# *Scaling-Up Deep Learning For Autonomous Vehicles*

JOSE M. ALVAREZ | NVIDIA | GPU TECHNOLOGY CONFERENCE

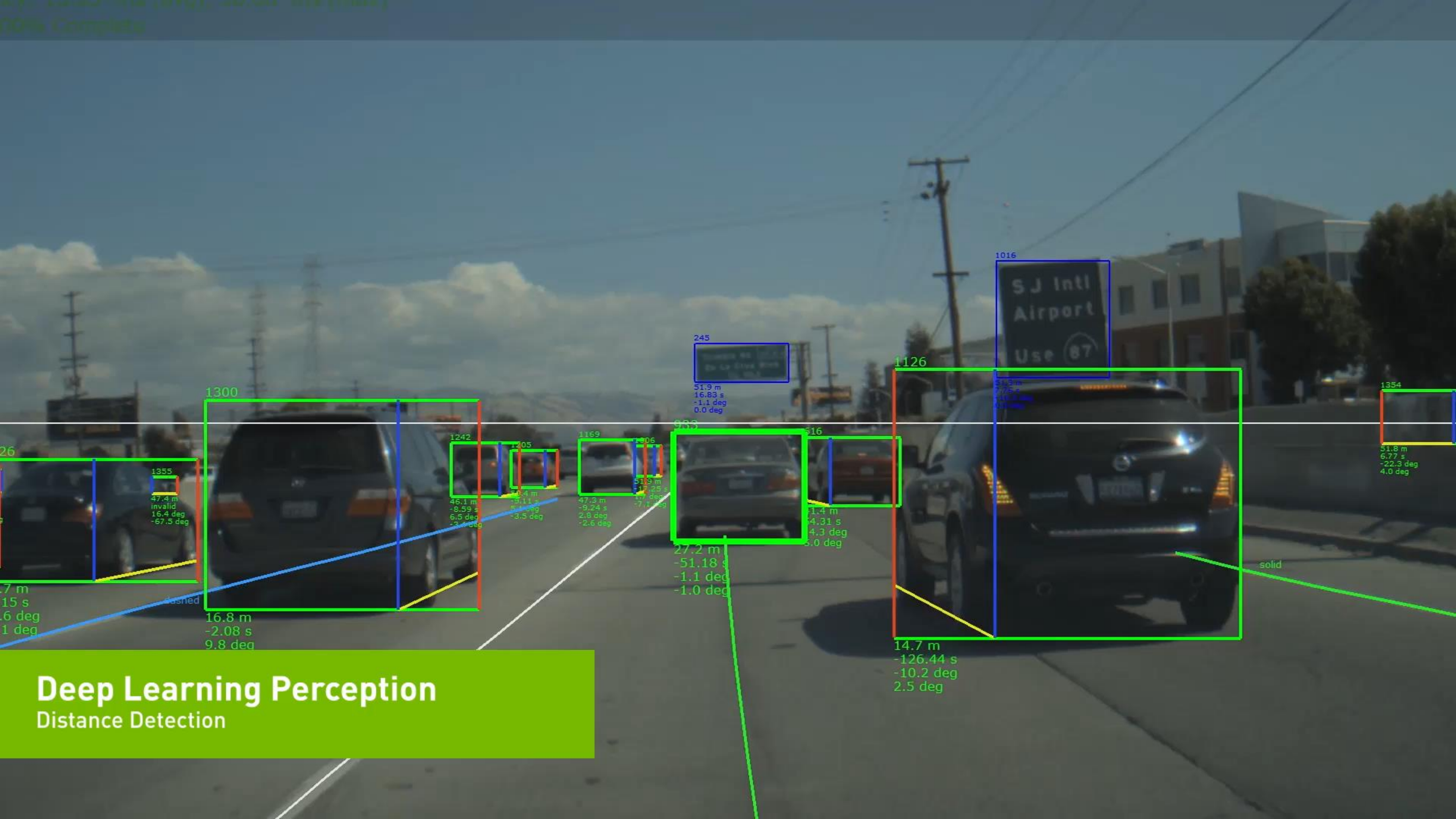
San Jose 2019

# NVIDIA
# AI-Infra

# AI-Infra Team

## One of our top Goals

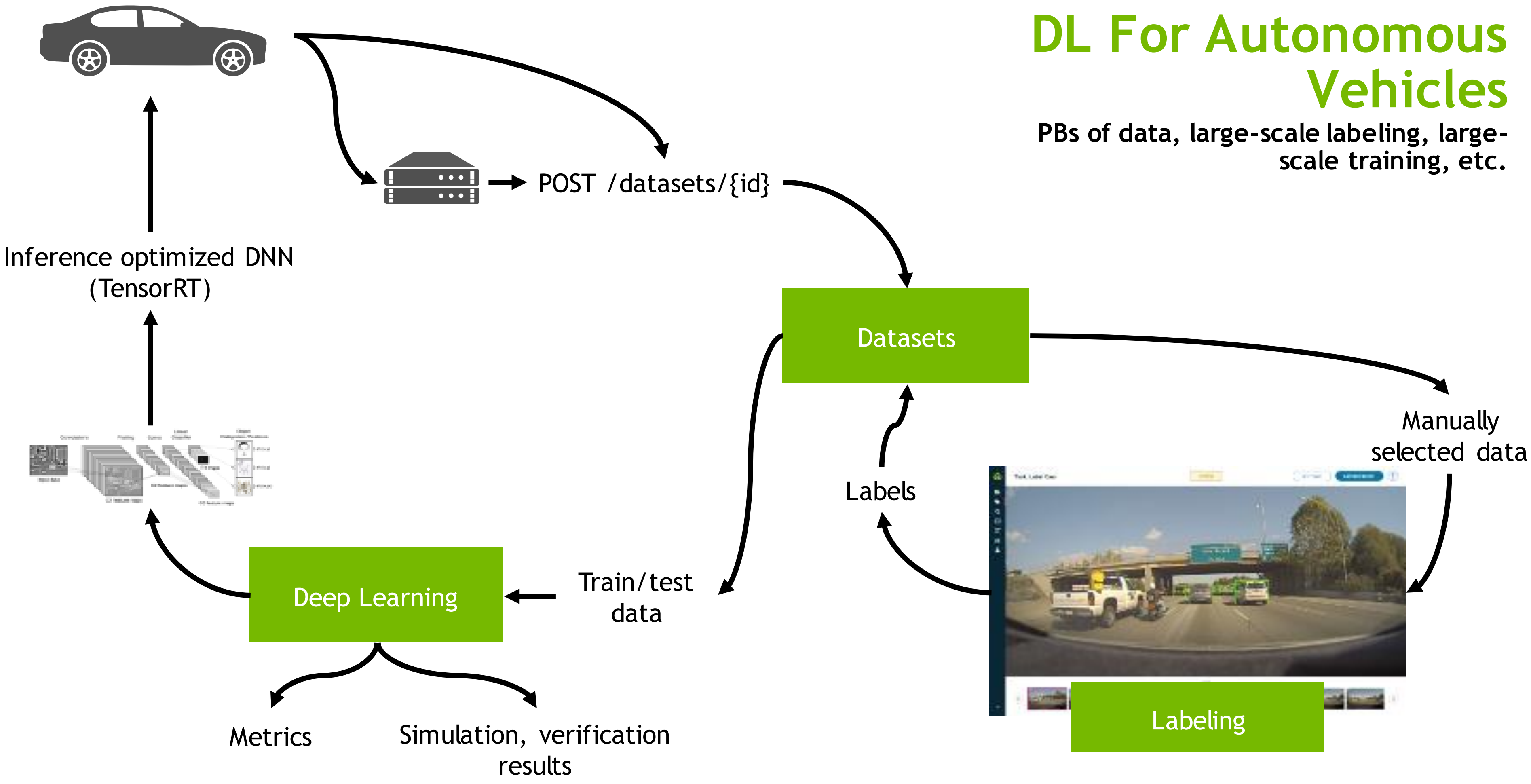
Industry grade Deep Learning to take **AV Perception DNN** into production, tested in multiple locations and conditions.



Deep Learning Perception
Distance Detection
S J Intl
Airport
Use 87
1016
245
51.9 m
16.83 s
-1.1 deg
0.0 deg
1300
16.8 m
-2.08 s
9.8 deg
1126
14.7 m
-126.44 s
-10.2 deg
2.5 deg
27.2 m
-51.18 s
-1.1 deg
-1.0 deg
1354
51.8 m
6.77 s
-22.3 deg
4.0 deg
1355
47.4 m
invalid
16.4 deg
-67.5 deg
1242
46.1 m
-8.59 s
6.5 deg
1169
47.3 m
-9.24 s
2.8 deg
-2.6 deg
solid
dashed

# DL For Autonomous Vehicles

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



# 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

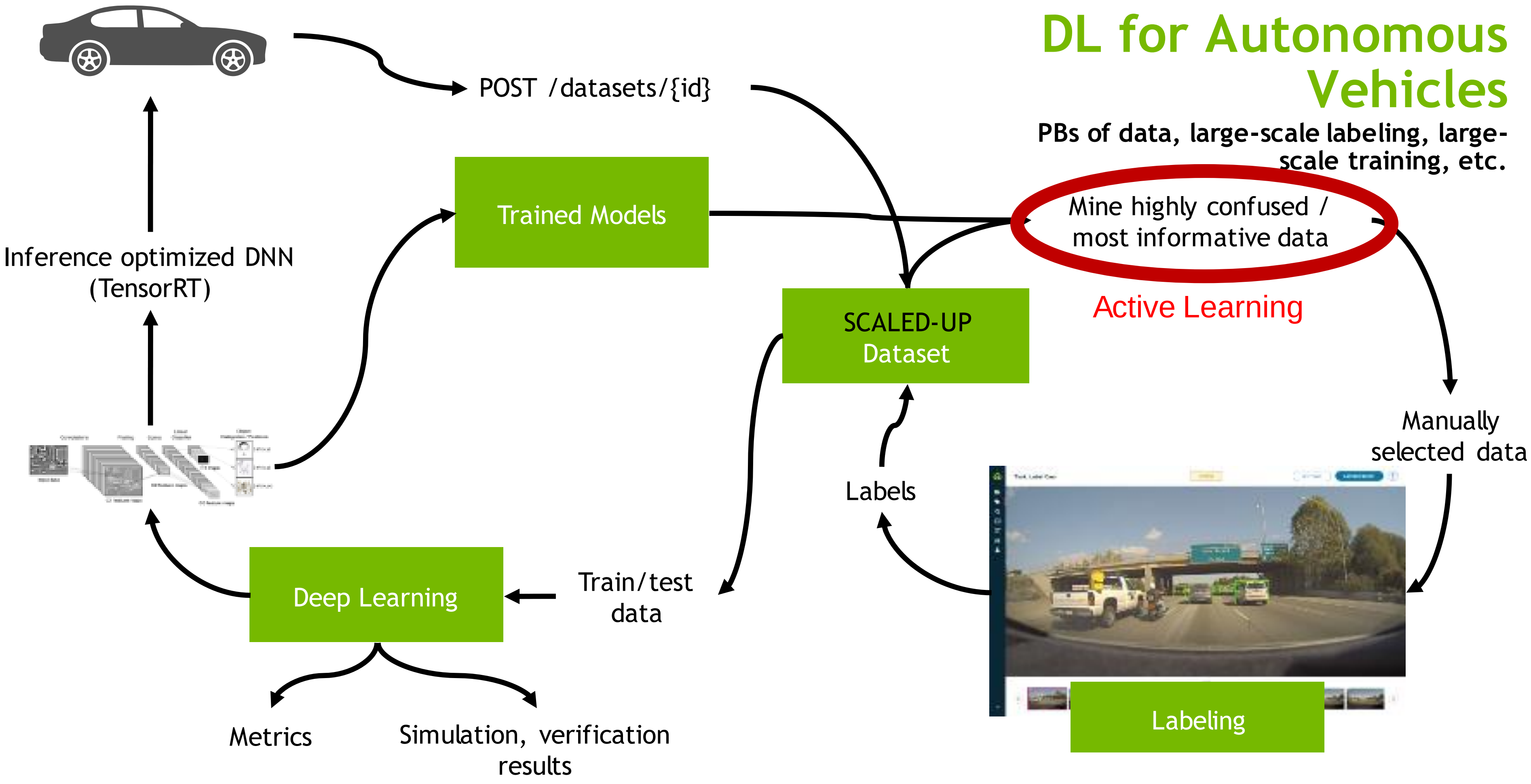
Assuming 5 2MP cameras per car, radar data, etc. => 1 TB / h / car

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.**



# DL for Autonomous Vehicles

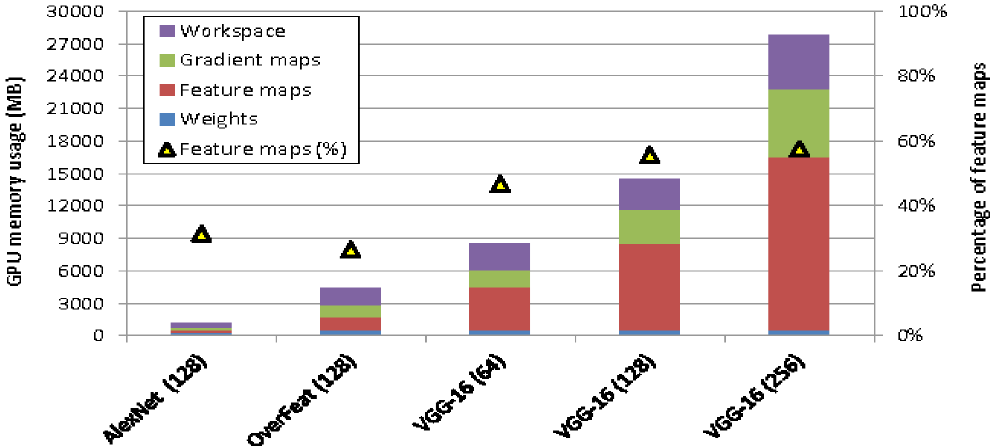
## The Challenge of Scale

Large Datasets:

**12.1 years training a ResNet50-like network on Pascal**

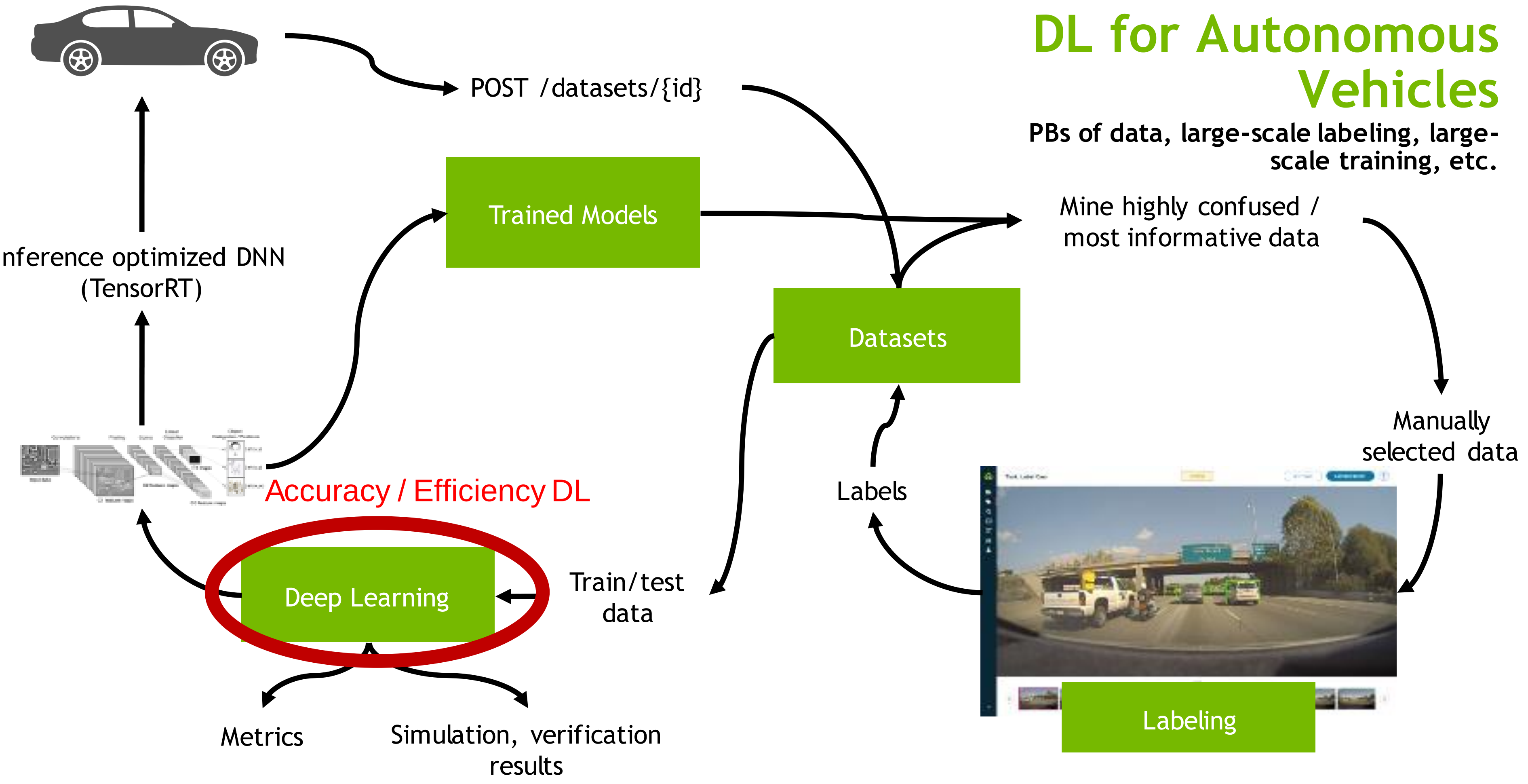
**1.5 years on DGX1 w/ Volta**

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



# DL for Autonomous Vehicles

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



# 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…

# DL for Autonomous Vehicles

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

POST /datasets/{id}

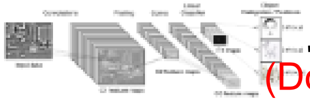
Trained Models

Mine highly confused / most informative data

Inference optimized DNN (TensorRT)

Datasets

Manually selected data



Robustness: (Domain Adaptation,...)

