

Scaled-up Dataset

Labeling

Inference optimized DNN (TensorRT)
DL for Autonomous Vehicles

The Challenge of Scale

Large Datasets:
12.1 years training a ResNet50-like network on Pascal
1.5 years on DGX1 w/ Volta

With 8 DGX1s, and 1/10th of that training data, can train in 1 week
Deep Learning

POST /datasets/{id}

Datasets

Trained Models

Accuracy / Efficiency DL

Deep Learning

Inference optimized DNN (TensorRT)

Metrics

Simulation, verification results

Train/test data

Labels

Labeling

Mine highly confused / most informative data

Manually selected data

PBs of data, large-scale labeling, large-scale training, etc.

Simulation, verification results

Inference optimized DNN (TensorRT)
DL for Autonomous Driving

The Challenge of Scale

Robustness / Reliable:

Tested around the world under multiple conditions

Need to show 0 failures in > 1M miles, covering 1000s of Conditions...
POST /datasets/{id}

DL for Autonomous Vehicles
PBs of data, large-scale labeling, large-scale training, etc.

Mine highly confused / most informative data

Manually selected data

Inference optimized DNN (TensorRT)

Deep Learning

Trained Models

Datasets

Labels

Train/test data

Deep Learning

Simulation, verification results

Robustness: (Domain Adaptation, …)

Metrics
Talk Road Map

- Creating the Right Datasets
  - Active Learning
  - Domain Adaptation
- Improving Network Accuracy / Efficiency via overparameterization
  - Joint Training and pruning
  - Exploiting linear redundancies to train small networks.
Creating the right datasets is the cornerstone of (supervised) machine learning.
Creating the Right Datasets

Some Samples Are Much More Informative Than Others
1. How do we find the **most informative** unlabeled data to build the right datasets the fastest?

2. How do we build training **datasets that are 1/1000 the size** for the same result?
Active Learning
Active Learning

Training models

Model uncertainty

Collecting data
Bayesian networks are the principled way to model uncertainty. However, they are computationally demanding:

- Training: Intractable without approximations.
- Testing: distributions need ~100 forward passes (varying the model)
Active Learning

Bayesian Deep Networks (BNN)

A common (cheaper) approach consists of using ensembles of networks:
- Samples from the same distribution as the training set will have consensus while other samples will not.
- Ensembles do not approximate uncertainty in the same manner as a BNN.
  - I.e., parameters in different members serve for different purpose.
Active Learning

Bayesian Deep Networks (BNN)

We propose an approximation to BNN to train a network using ensembles.

- We regularize the weights in the ensemble to approximate probability distributions.

[Chitta, Alvarez, Lesnikowski], Large-Scale Visual Active Learning with Deep Probabilistic Ensembles. Arxiv 2018
Active Learning
Bayesian Deep Networks (BNN)

Given this new network design, we can sample from this and quantify the uncertainty of the model on a new (unlabeled) sample.

Label those where the model is more uncertain.

[Chitta, Alvarez, Lesnikowski], Large-Scale Visual Active Learning with Deep Probabilistic Ensembles. Arxiv 2018
Classification Results
Active Learning

Quantitative Results

Image classification on Cifar-10:
- up to 50k training images
- 10K validation images
- ResNet-18

[Chitta, Alvarez, Lesnikowski], Large-Scale Visual Active Learning with Deep Probabilistic Ensembles. Arxiv 2018
Active Learning

Quantitative Results

Competitive results using \(~1/4^{th}\) of the training data

[Chitta, Alvarez, Lesnikowski], Large-Scale Visual Active Learning with Deep Probabilistic Ensembles. Arxiv 2018
Active Learning

Quantitative Results

Table 2. Validation Accuracies comparing the proposed approach to standard ensembling. Initial 4% is randomly sampled.

<table>
<thead>
<tr>
<th>Task</th>
<th>Data Sampling</th>
<th>8%</th>
<th>16%</th>
<th>32%</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>Random</td>
<td>80.60</td>
<td>86.80</td>
<td>91.08</td>
</tr>
<tr>
<td></td>
<td>Standard</td>
<td>82.41</td>
<td>90.05</td>
<td>94.13</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td><strong>82.88</strong></td>
<td><strong>90.15</strong></td>
<td><strong>94.33</strong></td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>Random</td>
<td>39.57</td>
<td>54.92</td>
<td>66.65</td>
</tr>
<tr>
<td></td>
<td>Standard</td>
<td>40.49</td>
<td>56.89</td>
<td>69.68</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td><strong>40.87</strong></td>
<td><strong>56.94</strong></td>
<td><strong>70.12</strong></td>
</tr>
</tbody>
</table>

[Chitta, Alvarez, Lesnikowski], Large-Scale Visual Active Learning with Deep Probabilistic Ensembles. Arxiv 2018
# Active Learning