Labels

Deep Learning

Train/test data

Labeling

Metrics

Simulation, verification results

# 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

vs

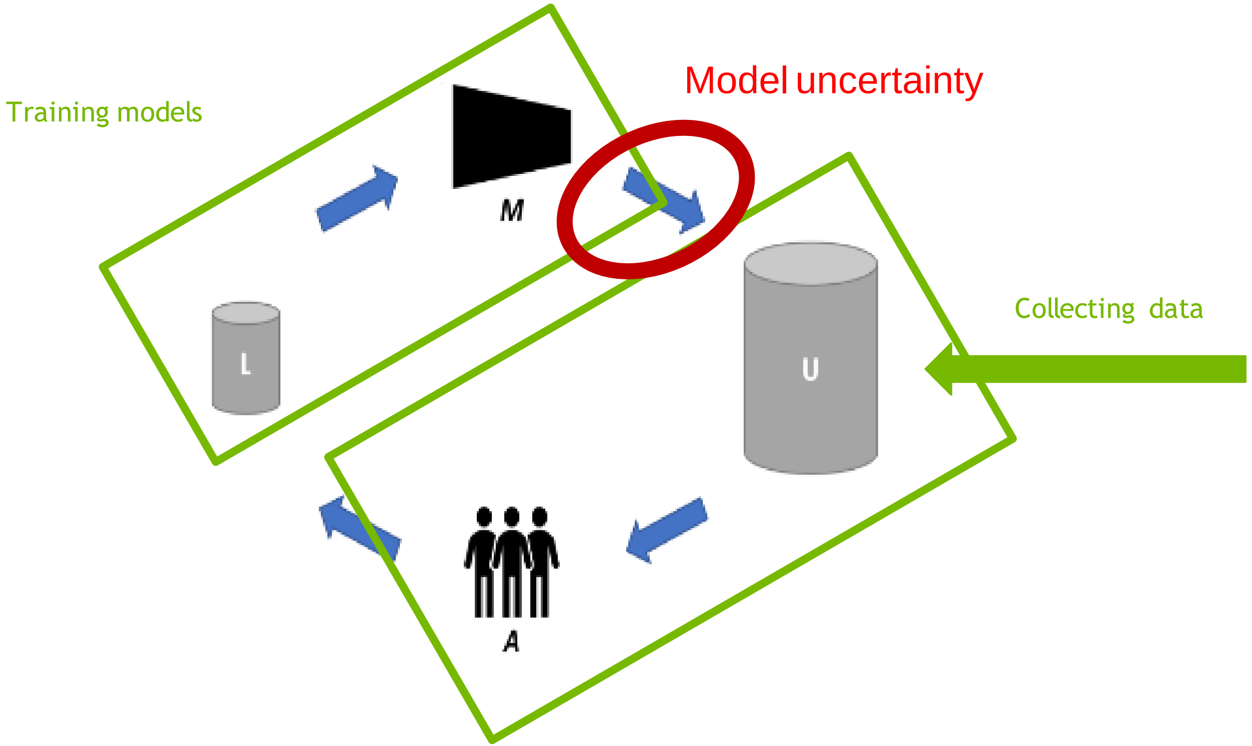

Some Samples Are Much More Informative Than Others

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

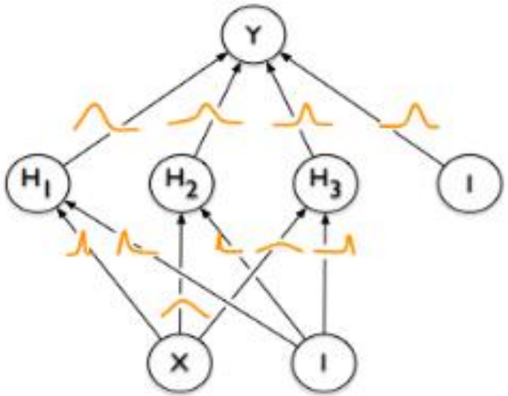


# Active Learning needs uncertainty

## Bayesian Deep Networks (BNN)

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:

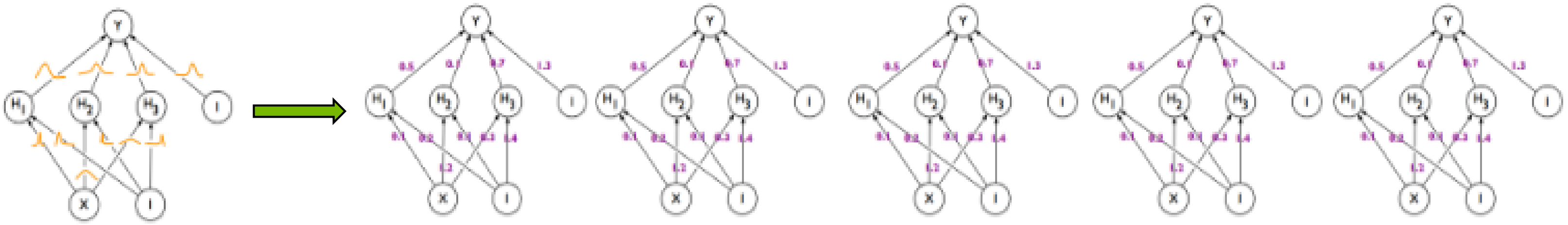
- Samples from the same distribution as the training set will have consensus while other samples will not.
- Ensembles do not approximate uncertainty in the same manner as a BNN.
  - I.e., parameters in different members serve for different purpose.

# 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.

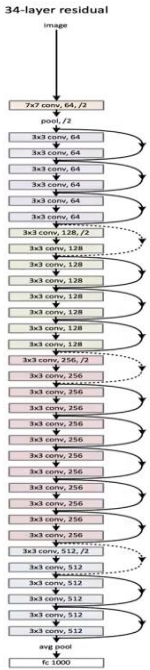
**Label those where the model is more uncertain.**

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

# Classification Results

# Active Learning

## Quantitative Results



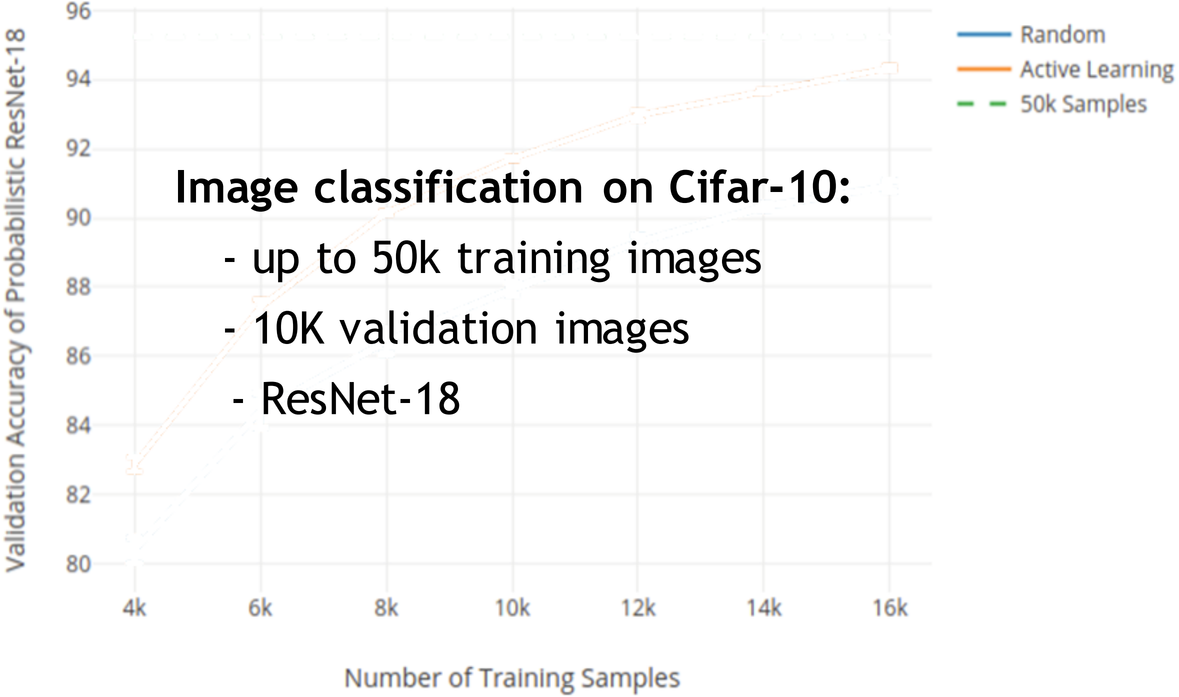



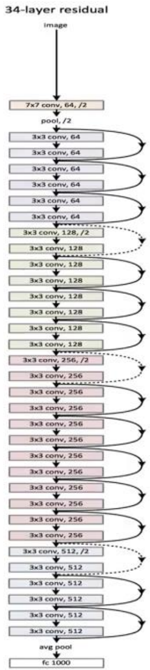

**Image classification on Cifar-10:**

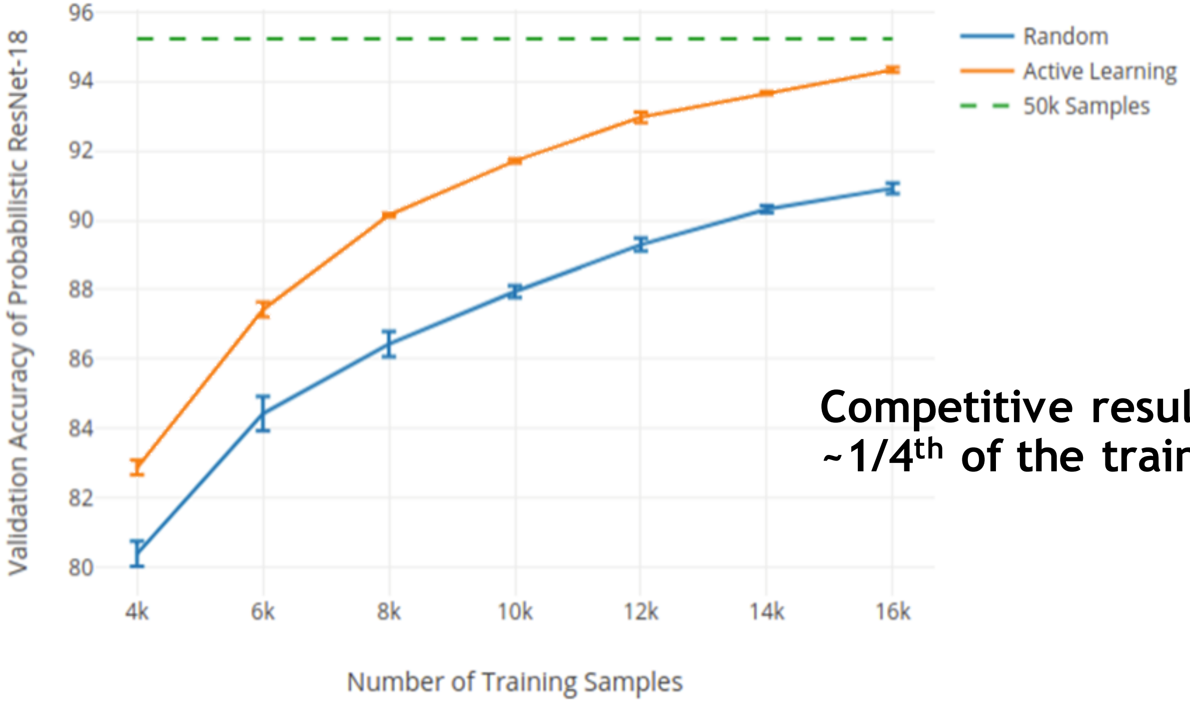
- up to 50k training images
- 10K validation images
- ResNet-18

[*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

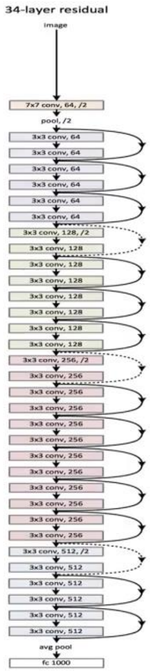


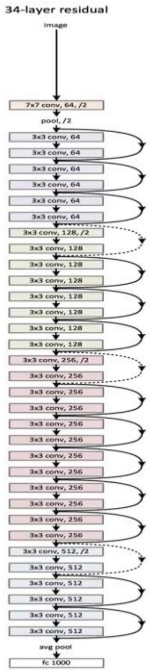

Table 2. Validation Accuracies comparing the proposed approach to standard ensembling. Initial 4% is randomly sampled.

| Task | Data Sampling | 8% | 16% | 32% |
|---|---|---|---|---|
| CIFAR-10 | Random | 80.60 | 86.80 | 91.08 |
| | Standard | 82.41 | 90.05 | 94.13 |
| | Ours | **82.88** | **90.15** | **94.33** |
| CIFAR-100 | Random | 39.57 | 54.92 | 66.65 |
| | Standard | 40.49 | 56.89 | 69.68 |
| | Ours | **40.87** | **56.94** | **70.12** |

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

# Active Learning

## Quantitative Results



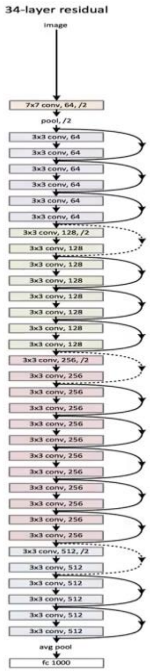

**CIFAR-10**

| Method | 10k (20%) | 50k (100%) | Ratio |
|---|---|---|---|
| Core-set [43] | 74 | 90 | 82.2 |
| Ensemble [2] | 85 | **95.5** | 89 |
| Single + Random | 85.2 | 94.4 | 90.3 |
| DPE + Random | 87.9 | 95.2 | 92.3 |
| Single + Linear-8 | 87.5 | 94.4 | 92.7 |
| **Ours** (DPE + Linear-8) | **92** | 95.2 | **96.3** |

[*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.

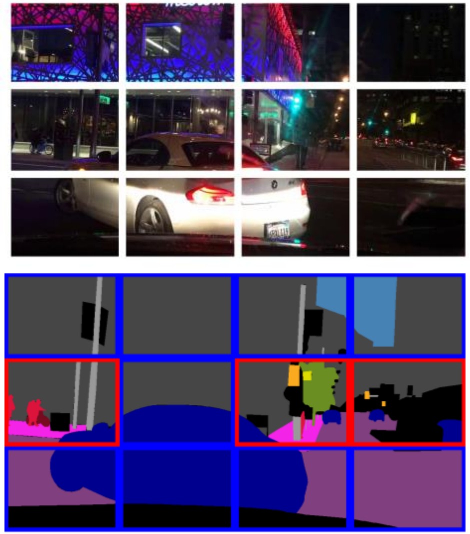
| Dataset | % data |
|---:|:---:|
| CIFAR-10 | ~50 |
| CIFAR-100 | ~80 |
| SVHN | ~25 |

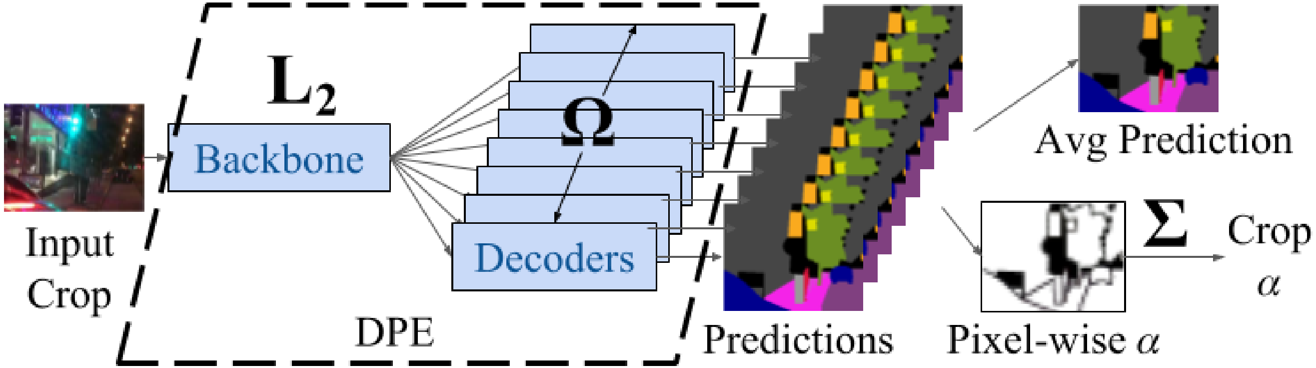
[*Chitta, Alvarez, Lesnikowski*], Large-Scale Visual Active Learning with Deep Probabilistic Ensembles. Arxiv 2018

# 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

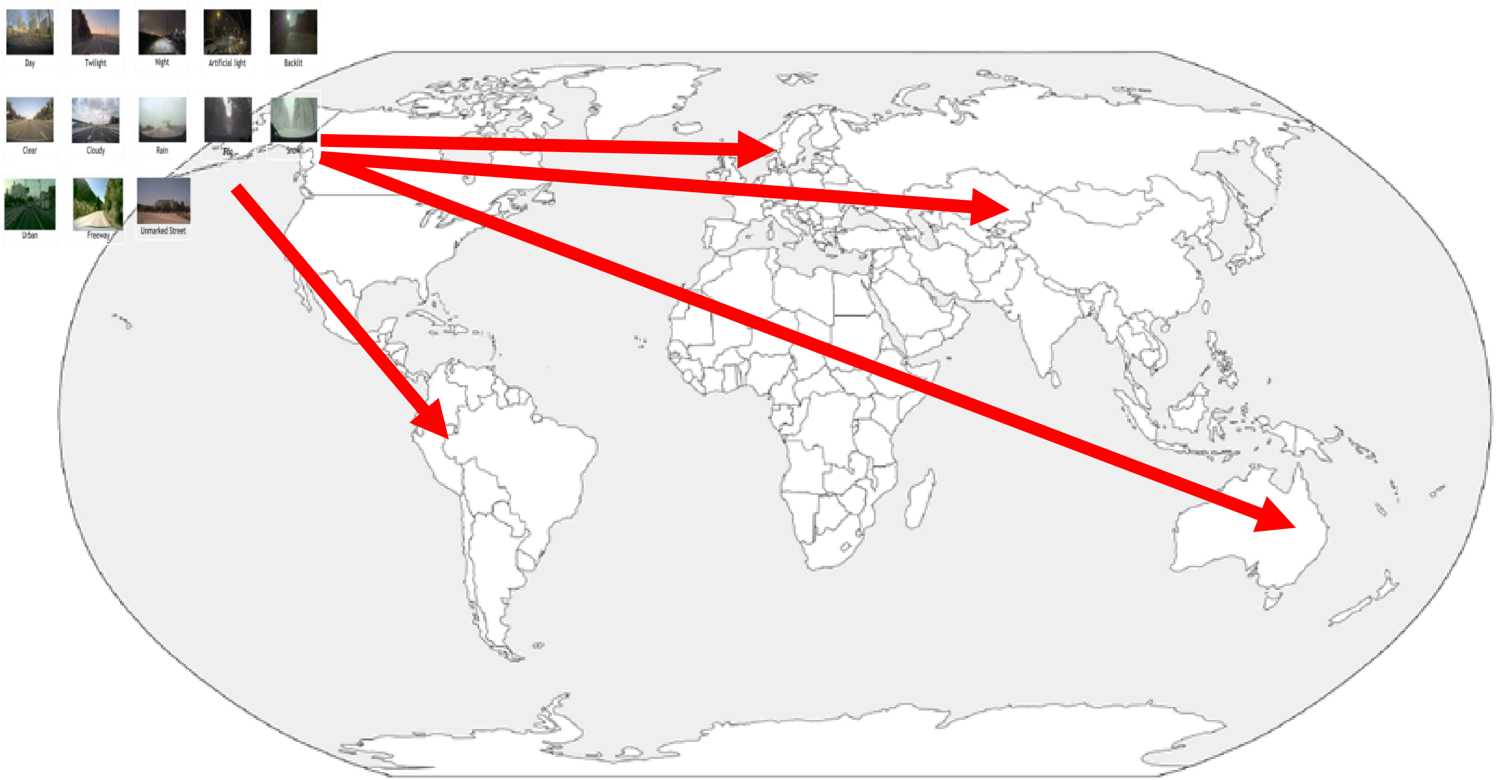

Urban


Freeway


Unmarked Street

Geographic Locations

# Domain Adaptation



# Domain Adaptation

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

Synthetic data can be obtained in large amounts and is labeled automatically.

| Domain | Images | Annotations |
|---|---|---|
| Source | ☺ | ☺ |
| Target | ☹ | ☹ |

# Domain Adaptation

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.

| Domain | Images | Annotations |
|---|---|---|
| Source | ☺ | ☺ |
| Target | ☹ | ☹ |

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.

| Domain | Images | Annotations |
|---|---|---|
| Source | ☺ | ☺ |
| Target | ☹ | ☹ |

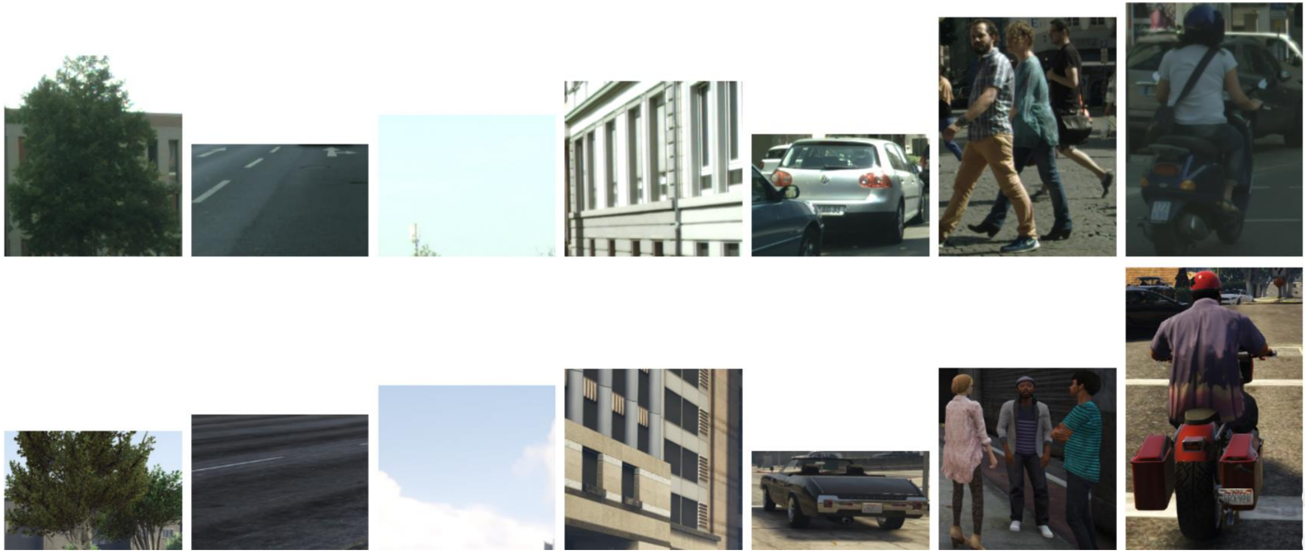
*[Saleh, Salzmann, Alvarez et al. 2018], Efficient use of Synthetic data for Semantic Segmentation, ECCV2018*

# 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.

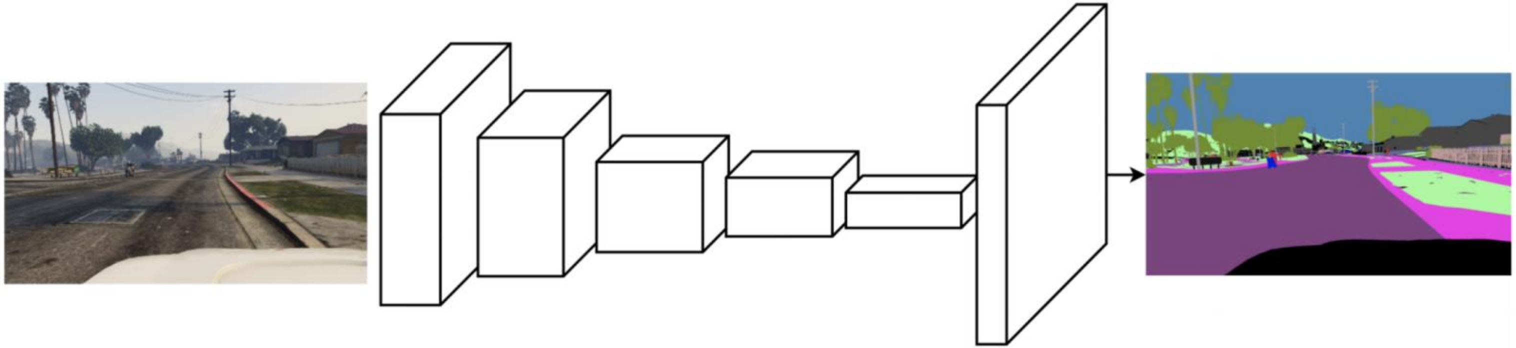


[Saleh, Salzmann, Alvarez et al. 2018], *Efficient use of Synthetic data for Semantic Segmentation, ECCV2018*

# Domain Adaptation

## Efficient use of Synthetic Data

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

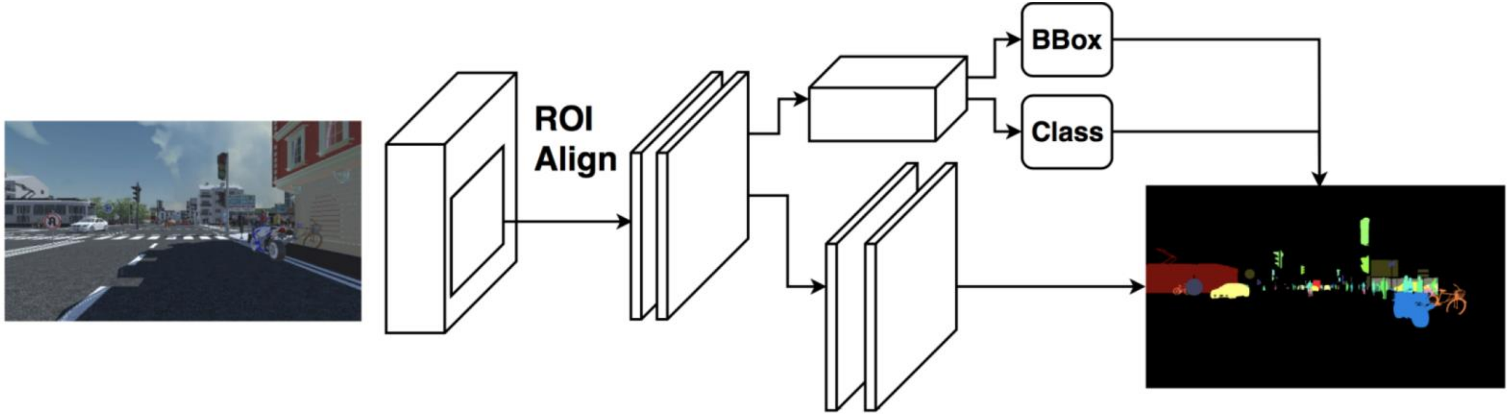


[Saleh, Salzmann, Alvarez et al. 2018], *Efficient use of Synthetic data for Semantic Segmentation, ECCV2018*

# 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.**

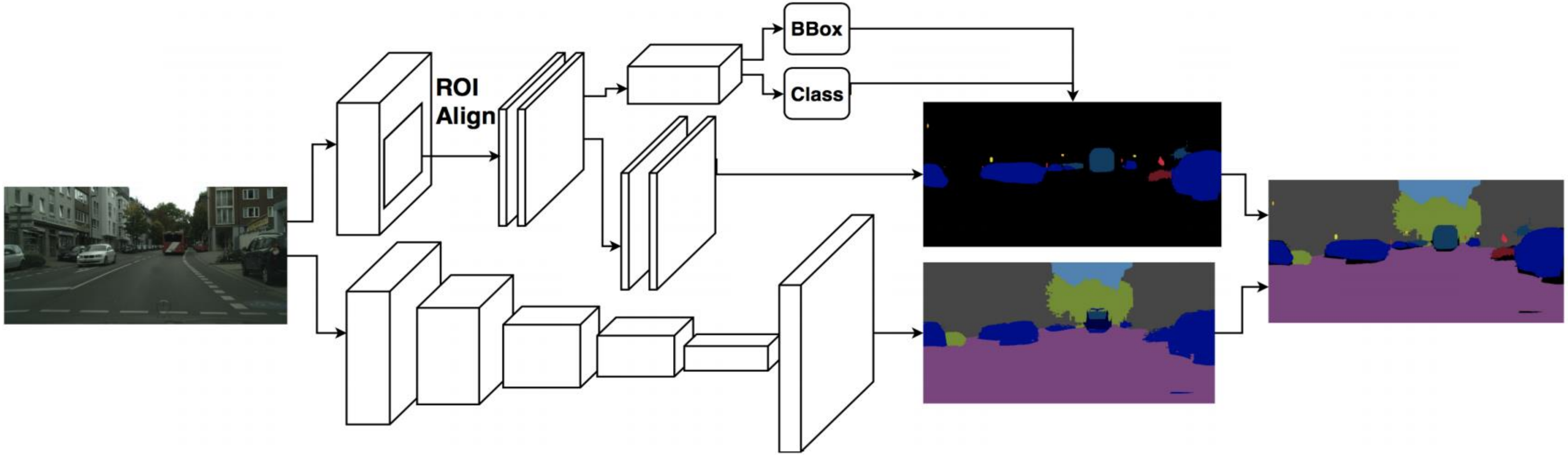



*[Saleh, Salzmann, Alvarez et al. 2018], Efficient use of Synthetic data for Semantic Segmentation, ECCV2018*

# 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.

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

*[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)*

| Domain | Images | Annotations |
|---|---|---|
| Source | ☺ | ☺ |
| Target | ☺ | ☹ |

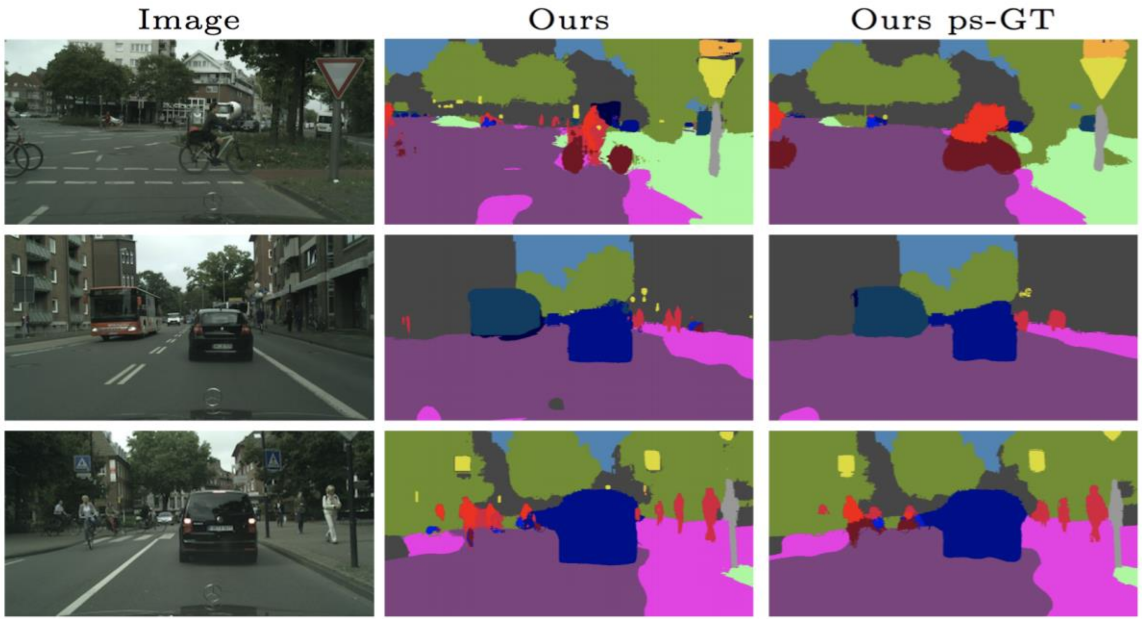
*[Saleh, Salzmann, Alvarez et al. 2018], Efficient use of Synthetic data for Semantic Segmentation, ECCV2018*

| Domain | Images | Annotations |
|---|---|---|
| Source | ☺ | ☺ |
| Target | ☺ | ☹ |

# Domain Adaptation

## Efficient use of Synthetic Data

### Adding Pseudo-labels:




### Comparison on models trained on synthetic data

| Methods | mIOU |
|---|---|
| GTA5 [5] | 21.9 |
| GTA5 | 31.3 |
| SYNTHIA | 22.1 |
| VIPER | 23.9 |
| VEIS | 24.4 |
| GTA5+VEIS | 32.8 |
| GTA5+VEIS&ps-GT | 33.3 |
| Ours | 38.0 |
| Ours&ps-GT | 42.5 |

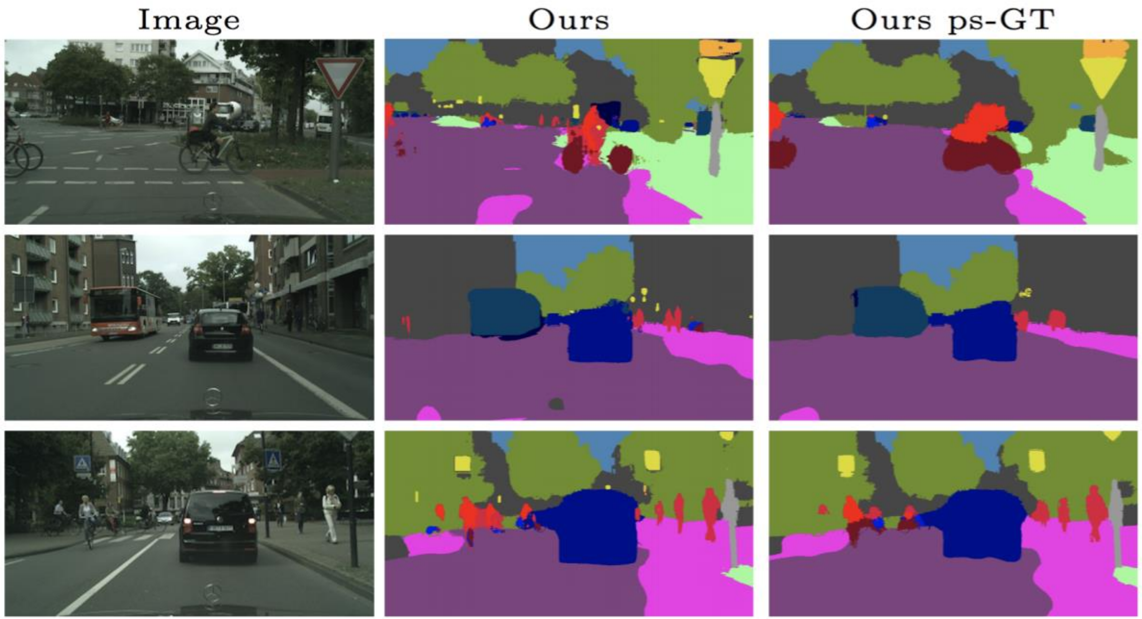
*[Saleh, Salzmann, Alvarez et al. 2018], Efficient use of Synthetic data for Semantic Segmentation, ECCV2018*

| Domain | Images | Annotations |
| --- | --- | --- |
| Source | ☺ | ☺ |
| Target | ☺ | ☹ |