## Quantitative Results

### CIFAR-10

<table>
<thead>
<tr>
<th>Method</th>
<th>10k (20%)</th>
<th>50k (100%)</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core-set [43]</td>
<td>74</td>
<td>90</td>
<td>82.2</td>
</tr>
<tr>
<td>Ensemble [2]</td>
<td>85</td>
<td>95.5</td>
<td>89</td>
</tr>
<tr>
<td>Single + Random</td>
<td>85.2</td>
<td>94.4</td>
<td>90.3</td>
</tr>
<tr>
<td>DPE + Random</td>
<td>87.9</td>
<td>95.2</td>
<td>92.3</td>
</tr>
<tr>
<td>Single + Linear-8</td>
<td>87.5</td>
<td>94.4</td>
<td>92.7</td>
</tr>
<tr>
<td><strong>Ours (DPE + Linear-8)</strong></td>
<td><strong>92</strong></td>
<td><strong>95.2</strong></td>
<td><strong>96.3</strong></td>
</tr>
</tbody>
</table>

[Chitta, Alvarez, Lesnikowski], Large-Scale Visual Active Learning with Deep Probabilistic Ensembles. Arxiv 2018
Active Learning

Quantitative Results

How much data we need to outperform the performance using the entire dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>% data</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>~50</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>~80</td>
</tr>
<tr>
<td>SVHN</td>
<td>~25</td>
</tr>
</tbody>
</table>

[Chitta, Alvarez, Lesnikowski], Large-Scale Visual Active Learning with Deep Probabilistic Ensembles. Arxiv 2018
Beyond Classification
Active Semantic Segmentation

Framework

[Chitta, Alvarez, Lesnikowski], Large-Scale Visual Active Learning with Deep Probabilistic Ensembles. Under review
Domain Adaptation
(Beyond a single domain / location)
Domain Adaptation

- Day
- Twilight
- Night
- Artificial light
- Backlit
- Clear
- Cloudy
- Rain
- Fog
- Snow
- Urban
- Freeway
- Unmarked Street

Geographic Locations
Domain Adaptation
Domain Adaptation

4. At train time, use only (synthetic) source images and annotations.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Images</th>
<th>Annotations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>☺</td>
<td>☻</td>
</tr>
<tr>
<td>Target</td>
<td>☹</td>
<td>☹</td>
</tr>
</tbody>
</table>

Synthetic data can be obtained in large amounts and is labeled automatically.
4. At train time, use only (synthetic) source images and annotations.

Unfortunately, in general, a network trained on synthetic data performs relatively poorly on real images. Most require access to real images, albeit unsupervised, during training.
Domain Adaptation
Efficient use of Synthetic Data

Our approach uses synthetic images and does not require seeing any real images at training time.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Images</th>
<th>Annotations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>😊</td>
<td>😊</td>
</tr>
<tr>
<td>Target</td>
<td>😞</td>
<td>😞</td>
</tr>
</tbody>
</table>
Domain Adaptation

Efficient use of Synthetic Data

Our approach uses synthetic images and does not require seeing any real images at training time.

Key observation:
Foreground and background classes are not affected in the same manner by the domain shift.

[Saleh, Salzmann, Alvarez et al. 2018], Efficient use of Synthetic data for Semantic Segmentation, ECCV2018
Domain Adaptation

Efficient use of Synthetic Data

1. Texture of background classes is realistic -> semantic segmentation.

[Saleh, Salzmann, Alvarez et al. 2018], Efficient use of Synthetic data for Semantic Segmentation, ECCV2018
Domain Adaptation

Efficient use of Synthetic Data

1. Texture of background classes is realistic -> semantic segmentation.
2. Texture of foreground classes is not photo-realistic, but their shape looks natural -> detection-based.
Domain Adaptation

Efficient use of Synthetic Data

Inference on real data

[Saleh, Salzmann, Alvarez et al. 2018], Efficient use of Synthetic data for Semantic Segmentation, ECCV2018
## Domain Adaptation

### Efficient use of Synthetic Data

Table 1: **Comparison of models trained on synthetic data.** All the results are reported on the Cityscapes validation set. Note that ps-GT (pseudo-GT) indicates the use of unlabeled real images during training.