# Domain Adaptation

## Efficient use of Synthetic Data

### Adding Pseudo-labels:




### Comparison to domain adaptation and weakly- supervised methods

| Methods | mIOU |
| --- | --- |
| Fully Sup. | 56.2 |
| Weakly-Sup.[2] | 23.6 |
| FCNs in Wld [3] | 27.1 |
| Curriculum [4] | 28.9 |
| ROAD [5] | 35.9 |
| CYCADA [6] | 35.4 |
| Ours | 38.0 |
| Ours+Pseudo-GT | 42.5 |

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

# Accuracy vs Efficiency
# (for Large datasets)

# Accuracy vs Efficiency

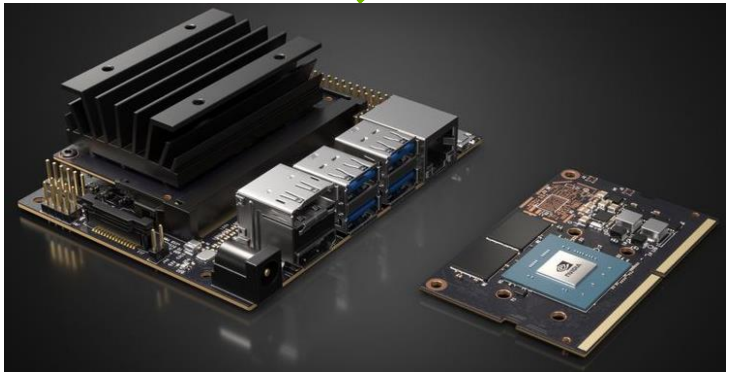

TRAINING


TESTING

# Accuracy vs Efficiency

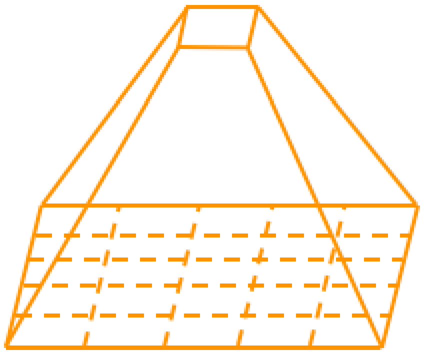
## Efficient Training of DNN

Goal: **maximize training resources** while obtaining **deployment 'friendly'** network.

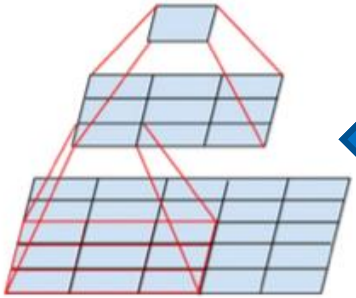


# Over-parameterization

# Accuracy vs Efficiency



5x5 convolution



two successive
3x3 convolutions

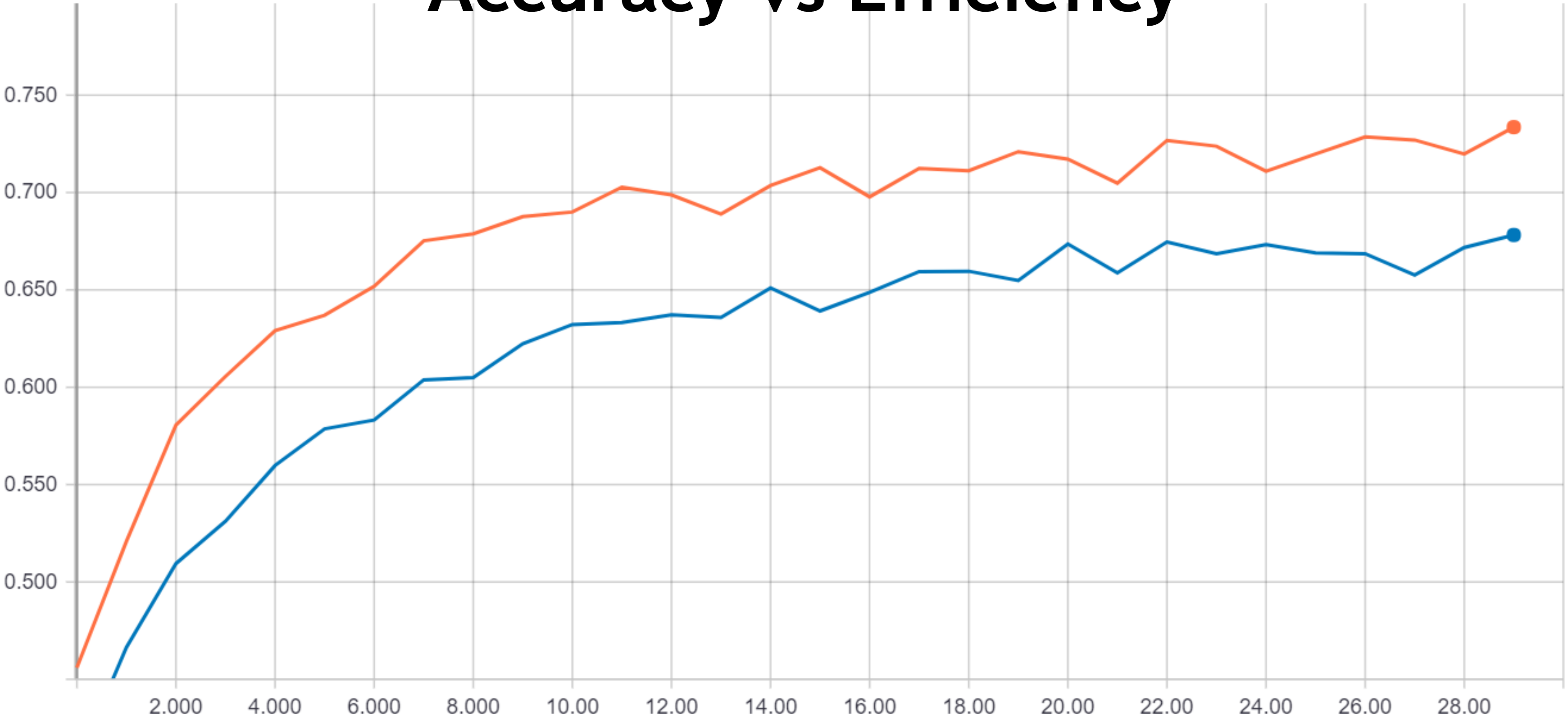
Non-linearity

Capacity

Num.
parameters

Same receptive field

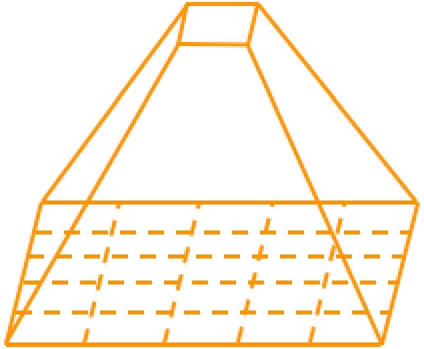
# Accuracy vs Efficiency




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

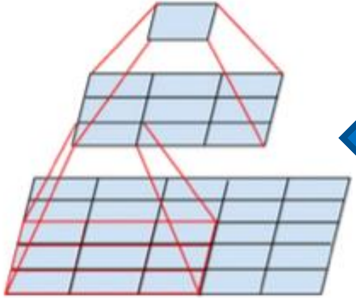
https://blog.sicara.com/about-convolutional-layer-convolution-kernel-9a7325d34f7d

# Accuracy vs Efficiency



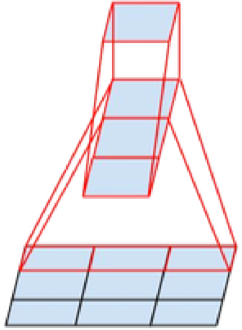
5x5 convolution

Same receptive field



two successive
3x3 convolutions

Non-linearity



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

Non-linearity

Capacity

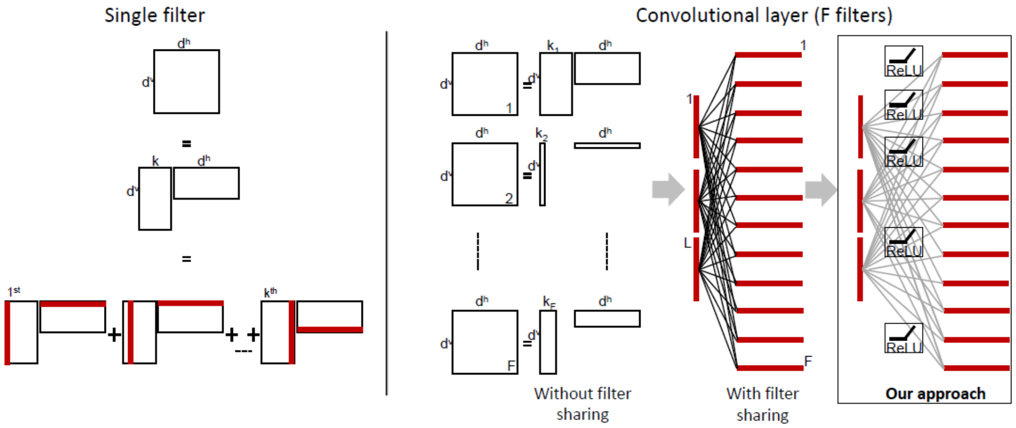
Num.
parameters

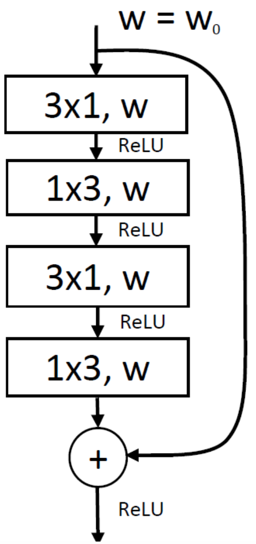
FLOPS

?

# Accuracy vs Efficiency

## Filter Decompositions for Real-time Semantic Segmentation



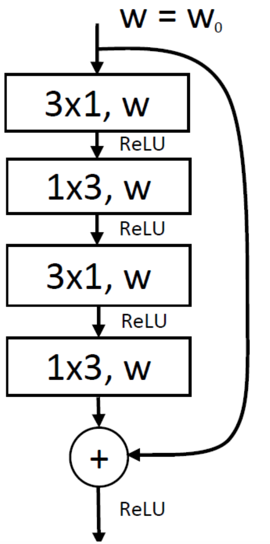





[Alvarez and Petersson], DecomposeMe: Simplifying ConvNets for End-to-End Learning. Arxiv 2016
[Romera, Alvarez et al.] , Efficient ConvNet for Real-Time Semantic Segmentation. IEEE-IV 2017, T-ITS 2018

# Accuracy vs Efficiency

## Filter Decompositions for Real-time Semantic Segmentation




**Cityscapes** dataset (19 classes, 7 categories)

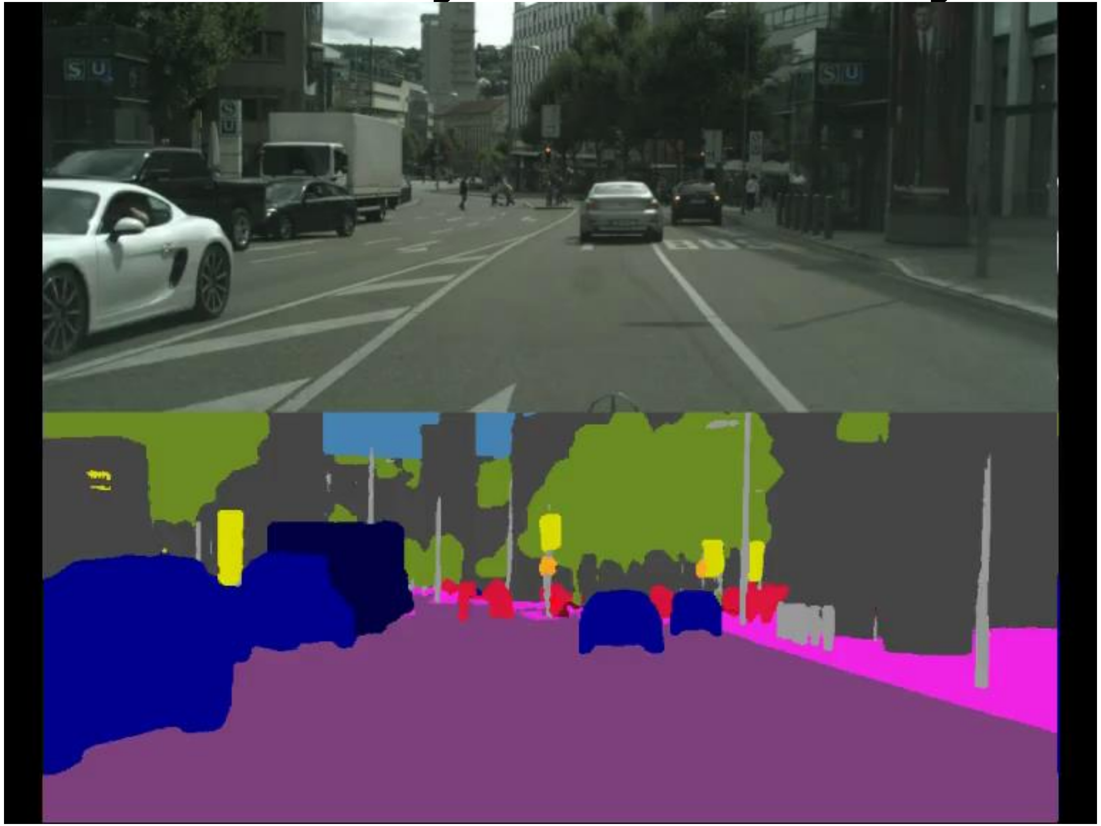
| Train mode | Pixel accuracy | Class IoU | Category IoU |
|---|---|---|---|
| Scratch | 94.7 % | 70.0 % | 86.0 % |
| Pre-trained | 95.1 % | 71.5 % | 86.9 % |

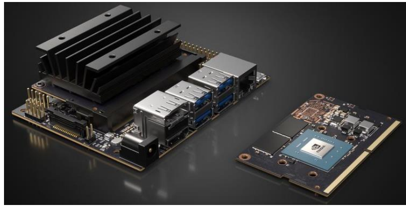
**Forward-Time: Cityscapes** 19 classes

| | TEGRA-TX1 | | | TITAN-X | | |
|---|---|---|---|---|---|---|
| Fwd Pass | 512x256 | 1024x512 | 2048x1024 | 512x256 | 1024x512 | 2048x1024 |
| Time | 85 ms | 310 ms | 1240 ms | 8 ms | 24 ms | 89 ms |
| FPS | 11.8 | 3.2 | 0.8 | 125.0 | 41.7 | 11.2 |

[Romera, Alvarez et al.] , Efficient ConvNet for Real-Time Semantic Segmentation. IEEE-IV 2017, T-ITS 2018

# Accuracy vs Efficiency



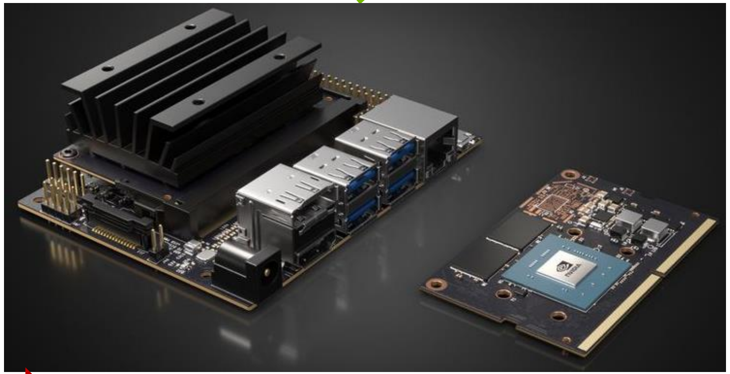


[Romera, Alvarez et al.] , Efficient ConvNet for Real-Time Semantic Segmentation. IEEE-IV 2017, T-ITS 2018

# Accuracy vs Efficiency

## Efficient Training of DNN

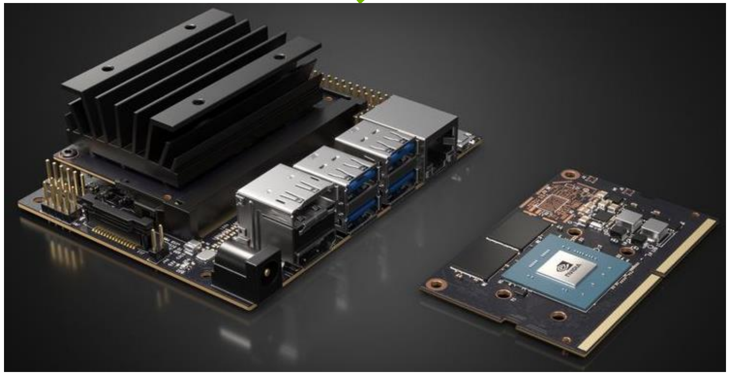
Goal: **maximize training resources** while obtaining **deployment 'friendly'** network.