<table>
<thead>
<tr>
<th></th>
<th>road</th>
<th>side.</th>
<th>build.</th>
<th>wall</th>
<th>fence</th>
<th>pole</th>
<th>light</th>
<th>sign</th>
<th>Vege.</th>
<th>terr.</th>
<th>sky</th>
<th>person</th>
<th>rider</th>
<th>car</th>
<th>truck</th>
<th>bus</th>
<th>train</th>
<th>motor</th>
<th>bike</th>
<th>mIOU</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTA5 [5]</td>
<td>29.8</td>
<td>16.0</td>
<td>56.6</td>
<td>9.2</td>
<td>17.3</td>
<td>13.5</td>
<td>13.6</td>
<td>9.8</td>
<td>74.9</td>
<td>6.7</td>
<td>54.3</td>
<td>41.9</td>
<td>2.9</td>
<td>45.0</td>
<td>3.3</td>
<td>13.1</td>
<td>1.3</td>
<td>6.0</td>
<td>0.0</td>
<td>21.9</td>
</tr>
<tr>
<td>GTA5</td>
<td>80.5</td>
<td>26.0</td>
<td>74.7</td>
<td>23.0</td>
<td>9.8</td>
<td>9.1</td>
<td>13.4</td>
<td>7.3</td>
<td>79.4</td>
<td>28.6</td>
<td>72.1</td>
<td>40.4</td>
<td>5.1</td>
<td>77.8</td>
<td>23.0</td>
<td>18.6</td>
<td>1.2</td>
<td>5.3</td>
<td>0.0</td>
<td>31.3</td>
</tr>
<tr>
<td>SYNTHIA</td>
<td>36.7</td>
<td>22.7</td>
<td>51.0</td>
<td>0.3</td>
<td>0.1</td>
<td>16.6</td>
<td>0.1</td>
<td>9.5</td>
<td>72.5</td>
<td>0.0</td>
<td>78.4</td>
<td>47.5</td>
<td>5.6</td>
<td>61.4</td>
<td>0.0</td>
<td>13.0</td>
<td>0.0</td>
<td>3.2</td>
<td>3.1</td>
<td>22.1</td>
</tr>
<tr>
<td>VIPER</td>
<td>36.9</td>
<td>19.0</td>
<td>74.7</td>
<td>0.0</td>
<td>5.3</td>
<td>7.1</td>
<td>10.0</td>
<td>10.1</td>
<td>78.7</td>
<td>13.6</td>
<td>69.6</td>
<td>43.0</td>
<td>0.0</td>
<td>41.2</td>
<td>20.8</td>
<td>13.9</td>
<td>0.0</td>
<td>9.1</td>
<td>0.0</td>
<td>23.9</td>
</tr>
<tr>
<td>VEIS</td>
<td>70.8</td>
<td>9.5</td>
<td>50.9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
<td>15.6</td>
<td>26.8</td>
<td>66.8</td>
<td>12.7</td>
<td>52.3</td>
<td>44.0</td>
<td>14.2</td>
<td>60.6</td>
<td>10.2</td>
<td>8.2</td>
<td>3.2</td>
<td>5.5</td>
<td>11.8</td>
<td>24.4</td>
</tr>
<tr>
<td>GTA5+VEIS</td>
<td>66.2</td>
<td>21.6</td>
<td>72.3</td>
<td>15.7</td>
<td>18.3</td>
<td>12.3</td>
<td>22.3</td>
<td>23.8</td>
<td>78.4</td>
<td>11.3</td>
<td>74.6</td>
<td>48.7</td>
<td>13.3</td>
<td>75.1</td>
<td>14.3</td>
<td>21.2</td>
<td>2.1</td>
<td>24.2</td>
<td>7.3</td>
<td>32.8</td>
</tr>
<tr>
<td>GTA5+VEIS&amp;ps-GT</td>
<td>77.6</td>
<td>26.8</td>
<td>75.5</td>
<td>19.4</td>
<td>19.5</td>
<td>4.8</td>
<td>18.7</td>
<td>19.8</td>
<td>79.5</td>
<td>21.7</td>
<td>78.9</td>
<td>47.3</td>
<td>8.7</td>
<td>77.6</td>
<td>23.1</td>
<td>16.1</td>
<td>2.2</td>
<td>15.6</td>
<td>0.0</td>
<td>33.3</td>
</tr>
<tr>
<td><strong>Ours</strong></td>
<td>71.9</td>
<td>23.8</td>
<td>75.5</td>
<td>23.4</td>
<td>14.9</td>
<td>9.3</td>
<td>26.7</td>
<td>42.5</td>
<td>80.1</td>
<td>34.0</td>
<td>76.3</td>
<td>52.2</td>
<td>28.5</td>
<td>76.2</td>
<td>19.6</td>
<td>31.6</td>
<td>6.9</td>
<td>18.1</td>
<td>9.8</td>
<td>38.0</td>
</tr>
</tbody>
</table>

[Saleh, Salzmann, Alvarez et al. 2018], Efficient use of Synthetic data for Semantic Segmentation, ECCV2018
### Domain Adaptation

**Efficient use of Synthetic Data**

Adding Pseudo-labels:
*(unsupervised real training data)*

<table>
<thead>
<tr>
<th>Domain</th>
<th>Images</th>
<th>Annotations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>😊</td>
<td>😊</td>
</tr>
<tr>
<td>Target</td>
<td>😊</td>
<td>😞</td>
</tr>
</tbody>
</table>
 Domain Adaptation

Adding Pseudo-labels:

Efficient use of Synthetic Data

Comparison on models trained on synthetic data

<table>
<thead>
<tr>
<th>Methods</th>
<th>mIOU</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTA5 [5]</td>
<td>21.9</td>
</tr>
<tr>
<td>GTA5</td>
<td>31.3</td>
</tr>
<tr>
<td>SYNTHIA</td>
<td>22.1</td>
</tr>
<tr>
<td>VIPER</td>
<td>23.9</td>
</tr>
<tr>
<td>VEIS</td>
<td>24.4</td>
</tr>
<tr>
<td>GTA5 + VEIS</td>
<td>32.8</td>
</tr>
<tr>
<td>GTA5 + VEIS &amp; ps-GT</td>
<td>33.3</td>
</tr>
<tr>
<td>Ours</td>
<td>38.0</td>
</tr>
<tr>
<td>Ours &amp; ps-GT</td>
<td>42.5</td>
</tr>
</tbody>
</table>