# Accuracy vs Efficiency

## Efficient Training of DNN

**Goal:** **maximize training resources** while obtaining **deployment 'friendly'** network.



# Accuracy vs Efficiency

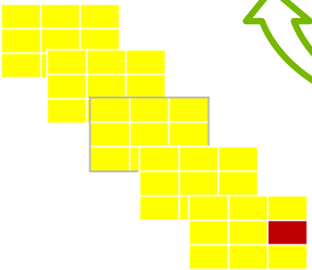
## Common Approach

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

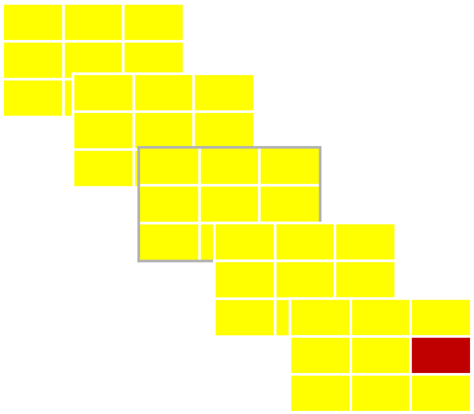
## Joint Training and Pruning Deep Networks

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

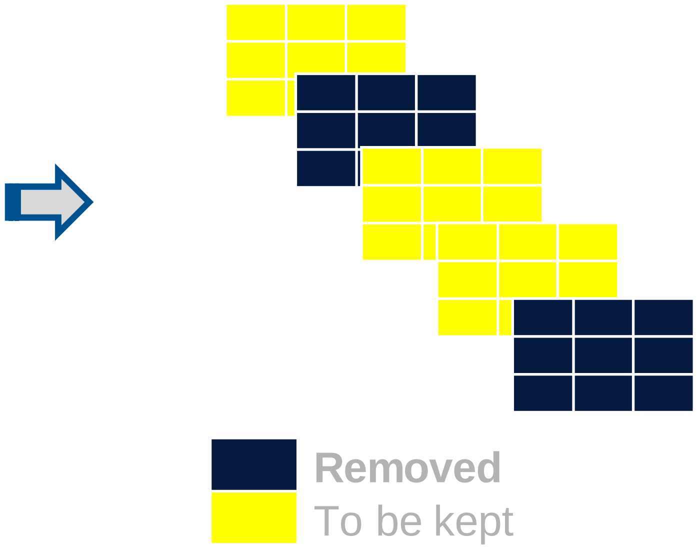
**Joint Train / Pruning** → **DEPLOY**
Optimize for Specific hardware

# Accuracy vs Efficiency

## Joint Training and Pruning Deep Networks



Convolutional layer
5x1x3x3

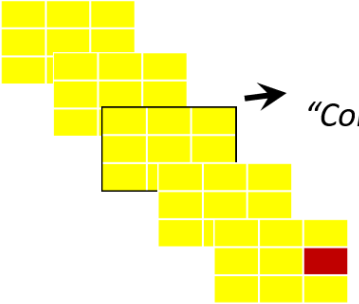


# Accuracy vs Efficiency

## Joint Training and Pruning Deep Networks

**Common approach:**

$$\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(\mathbf{x}_i, \Theta)) + \frac{\rho}{2} ||\Theta||_2^2$$



Neuron
*"Convolutional kernel"*

Convolutional layer
5x1x3x3

*Considers each parameter independently*

Weight Decay
(prevent weights with large values)

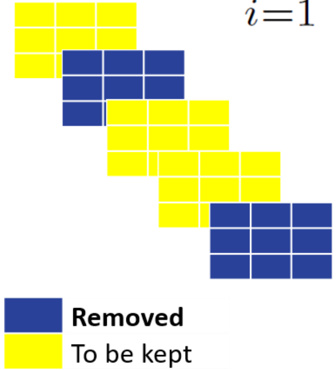
*[Alvarez and Salzmann], Learning the number of neurons in Neural Nets, NIPS 2016*
*[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017*

# Accuracy vs Efficiency

## Joint Training and Pruning Deep Networks

**Our Approach:**

$$\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(\mathbf{x}_i, \Theta)) + \frac{\rho}{2}||\Theta||_2^2 + r(\Theta) \,,$$




$$r(\Theta) = \sum_{l=1}^{L} \lambda_l \sqrt{P_l} \sum_{n=1}^{N_l} ||\theta_l^n||_2$$

Size of the group

*[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).

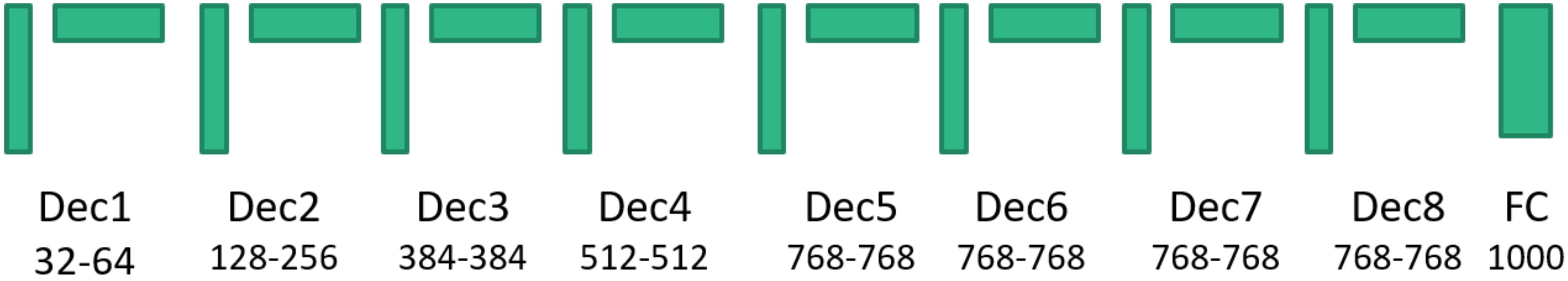
*[Alvarez and Salzmann], Learning the number of neurons in Neural Nets, NIPS 2016*
*[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017*

# Accuracy vs Efficiency

## Joint Training and Pruning Deep Networks

Quantitative Results on **ImageNet**

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

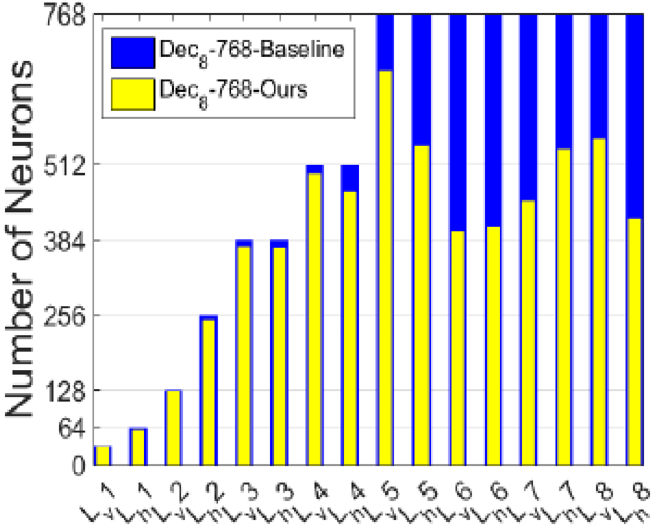



*[Alvarez and Salzmann], Learning the number of neurons in Neural Nets, NIPS 2016*
*[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017*

# Accuracy vs Efficiency


## Joint Training and Pruning Deep Networks

Quantitative Results on **ImageNet**




| $Dec_8$ on ImageNet (in %) | | | | |
|---|---|---|---|---|
| | $Dec_8$ | $Dec_8$-640 | | $Dec_8$-768 |
| | GS | SGL | GS | GS |
| neurons | 3.39 | 12.42 | 4.02 | 26.83 |
| group param | 2.46 | 13.69 | 4.22 | 31.53 |
| total param | 2.46 | 22.72 | 4.22 | 31.63 |
| total induced | 2.82 | 23.33 | 10.83 | 32.26 |
| accuracy gap | 0.01 | 0.94 | 2.45 | -0.02 |

*[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**

TESCO, Value
Washing Up Liquid


PEPSI



The Rab Butler Building

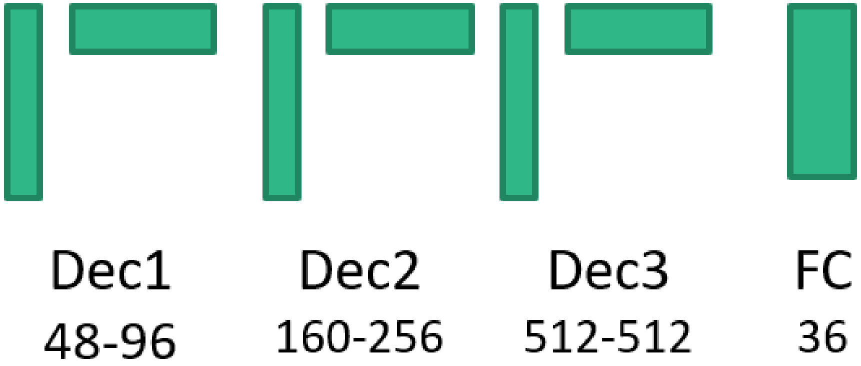
*[Alvarez and Salzmann], Learning the number of neurons in Neural Nets, NIPS 2016*
*[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017*

# 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 (*$Dec_3$-512*)

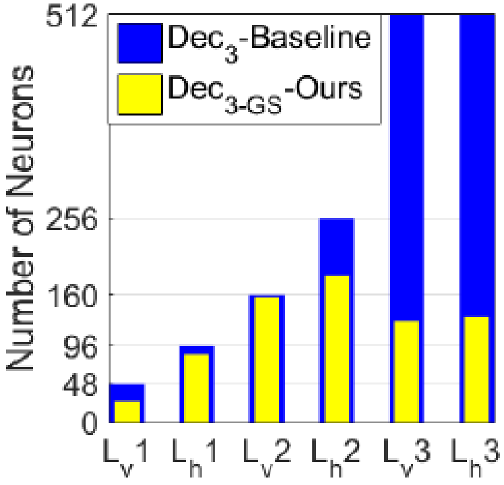



*[Alvarez and Salzmann], Learning the number of neurons in Neural Nets, NIPS 2016*
*[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017*

# Accuracy vs Efficiency

## Joint Training and Pruning Deep Networks

Quantitative Results on **ICDAR character recognition dataset**




| Dec$_3$ on ICDAR (in %) | S-GS | GS |
|---|---|---|
| neurons | 38.64 | 55.11 |
| group param | 32.57 | 66.48 |
| total param | 72.41 | 66.48 |
| total induced | 72.08 | 66.52 |
| accuracy gap | 1.24 | 1.38 |

| Top-1 acc. on ICDAR | |
|---|---|
| MaxOut$_{Dec}$[a] | 91.3% |
| MaxOut[b] | 89.8% |
| MaxPool$_{2Dneurons}$ | 83.8% |
| Dec$_3$ (baseline) | 89.3% |
| Ours-Dec$_{3-SGL}$ | 89.9% |
| Ours-Dec$_{3-GS}$ | 90.1% |

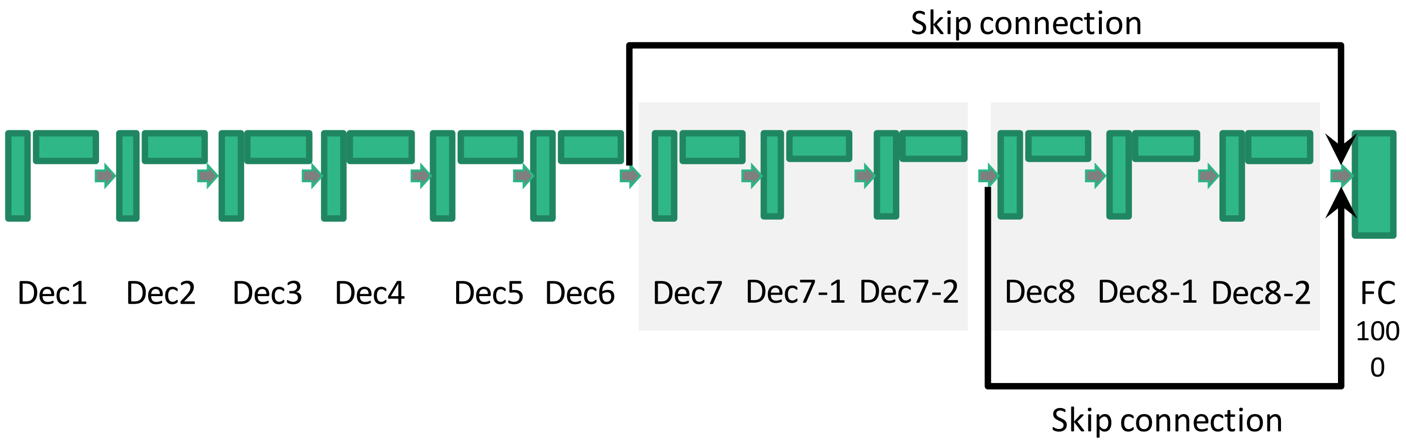
[a] Results from Jaderberg et al. [2014a] using MaxOut layer instead of Max-Pooling and decompositions as post-processing step

[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*

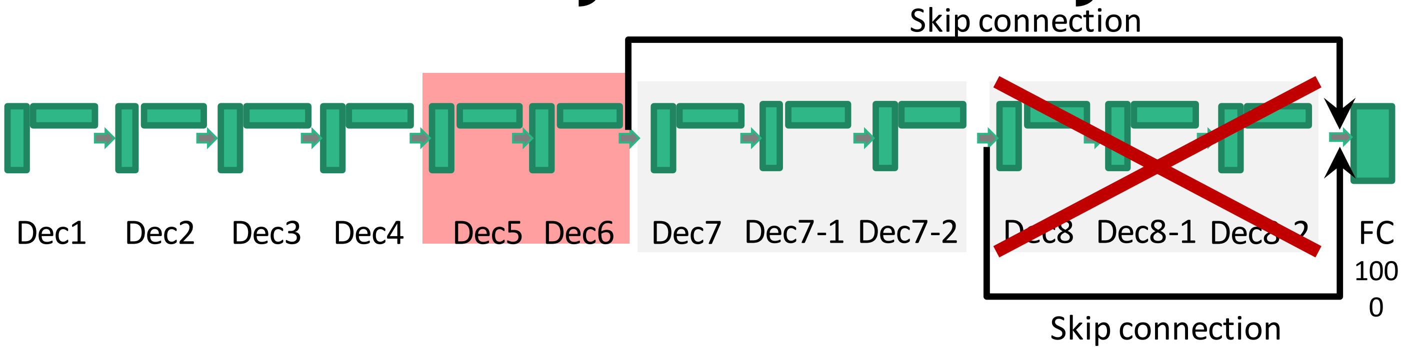
# Accuracy vs Efficiency

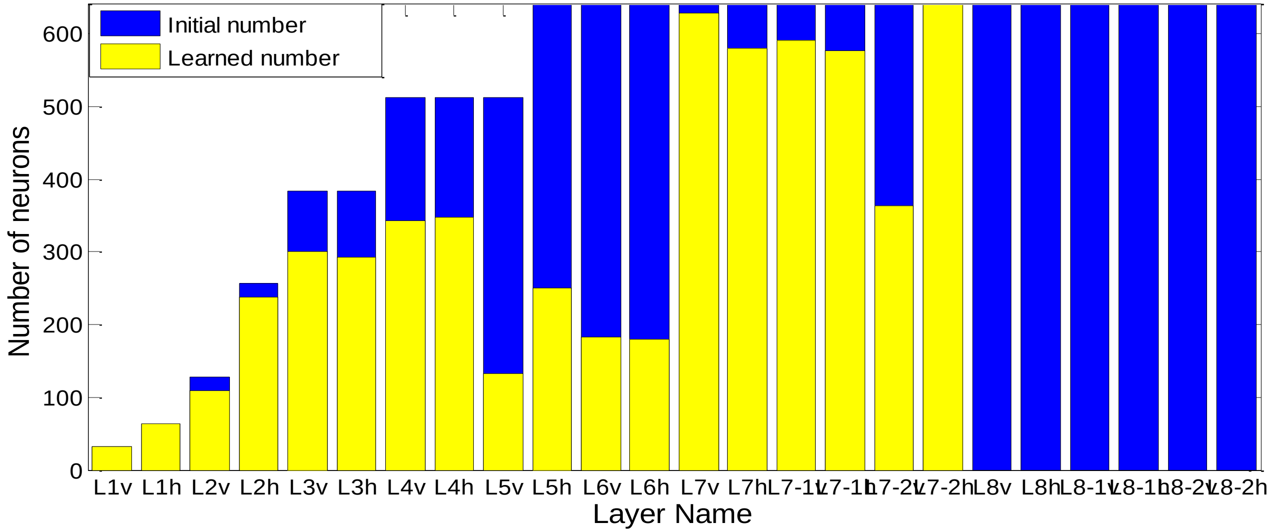
## Joint Training and Pruning Deep Networks