[Saleh, Salzmann, Alvarez et al. 2018], Efficient use of Synthetic data for Semantic Segmentation, ECCV2018
Domain Adaptation

Adding Pseudo-labels:

Efficient use of Synthetic Data

Comparison to domain adaptation and weakly-supervised methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>mIOU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully Sup.</td>
<td>56.2</td>
</tr>
<tr>
<td>Weakly-Sup.[2]</td>
<td>23.6</td>
</tr>
<tr>
<td>FCNs in Wld [3]</td>
<td>27.1</td>
</tr>
<tr>
<td>ROAD [5]</td>
<td>35.9</td>
</tr>
<tr>
<td>CYCADA [6]</td>
<td>35.4</td>
</tr>
<tr>
<td>Ours</td>
<td>38.0</td>
</tr>
<tr>
<td>Ours + Pseudo-GT</td>
<td>42.5</td>
</tr>
</tbody>
</table>

[Saleh, Salzmann, Alvarez et al. 2018], Efficient use of Synthetic data for Semantic Segmentation, ECCV2018
Accuracy vs Efficiency (for Large datasets)
Accuracy vs Efficiency
Accuracy vs Efficiency

Efficient Training of DNN

Goal: maximize training resources while obtaining deployment ‘friendly’ network.
Over-parameterization
Accuracy vs Efficiency

5x5 convolution

Same receptive field

two successive
3x3 convolutions

Non-linearity

Capacity

Num. parameters
Accuracy vs Efficiency

Validation Accuracy on a 3x3-based Convnet (orange) and the equivalent 5x5-based Convnet (blue)

https://blog.sicara.com/about-convolutional-layer-convolution-kernel-9a7325d34f7d
Accuracy vs Efficiency

5x5 convolution

Same receptive field

n x n as [1 x n] and [n x 1]

two successive 3x3 convolutions

Non-linearity

Non-linearity

Capacity

Num. parameters

FLOPS

??
Accuracy vs Efficiency
Filter Decompositions for Real-time Semantic Segmentation

Accuracy vs Efficiency

Filter Decompositions for Real-time Semantic Segmentation

Cityscapes dataset (19 classes, 7 categories)

<table>
<thead>
<tr>
<th>Train mode</th>
<th>Pixel accuracy</th>
<th>Class IoU</th>
<th>Category IoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scratch</td>
<td>94.7 %</td>
<td>70.0 %</td>
<td>86.0 %</td>
</tr>
<tr>
<td>Pre-trained</td>
<td>95.1 %</td>
<td>71.5 %</td>
<td>86.9 %</td>
</tr>
</tbody>
</table>

Forward-Time: Cityscapes 19 classes

<table>
<thead>
<tr>
<th></th>
<th>TEGRA-TX1</th>
<th>TITAN-X</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fwd Pass</td>
<td>512x256</td>
<td>1024x512</td>
</tr>
<tr>
<td>Time</td>
<td>85 ms</td>
<td>310 ms</td>
</tr>
<tr>
<td>FPS</td>
<td>11.8</td>
<td>3.2</td>
</tr>
</tbody>
</table>

Accuracy vs Efficiency

Accuracy vs Efficiency

Efficient Training of DNN

Goal: maximize training resources while obtaining deployment ‘friendly’ network.
Accuracy vs Efficiency

Efficient Training of DNN

Goal: maximize training resources while obtaining deployment ‘friendly’ network.
Accuracy vs Efficiency

Common Approach

Train a large model (trade-off accuracy / computational cost)

TRAIN
Promising model

Prune / Optimize
For a specific application

DEPLOY
Optimize for Specific hardware

Regularization at parameter level
Accuracy vs Efficiency

Joint Training and Pruning Deep Networks

Train a large model (trade-off accuracy / computational cost)
Accuracy vs Efficiency

Joint Training and Pruning Deep Networks

Convolutional layer 5x1x3x3

Removed

To be kept
Accuracy vs Efficiency

Joint Training and Pruning Deep Networks

Common approach:

\[
\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i, \Theta)) + \frac{\rho}{2} \|\Theta\|_2^2
\]

- Neuron “Convolutional kernel”
- Weight Decay (prevent weights with large values)
- Considers each parameter independently

References:
[Alvarez and Salzmann], Learning the number of neurons in Neural Nets, NIPS 2016
[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017
Accuracy vs Efficiency

Joint Training and Pruning Deep Networks

Our Approach:

\[
\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i, \Theta)) + \frac{\rho}{2} \|\Theta\|_2^2 + r(\Theta).
\]

[Alvarez and Salzmann], Learning the number of neurons in Neural Nets, NIPS 2016
[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017
Classification Results
Accuracy vs Efficiency

Joint Training and Pruning Deep Networks

Quantitative Results on ImageNet dataset:
1.2 million training images and 50.000 for validation split in 1000 categories.
Between 5000 and 30000 training images per class.
No data augmentation (random flip).

[Alvarez and Salzmann], Learning the number of neurons in Neural Nets, NIPS 2016
[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017
Accuracy vs Efficiency

Joint Training and Pruning Deep Networks

Quantitative Results on ImageNet

Train an over-parameterized architecture up to 768 neurons per layer ($Dec_8\cdot768$)

<table>
<thead>
<tr>
<th>Dec1</th>
<th>Dec2</th>
<th>Dec3</th>
<th>Dec4</th>
<th>Dec5</th>
<th>Dec6</th>
<th>Dec7</th>
<th>Dec8</th>
<th>FC</th>
</tr>
</thead>
<tbody>
<tr>
<td>32-64</td>
<td>128-256</td>
<td>384-384</td>
<td>512-512</td>
<td>768-768</td>
<td>768-768</td>
<td>768-768</td>
<td>768-768</td>
<td>1000</td>
</tr>
</tbody>
</table>

[Alvarez and Salzmann], Learning the number of neurons in Neural Nets, NIPS 2016
[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017
Accuracy vs Efficiency

Joint Training and Pruning Deep Networks

Quantitative Results on ImageNet

![ImageNet](image.png)

<table>
<thead>
<tr>
<th>Dec8 on ImageNet (in %)</th>
<th>(\text{Dec}_8)</th>
<th>(\text{Dec}_8)-640</th>
<th>(\text{Dec}_8)-768</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS</td>
<td>3.39</td>
<td>12.42</td>
<td>26.83</td>
</tr>
<tr>
<td>SGL</td>
<td>2.46</td>
<td>13.69</td>
<td>31.53</td>
</tr>
<tr>
<td>GS</td>
<td>2.46</td>
<td>22.72</td>
<td>31.63</td>
</tr>
<tr>
<td>neurons</td>
<td>2.82</td>
<td>23.33</td>
<td>32.26</td>
</tr>
<tr>
<td>group param</td>
<td>0.01</td>
<td>0.94</td>
<td>-0.02</td>
</tr>
<tr>
<td>total param</td>
<td>2.46</td>
<td>12.42</td>
<td>26.83</td>
</tr>
<tr>
<td>total induced</td>
<td>2.46</td>
<td>13.69</td>
<td>31.53</td>
</tr>
<tr>
<td>accuracy gap</td>
<td>2.46</td>
<td>22.72</td>
<td>31.63</td>
</tr>
</tbody>
</table>

[Alvarez and Salzmann], Learning the number of neurons in Neural Nets, NIPS 2016
[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017
Accuracy vs Efficiency

Joint Training and Pruning Deep Networks

Quantitative Results on ICDAR character recognition dataset

[Alvarez and Salzmann], Learning the number of neurons in Neural Nets, NIPS 2016
[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017
Accuracy vs Efficiency
Joint Training and Pruning Deep Networks

Quantitative Results on ICDAR character recognition dataset

Train an over-parameterized architecture up to 512 neurons per layer (Dec₃-512)

```
Dec1  | Dec2  | Dec3  | FC
48-96 | 160-256 | 512-512 | 36
```

[Alvarez and Salzmann], Learning the number of neurons in Neural Nets, NIPS 2016
[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017
Accuracy vs Efficiency

Joint Training and Pruning Deep Networks

Quantitative Results on ICDAR character recognition dataset

<table>
<thead>
<tr>
<th></th>
<th>Dec3 on ICDAR (in %)</th>
<th>Top-1 acc. on ICDAR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S-GS</td>
<td>GS</td>
</tr>
<tr>
<td>neurons</td>
<td>38.64</td>
<td>55.11</td>
</tr>
<tr>
<td>group param</td>
<td>32.57</td>
<td>66.48</td>
</tr>
<tr>
<td>total param</td>
<td>72.41</td>
<td>66.48</td>
</tr>
<tr>
<td>total induced</td>
<td>72.08</td>
<td>66.52</td>
</tr>
<tr>
<td>accuracy gap</td>
<td>1.24</td>
<td>1.38</td>
</tr>
<tr>
<td>MaxOut$_{Dec}$</td>
<td></td>
<td>91.3%</td>
</tr>
<tr>
<td>MaxOut$^b$</td>
<td></td>
<td>89.8%</td>
</tr>
<tr>
<td>MaxPool$_{2Dneurons}$</td>
<td></td>
<td>83.8%</td>
</tr>
<tr>
<td>Dec3 (baseline)</td>
<td></td>
<td>89.3%</td>
</tr>
<tr>
<td>Ours-Dec$^3$-SGL</td>
<td></td>
<td>89.9%</td>
</tr>
<tr>
<td>Ours-Dec$^3$-GS</td>
<td></td>
<td>90.1%</td>
</tr>
</tbody>
</table>

$^a$ Results from Jaderberg et al. [2014a] using MaxOut layer instead of Max-Pooling and decompositions as post-processing step