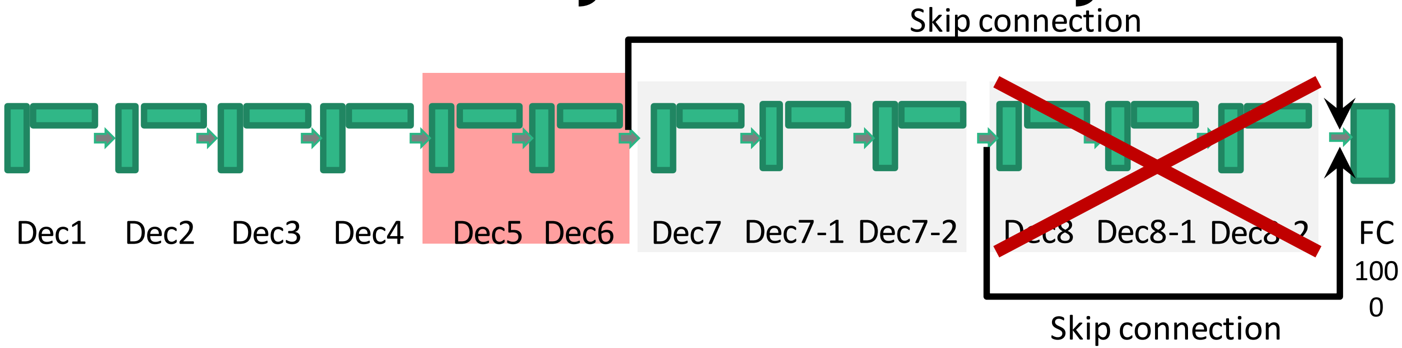

*[Alvarez and Salzmann], Learning the number of neurons in Neural Nets, NIPS 2016*
*[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017*

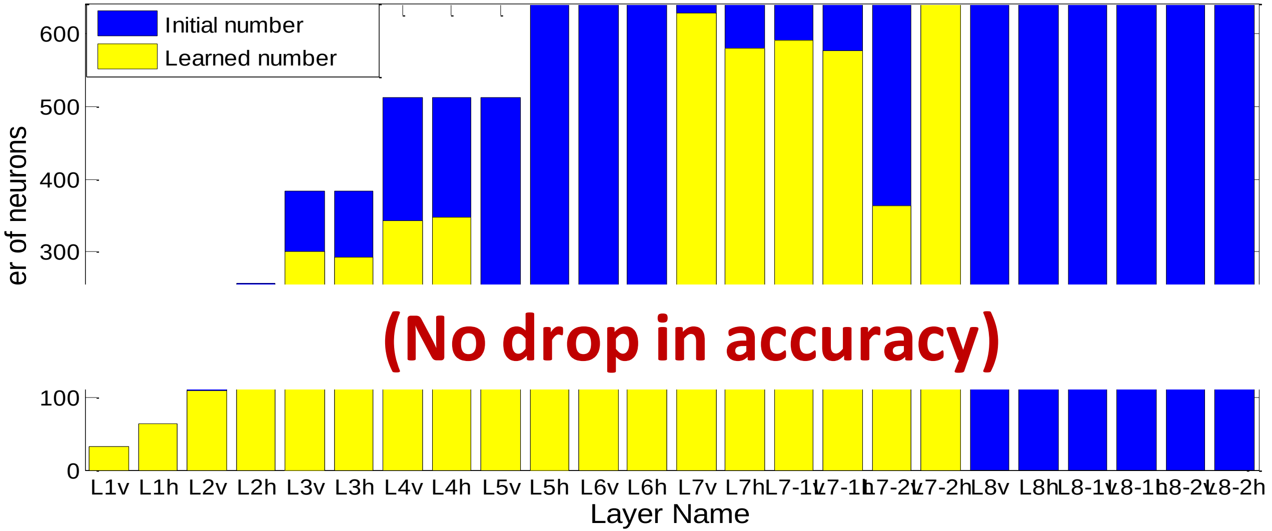
# Accuracy vs Efficiency



# Accuracy vs Efficiency

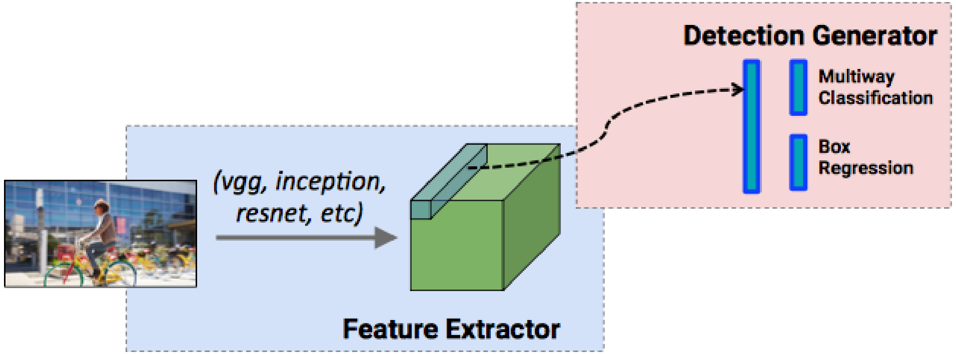



**(No drop in accuracy)**

KITTI

# Object Detection Results



# Accuracy vs Efficiency

## Object Detection

KITTI

**TRAIN**
Promising model

**Prune / Optimize**
For a specific application

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

# Accuracy vs Efficiency

## Object Detection

KITTI

TRAIN

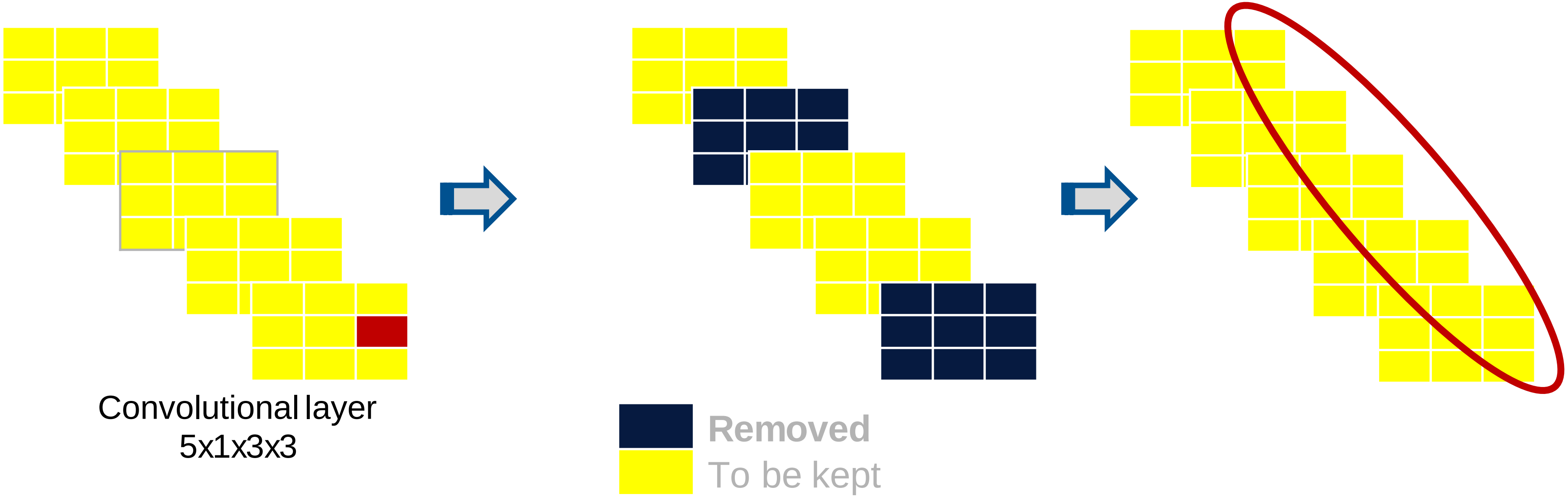
Prune / Optimize

| model | car | | | | cyclist | | | | pedestrian | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | weighted | hard | easy | mdrt | weighted | hard | easy | mdrt | weighted | hard | easy | mdrt | params |
| Baseline (L1) model_p_9 | 83.74 | 82.96 | 92.29 | 90.27 | 51.59 | 47.56 | 61.32 | 48.68 | 68.19 | 58.32 | 71.06 | 64.07 | 283,116 |
| 0.0005 (GS) model_p_9 | 85.22 | 84.67 | 92.81 | 91.81 | 58.42 | 55.02 | 71.57 | 55.43 | 71.46 | 62.00 | 74.68 | 67.63 | 535,463 |

Joint Train / Pruning

# Accuracy vs Efficiency

## Compression-aware Training of DNN



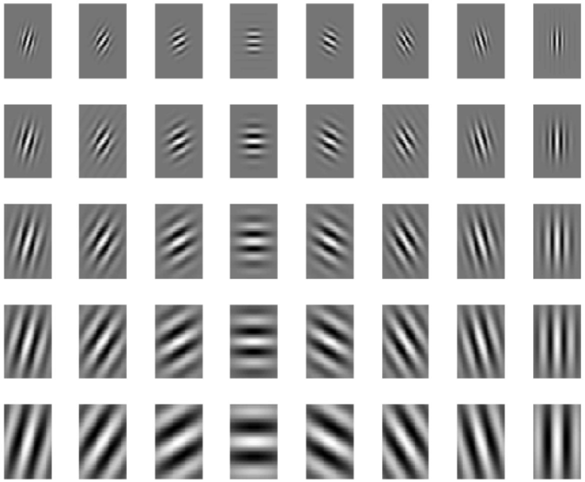

*[Alvarez and Salzmann], Learning the number of neurons in Neural Nets, NIPS 2016*
*[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017*

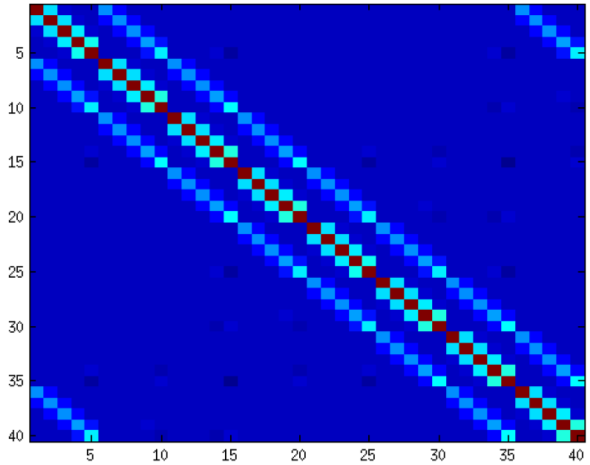
# Accuracy vs Efficiency

## 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

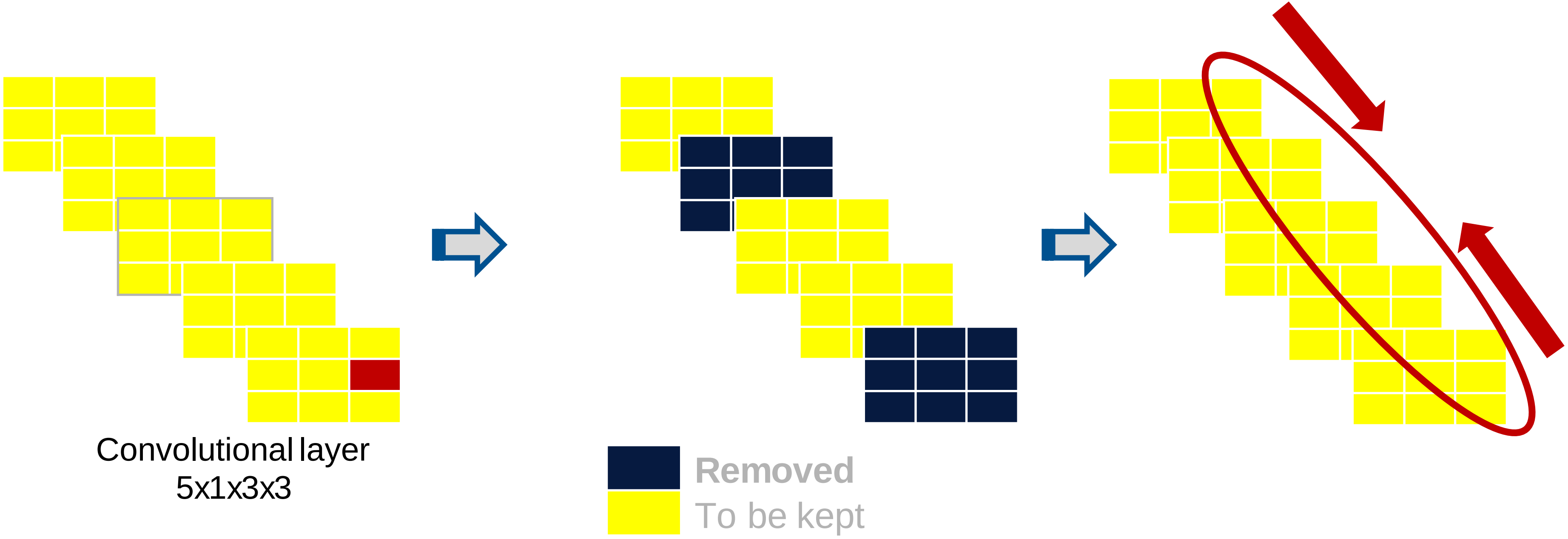
Weak-Points

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

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

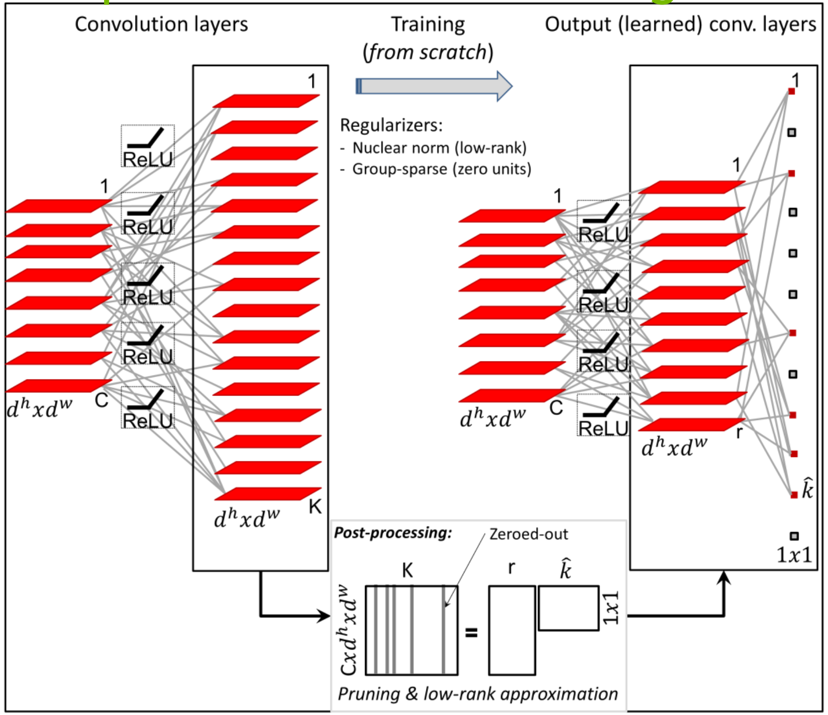
# Accuracy vs Efficiency

## Compression-aware Training of DNN



# Accuracy vs Efficiency

## Compression-aware Training of DNN



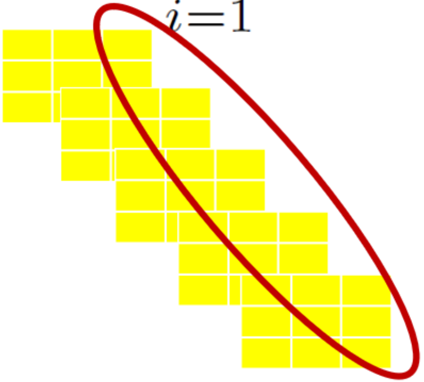

*[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017*

# Accuracy vs Efficiency

## Compression-aware Training of DNN

**Our Approach:**

$$\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(\mathbf{x}_i, \Theta)) + \frac{\rho}{2} ||\Theta||_2^2 + r(\Theta) + h(\Theta) \; .$$



Kernel Similarity at layer level

*Considers each Layer independently*

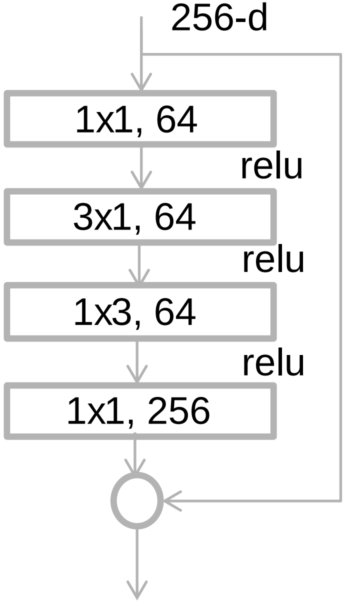
*[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017*

# Classification Results

# Accuracy vs Efficiency

## Compression-aware Training of DNN


Quantitative Results on **ImageNet using ResNet50***




| Resnet-50*, $e_l = 90\%$ | top-1 | Params. |
|---|---|---|
| Baseline | 74.7 | 18M |
| [14] | 74.0 | -4.0% |
| Group-sparse [2] | 74.5 | -17.0% |
| Ours (low-rank) | 75.0 | -20.6% |
| Low-rank + group-sparse | 75.2 | -27.0% |

*Modified to use 1D kernels.