$^b$ Results from Jaderberg et al. [2014a]

[Alvarez and Salzmann], Learning the number of neurons in Neural Nets, NIPS 2016
[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017
Accuracy vs Efficiency

Joint Training and Pruning Deep Networks

[Alvarez and Salzmann], Learning the number of neurons in Neural Nets, NIPS 2016
[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017
Accuracy vs Efficiency

Dec1 Dec2 Dec3 Dec4 Dec5 Dec6 Dec7 Dec7-1 Dec7-2 Dec8 Dec8-1 Dec8-2 FC

Initial number
Learned number

(No drop in accuracy)
Object Detection Results
### Accuracy vs Efficiency

**Object Detection**

#### Promising model

- **Train**

#### For a specific application

- **Prune / Optimize**

<table>
<thead>
<tr>
<th>model</th>
<th>weighted</th>
<th>hard</th>
<th>easy</th>
<th>mdrt</th>
<th>weighted</th>
<th>hard</th>
<th>easy</th>
<th>mdrt</th>
<th>weighted</th>
<th>hard</th>
<th>easy</th>
<th>mdrt</th>
<th>Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>model</td>
<td>75.03</td>
<td>69.59</td>
<td>84.45</td>
<td>77.37</td>
<td>28.42</td>
<td>21.95</td>
<td>26.78</td>
<td>21.72</td>
<td>23.94</td>
<td>18.89</td>
<td>19.02</td>
<td>19.95</td>
<td>11,022,095</td>
</tr>
<tr>
<td>model_p_0</td>
<td>70.83</td>
<td>66.89</td>
<td>87.30</td>
<td>76.50</td>
<td>11.40</td>
<td>10.23</td>
<td>13.83</td>
<td>10.56</td>
<td>37.16</td>
<td>29.30</td>
<td>33.42</td>
<td>32.68</td>
<td>11,022,095</td>
</tr>
<tr>
<td>model_p_1</td>
<td>81.89</td>
<td>80.36</td>
<td>92.39</td>
<td>88.61</td>
<td>27.11</td>
<td>23.32</td>
<td>26.62</td>
<td>22.44</td>
<td>54.25</td>
<td>45.08</td>
<td>53.40</td>
<td>50.22</td>
<td>9,125,417</td>
</tr>
<tr>
<td>model_p_2</td>
<td>83.50</td>
<td>82.07</td>
<td>91.99</td>
<td>89.47</td>
<td>39.37</td>
<td>35.81</td>
<td>43.17</td>
<td>35.72</td>
<td>62.25</td>
<td>51.93</td>
<td>62.98</td>
<td>57.73</td>
<td>1,664,987</td>
</tr>
<tr>
<td>model_p_3</td>
<td>83.32</td>
<td>82.62</td>
<td>92.45</td>
<td>89.96</td>
<td>48.23</td>
<td>45.07</td>
<td>56.91</td>
<td>45.13</td>
<td>63.70</td>
<td>53.49</td>
<td>64.40</td>
<td>59.09</td>
<td>576,746</td>
</tr>
<tr>
<td>model_p_4</td>
<td>83.50</td>
<td>82.78</td>
<td>92.67</td>
<td>89.89</td>
<td>51.92</td>
<td>48.18</td>
<td>62.21</td>
<td>49.31</td>
<td>65.33</td>
<td>54.93</td>
<td>66.17</td>
<td>60.39</td>
<td>407,856</td>
</tr>
<tr>
<td>model_p_5</td>
<td>83.64</td>
<td>82.78</td>
<td>92.56</td>
<td>89.91</td>
<td>52.39</td>
<td>49.91</td>
<td>62.73</td>
<td>50.66</td>
<td>66.44</td>
<td>56.24</td>
<td>67.34</td>
<td>61.69</td>
<td>332,454</td>
</tr>
<tr>
<td>model_p_6</td>
<td>83.86</td>
<td>82.65</td>
<td>92.55</td>
<td>90.11</td>
<td>52.21</td>
<td>48.34</td>
<td>61.63</td>
<td>49.07</td>
<td>67.02</td>
<td>56.85</td>
<td>68.71</td>
<td>62.70</td>
<td>310,016</td>
</tr>
<tr>
<td>model_p_7</td>
<td>84.23</td>
<td>83.14</td>
<td>92.83</td>
<td>90.32</td>
<td>52.42</td>
<td>48.56</td>
<td>61.27</td>
<td>49.74</td>
<td>68.77</td>
<td>58.99</td>
<td>71.68</td>
<td>64.83</td>
<td>300,543</td>
</tr>
<tr>
<td>model_p_8</td>
<td>84.22</td>
<td>83.14</td>
<td>92.20</td>
<td>90.31</td>
<td>51.97</td>
<td>47.90</td>
<td>61.73</td>
<td>48.69</td>
<td>67.89</td>
<td>57.92</td>
<td>70.22</td>
<td>63.70</td>
<td>292,217</td>
</tr>
<tr>
<td>model_p_9</td>
<td>83.74</td>
<td>82.96</td>
<td>92.29</td>
<td>90.27</td>
<td>51.59</td>
<td>47.56</td>
<td>61.32</td>
<td>48.68</td>
<td>68.19</td>
<td>58.32</td>
<td>71.06</td>
<td>64.07</td>
<td>283,116</td>
</tr>
</tbody>
</table>