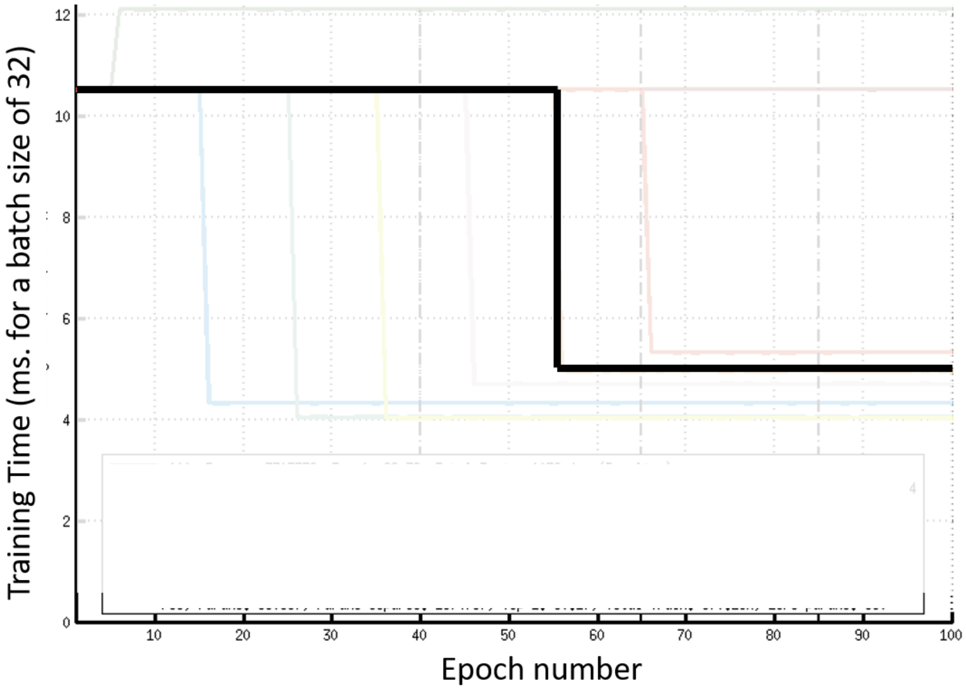
*[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017*

# Training Efficient

(side benefit)

# Accuracy vs Efficiency

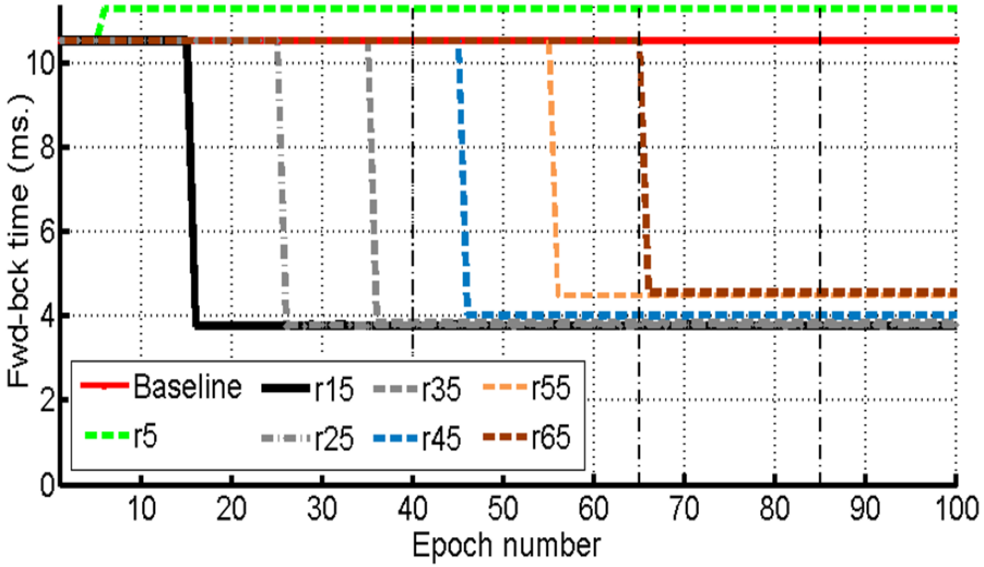
## Compression-aware Training of DNN




[Alvarez and Salzmann], *Compression-aware training of DNN, NIPS 2017*

# Accuracy vs Efficiency

## Compression-aware Training of DNN

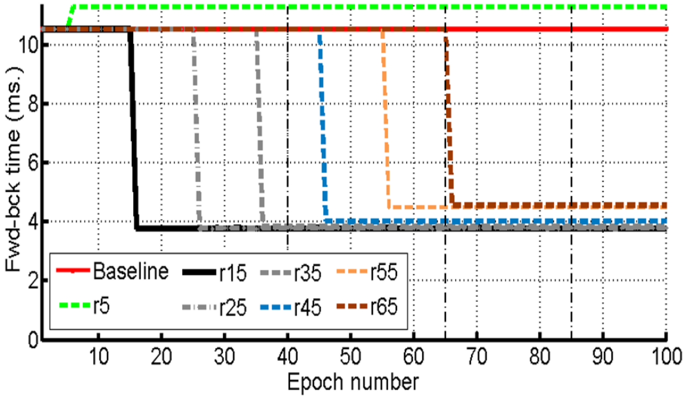



*[Alvarez and Salzmann], Compression-aware training of DNN, NIPS 2017*

# Accuracy vs Efficiency

## Compression-aware Training of DNN

| | Epoch reload | Num. parameters Total | Num. parameters no SVD | accuracy top-1 | Total train-time |
|---|---|---|---|---|---|
| Baseline | – | 3.7M | – | 88.4% | 1.69h |
| r5 | 5 | 3.2M | 3.71M | 89.8% | 1.81h |
| **r15** | **15** | **210K** | **2.08M** | **90.0%** | **0.77h** |
| r25 | 25 | 218K | 1.60M | 90.0% | 0.88h |
| r35 | 35 | 222K | 1.52M | 89.0% | 0.99h |
| r45 | 45 | 324K | 1.24M | 90.1% | 1.12h |
| r55 | 55 | 388K | 1.24M | 89.2% | 1.26h |
| r65 | 65 | 414K | 1.23M | 87.7% | 1.36h |




**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?

| | #Epochs | | Training | | Test | | Parameters | |
|---|---|---|---|---|---|---|---|---|
| | Total | Reload | top-1 | top-5 | top-1 | top-5 | Total | Zeroed-out |
| baseline | 75 | – | 99.73% | 99.96% | 88.59% | 96.73% | 3717924 | 3088 (0) |
| Ours | 75 | 55 | 97.71% | 99.62% | 89.73% | 97.25% | 225851 | 782 (16) |
| Compact | 75 | 0 | 98.24% | 99.76% | 87.53% | 96.65% | 225851 | 34 (0) |

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

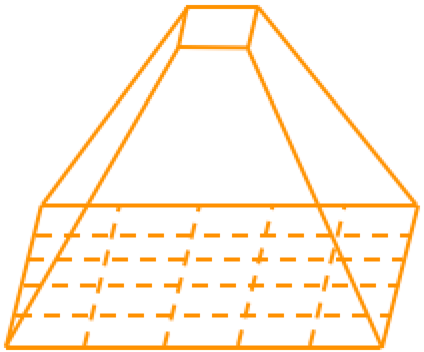
Data Movements may be more significant than current savings.



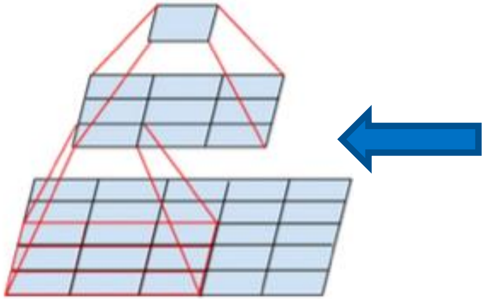
[Alvarez and Salzmann], *Compression-aware training of DNN, NIPS 2017*

# Accuracy vs Efficiency
# (more on over-parameterization)

# Accuracy vs Efficiency



5x5 convolution



two successive
3x3 convolutions

Non-linearity

Same receptive field

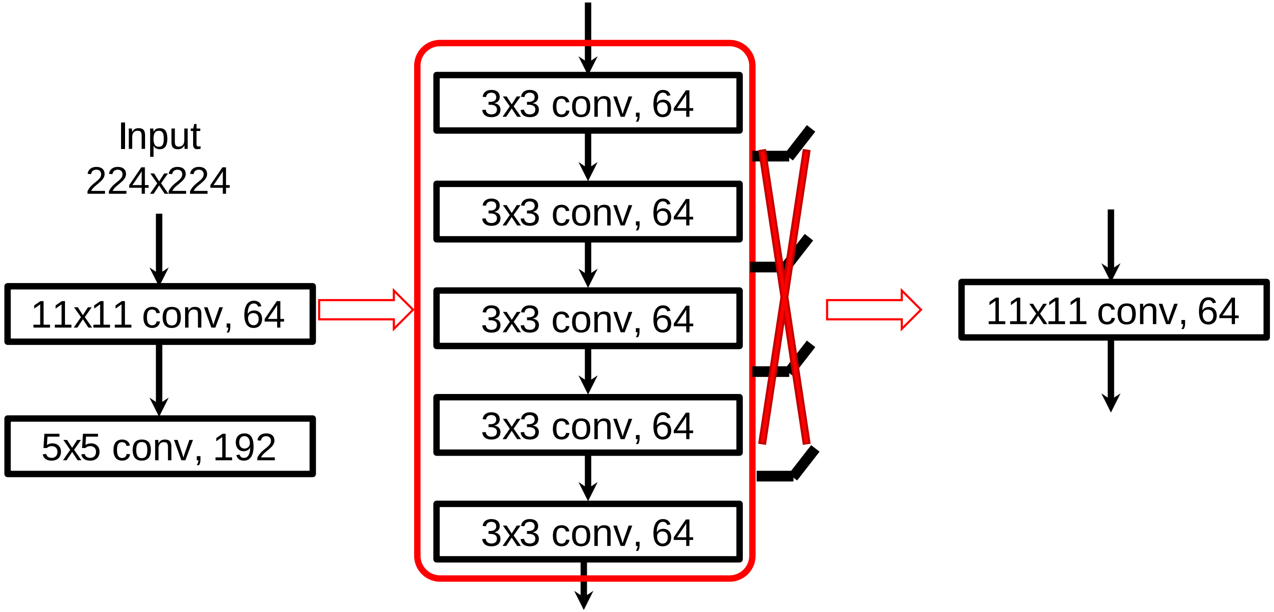
Capacity

Num. parameters

Num. layers

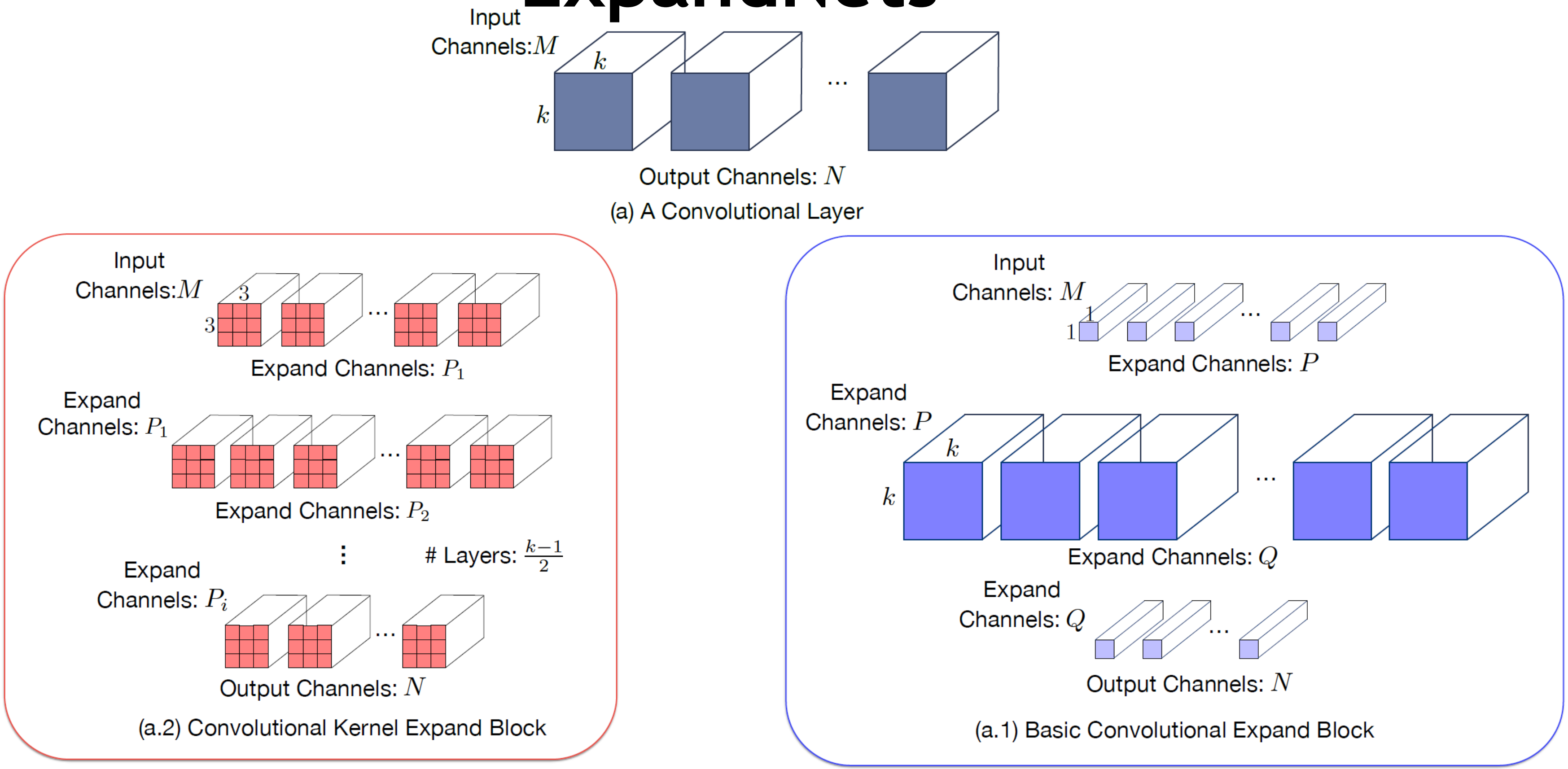
# ExpandNets
# Exploiting Linear Redundancies

# ExpandNets



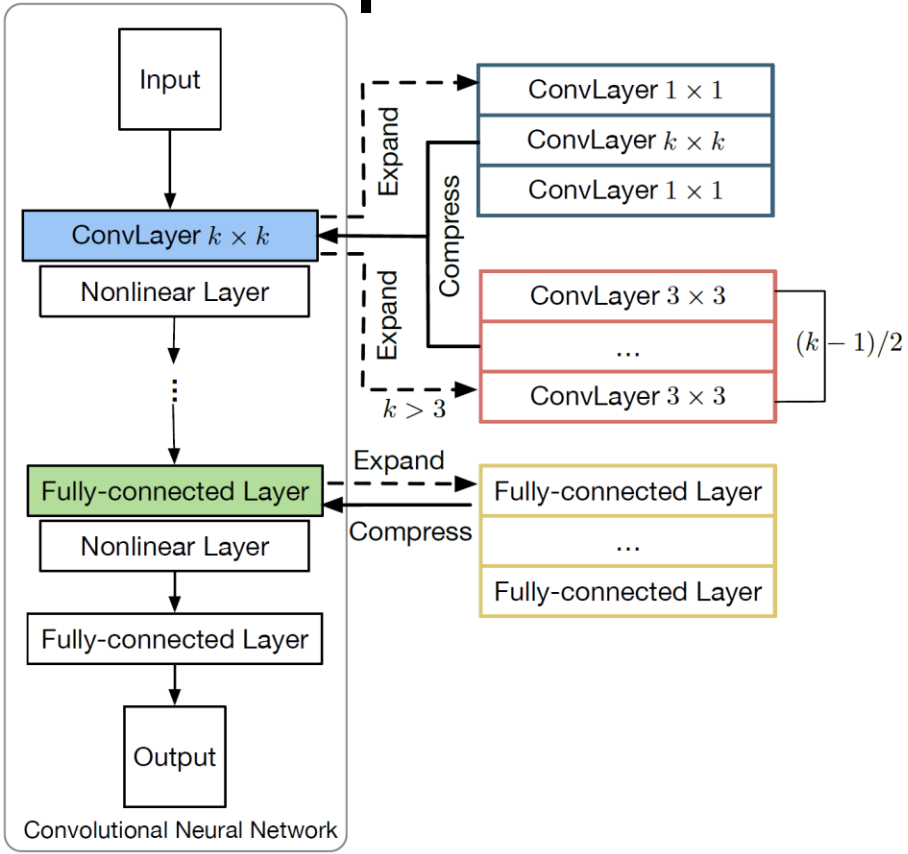

[*Guo, Alvarez, Salzmann*], ExpandNets: Exploiting Linear Redundancy to Train Small Networks. Arxiv 2018