*KITTl*
### Accuracy vs Efficiency

**Object Detection**

<table>
<thead>
<tr>
<th>model</th>
<th>car</th>
<th>cyclist</th>
<th>pedestrian</th>
<th>params</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>weighted</td>
<td>hard</td>
<td>easy</td>
<td>mdrt</td>
</tr>
<tr>
<td>Baseline (L1) model</td>
<td>83.74</td>
<td>82.96</td>
<td>92.29</td>
<td>90.27</td>
</tr>
<tr>
<td>0.0005 (GS) model</td>
<td>85.22</td>
<td>84.67</td>
<td>92.81</td>
<td>91.81</td>
</tr>
</tbody>
</table>

**TRAIN**

**Prune / Optimize**

**Joint Train / Pruning**

**KITTI**
Accuracy vs Efficiency

Compression-aware Training of DNN

Convolutional layer
5x1x3x3

Removed
To be kept

[Alvarez and Salzmann], Learning the number of neurons in Neural Nets, NIPS 2016
[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017
Accuracy vs Efficiency

Compression-aware Training of DNN

Uncorrelated filters should maximize the use of each parameter / kernel

Cross-correlation of Gabor Filters.
Accuracy vs Efficiency

Compression-aware Training of DNN

Weak-Points

- Significantly larger training time (prohibitive at large scale).
- Usually drops in accuracy.
- **Orthogonal filters are difficult to compress** (post-processing).

[P Rodríguez, J González, G Cucurull, J. M. Gonfaus, X Roca] Regularizing CNNs with Locally Constrained Decorrelations. ICLR 2017
Accuracy vs Efficiency

Compression-aware Training of DNN

Convolutional layer 5x1x3x3

Removed
To be kept
Accuracy vs Efficiency

Compression-aware Training of DNN

[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017
Accuracy vs Efficiency

Compression-aware Training of DNN

Our Approach:

\[
\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i, \Theta)) + \frac{\rho}{2} ||\Theta||_2^2 + r(\Theta) + h(\Theta),
\]

Considers each Layer independently

Kernel Similarity at layer level

[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017
Classification Results
Accuracy vs Efficiency

Compression-aware Training of DNN

Quantitative Results on ImageNet using ResNet50*

<table>
<thead>
<tr>
<th></th>
<th>top-1</th>
<th>Params.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Resnet-50</strong>, $e_l = 90%$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Baseline</td>
<td>74.7</td>
<td>18M</td>
</tr>
<tr>
<td>[14]</td>
<td>74.0</td>
<td>-4.0%</td>
</tr>
<tr>
<td>Group-sparse [2]</td>
<td>74.5</td>
<td>-17.0%</td>
</tr>
<tr>
<td><strong>Ours (low-rank)</strong></td>
<td>75.0</td>
<td>-20.6%</td>
</tr>
<tr>
<td>Low-rank + group-sparse</td>
<td>75.2</td>
<td>-27.0%</td>
</tr>
</tbody>
</table>

*Modified to use 1D kernels.

[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017
Training Efficient
(side benefit)
Accuracy vs Efficiency

Compression-aware Training of DNN

[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017
Accuracy vs Efficiency

Compression-aware Training of DNN

[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017
Accuracy vs Efficiency

Compression-aware Training of DNN

<table>
<thead>
<tr>
<th>Epoch reload</th>
<th>Num. parameters Total</th>
<th>Num. parameters no SVD</th>
<th>accuracy top-1</th>
<th>Total train-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>3.7M</td>
<td>–</td>
<td>88.4%</td>
<td>1.69h</td>
</tr>
<tr>
<td>r5</td>
<td>3.2M</td>
<td>3.71M</td>
<td>89.8%</td>
<td>1.81h</td>
</tr>
<tr>
<td>r15</td>
<td>210K</td>
<td>2.08M</td>
<td>90.0%</td>
<td>0.77h</td>
</tr>
<tr>
<td>r25</td>
<td>218K</td>
<td>1.60M</td>
<td>90.0%</td>
<td>0.88h</td>
</tr>
<tr>
<td>r35</td>
<td>222K</td>
<td>1.52M</td>
<td>89.0%</td>
<td>0.99h</td>
</tr>
<tr>
<td>r45</td>
<td>324K</td>
<td>1.24M</td>
<td>90.1%</td>
<td>1.12h</td>
</tr>
<tr>
<td>r55</td>
<td>388K</td>
<td>1.24M</td>
<td>89.2%</td>
<td>1.26h</td>
</tr>
<tr>
<td>r65</td>
<td>414K</td>
<td>1.23M</td>
<td>87.7%</td>
<td>1.36h</td>
</tr>
</tbody>
</table>

Up to 70% train speed-up
(similar accuracy)

[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017
Accuracy vs Efficiency

Compression-aware Training of DNN

Is Over-parameterization needed?