# ExpandNets




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

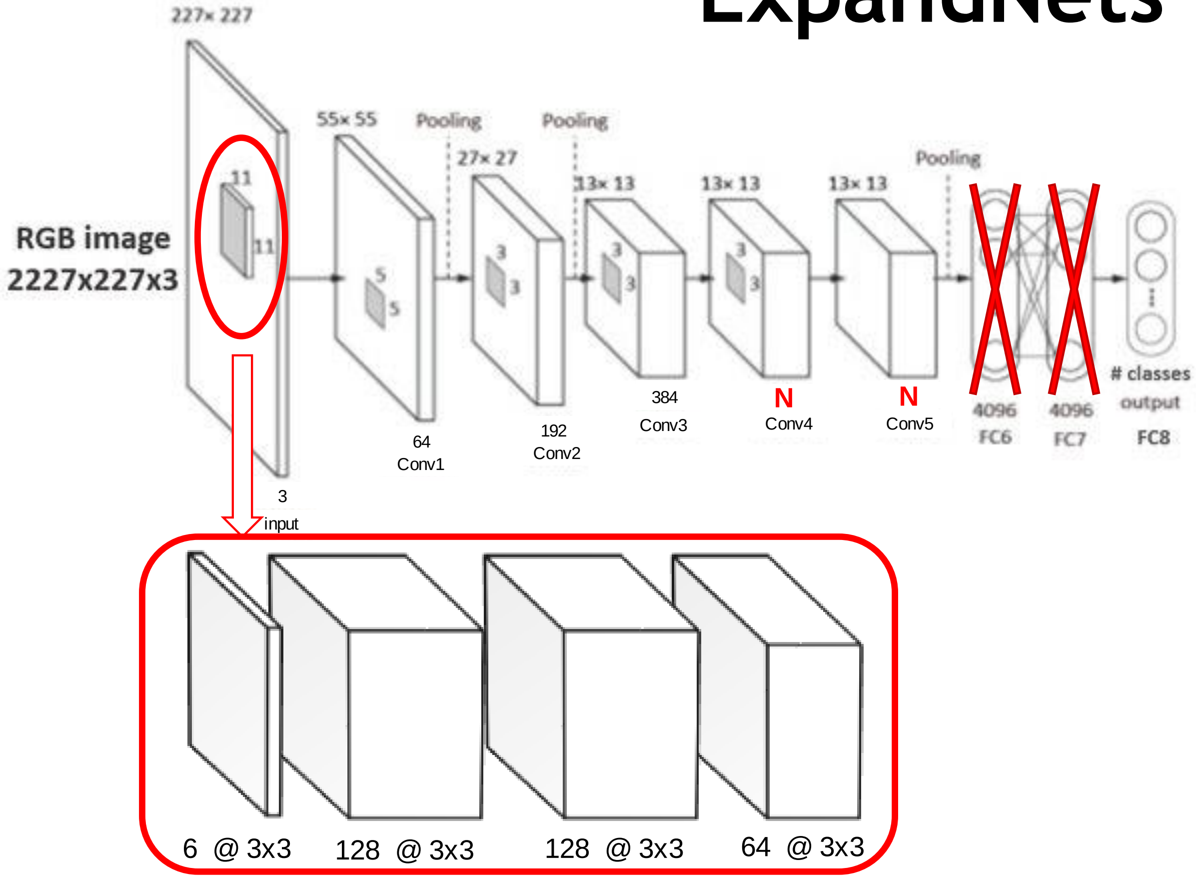
# ExpandNets




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

# Classification Results

# ExpandNets



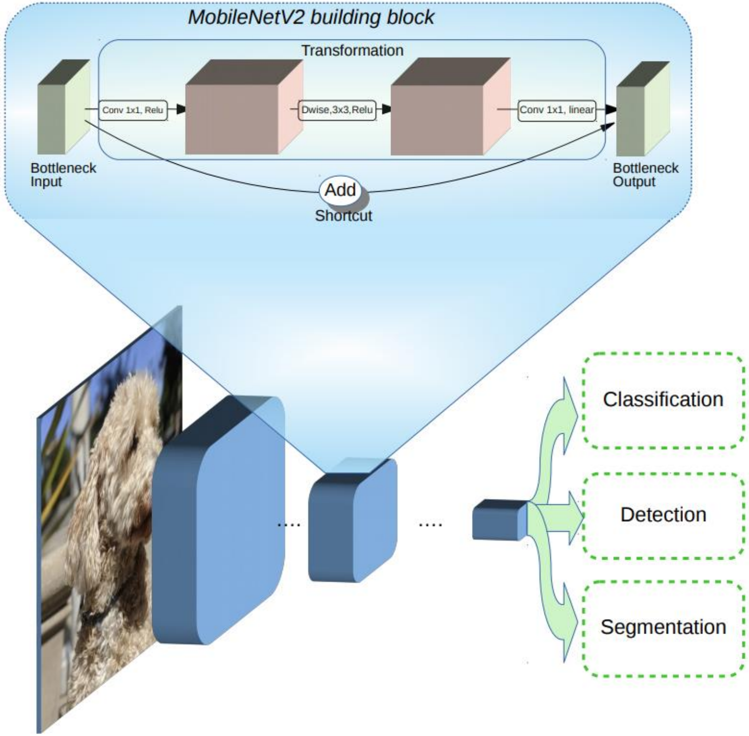


| ImageNet | Baseline | Expanded |
|---|---|---|
| N=128 | 46.72% | 49.66% |
| N=256 | 54.08% | 55.46% |
| N=512 | 58.35% | 58.75% |

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

# ExpandNets

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

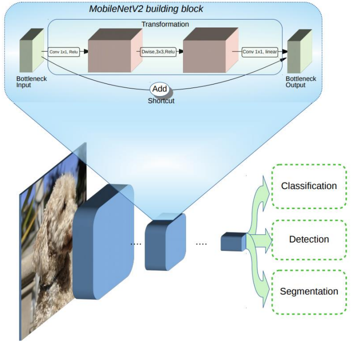


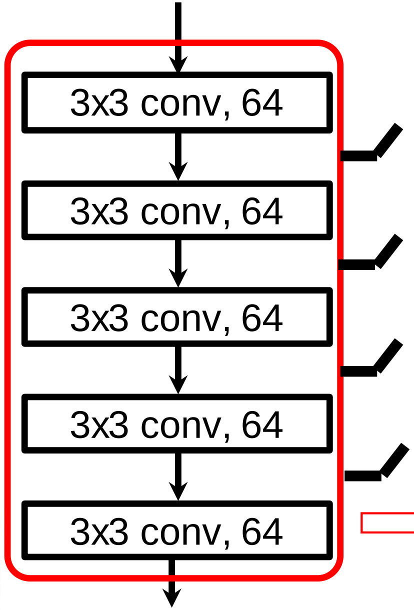

| Model | Top-1 | Top-5 |
|---|---|---|
| MobileNetV2 | 70.78% | 91.47% |
| MobileNetV2- expanded | 74.85% | 92.15% |

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

# ExpandNets

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




| Model | Top-1 | Top-5 |
|---|---|---|
| MobileNetV2 | 70.78% | 91.47% |
| MobileNetV2-expanded | 74.85% | 92.15% |
| MobileNetV2-expanded-nonlinear | 74.17% | 91.61% |
| MobileNetV2-expanded (nonlinear Init) | **75.46%** | **92.58%** |

[*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 ~**2.34%**
improvement on fscore

Thanks Ian Ivanecky!

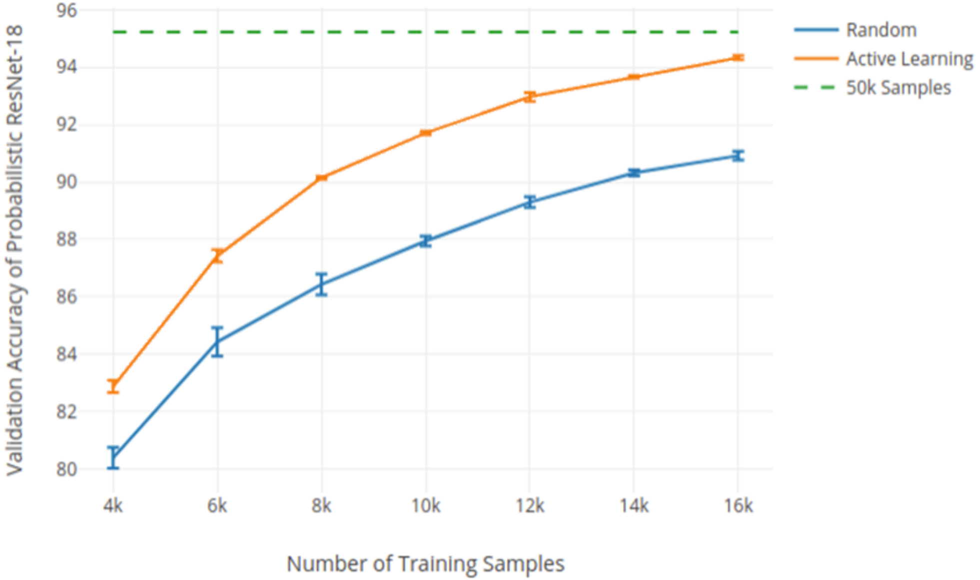
[*Guo, Alvarez, Salzmann*], ExpandNets: Exploiting Linear Redundancy to Train Small Networks. Arxiv 2018

# Summary

# Summary

Creating the right datasets

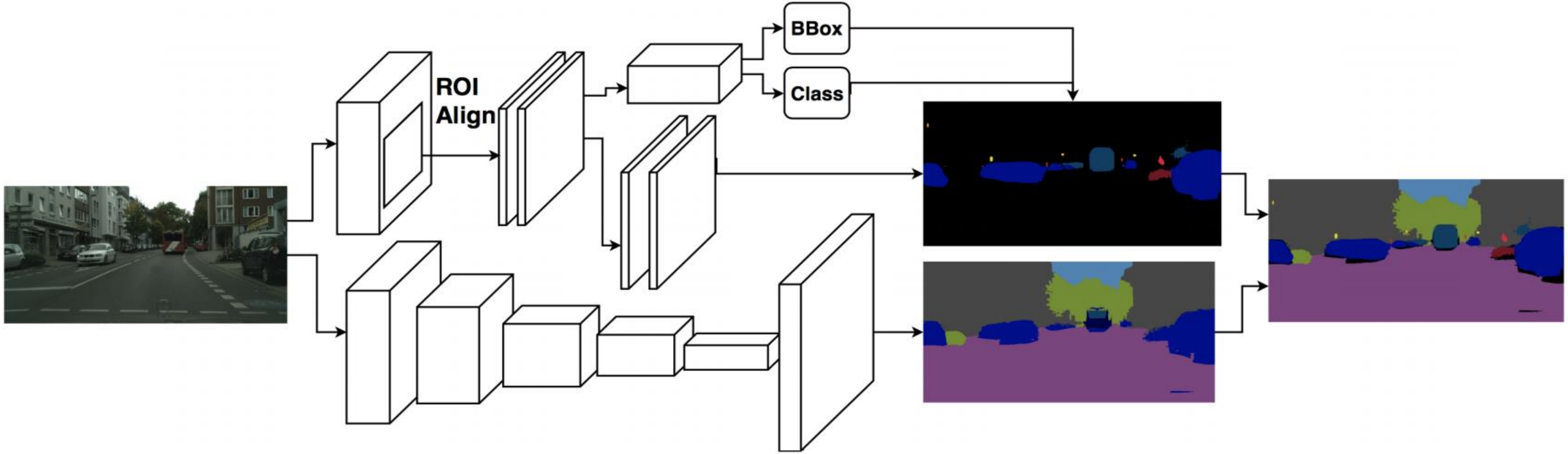
- Active Learning: Our Deep Probabilistic Ensembles achieve competitive performance using 1/4th of the training data (progressively selected).



# Summary

**Creating the right datasets**

- Synthetic to real

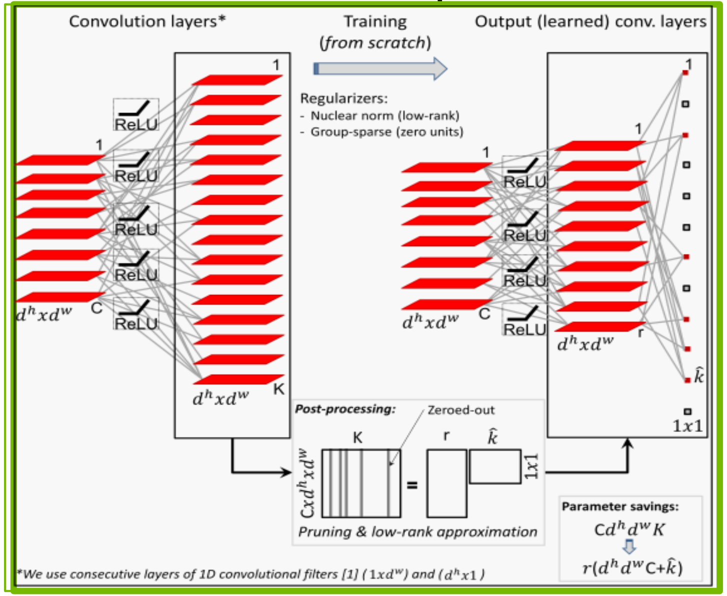


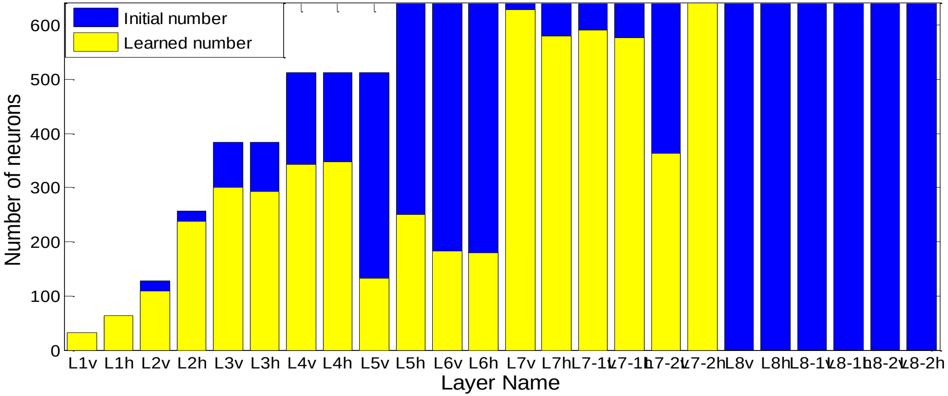
# Summary

Creating the right datasets

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

- Joint train and prune



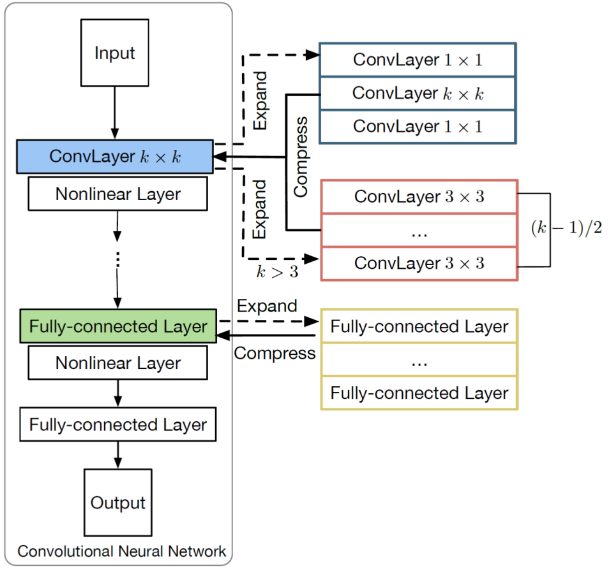



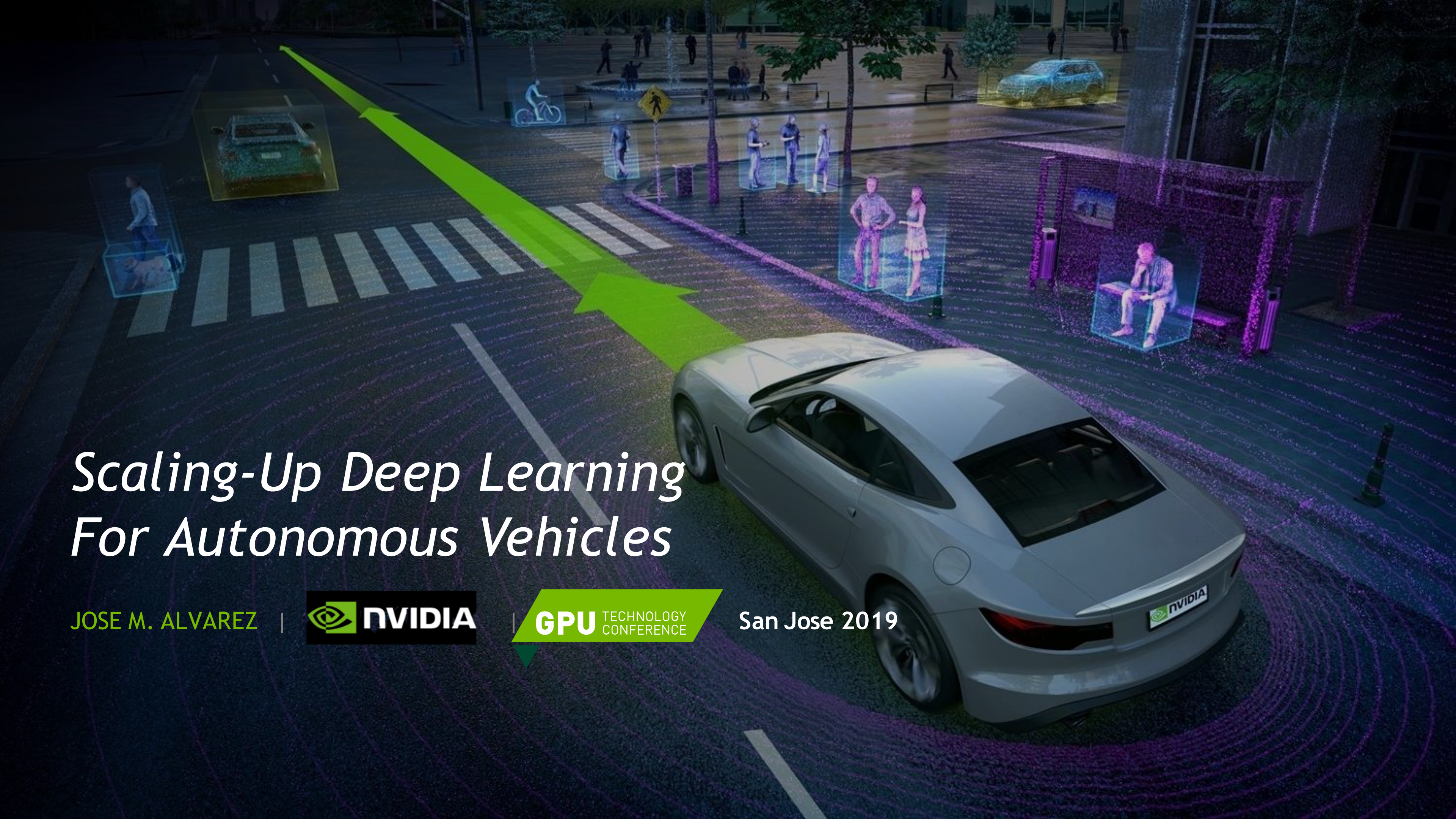
# Summary

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