<table>
<thead>
<tr>
<th></th>
<th>#Epochs</th>
<th>Training</th>
<th>Test</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>top-1</td>
<td>top-5</td>
<td>Total</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>Reload</td>
<td></td>
<td></td>
</tr>
<tr>
<td>baseline</td>
<td>75</td>
<td>–</td>
<td>99.73%</td>
<td>99.96%</td>
</tr>
<tr>
<td>Ours</td>
<td>75</td>
<td>55</td>
<td>97.71%</td>
<td>99.62%</td>
</tr>
<tr>
<td>Compact</td>
<td>75</td>
<td>0</td>
<td>98.24%</td>
<td>99.76%</td>
</tr>
</tbody>
</table>

Observations:

- Additional training parameters are needed to initially help the optimizer.
- Small models are explicitly constrained, same training regime may not be fair.
- Other optimizers lead to slightly better results in optimizing compact networks from scratch.

[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017
Accuracy vs Efficiency

Compression-aware Training of DNN

Number of parameters **decreases**
Number of layers **increases**

Data Movements may be more significant than current savings.

[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017
Accuracy vs Efficiency
(more on over-parameterization)
Accuracy vs Efficiency

5x5 convolution

Same receptive field

two successive 3x3 convolutions

Non-linearity

Capacity
Num. parameters
Num. layers

91
ExpandNets
Exploiting Linear Redundancies
ExpandNets

[Guo, Alvarez, Salzmann], ExpandNets: Exploiting Linear Redundancy to Train Small Networks. Arxiv 2018
ExpandNets: Exploiting Linear Redundancy to Train Small Networks.

[Guo, Alvarez, Salzmann], Arxiv 2018
ExpandNets

[Guo, Alvarez, Salzmann], ExpandNets: Exploiting Linear Redundancy to Train Small Networks. Arxiv 2018
Classification Results
ExpandNets

[Guo, Alvarez, Salzmann], ExpandNets: Exploiting Linear Redundancy to Train Small Networks. Arxiv 2018

<table>
<thead>
<tr>
<th>N</th>
<th>ImageNet</th>
<th>Baseline</th>
<th>Expanded</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>46.72%</td>
<td>49.66%</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>54.08%</td>
<td>55.46%</td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>58.35%</td>
<td>58.75%</td>
<td></td>
</tr>
</tbody>
</table>
ExpandNets

MobileNetV2: The Next Generation of On-Device Computer Vision Networks

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-1</th>
<th>Top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNetV2</td>
<td>70.78%</td>
<td>91.47%</td>
</tr>
<tr>
<td>MobileNetV2- expanded</td>
<td>74.85%</td>
<td>92.15%</td>
</tr>
</tbody>
</table>

[Guo, Alvarez, Salzmann], ExpandNets: Exploiting Linear Redundancy to Train Small Networks. Arxiv 2018
ExpandNets

MobileNetV2: The Next Generation of On-Device Computer Vision Networks

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-1</th>
<th>Top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNetV2</td>
<td>70.78%</td>
<td>91.47%</td>
</tr>
<tr>
<td>MobileNetV2- expanded</td>
<td>74.85%</td>
<td>92.15%</td>
</tr>
<tr>
<td>MobileNetV2- expanded-nonlinear</td>
<td>74.17%</td>
<td>91.61%</td>
</tr>
<tr>
<td>MobileNetV2- expanded (nonlinear Init)</td>
<td>75.46%</td>
<td>92.58%</td>
</tr>
</tbody>
</table>

[Guo, Alvarez, Salzmann], ExpandNets: Exploiting Linear Redundancy to Train Small Networks. Arxiv 2018
ExpandNet beyond classification
ExpandNets on Semantic Segmentation

CITYSCAPES

Relative ~2.2% improvement on mIoU

[Guo, Alvarez, Salzmann], ExpandNets: Exploiting Linear Redundancy to Train Small Networks. Arxiv 2018
ExpandNets on Traffic Sign Recognition

Internal Dataset

Relative $\sim 2.34\%$ improvement on f-score

Thanks Ian Ivanecky!

[Guo, Alvarez, Salzmann], ExpandNets: Exploiting Linear Redundancy to Train Small Networks. Arxiv 2018
Summary
Summary

Creating the right datasets

• Active Learning: Our Deep Probabilistic Ensembles achieve competitive performance using 1/4th of the training data (progressively selected).
Creating the right datasets

• Synthetic to real
Summary

Creating the right datasets

**Accuracy vs Efficiency** *(aka, the use of overparameterization)*

- Joint train and prune
Summary

Creating the right datasets

**Accuracy vs Efficiency** (aka, *the use of overparameterization*)

- ExpandNets: Exploiting linear redundancy to Train Small Nets
Scaling-Up Deep Learning For Autonomous Vehicles

JOSE M. ALVAREZ | NVIDIA | GPU TECHNOLOGY CONFERENCE | San Jose 2019