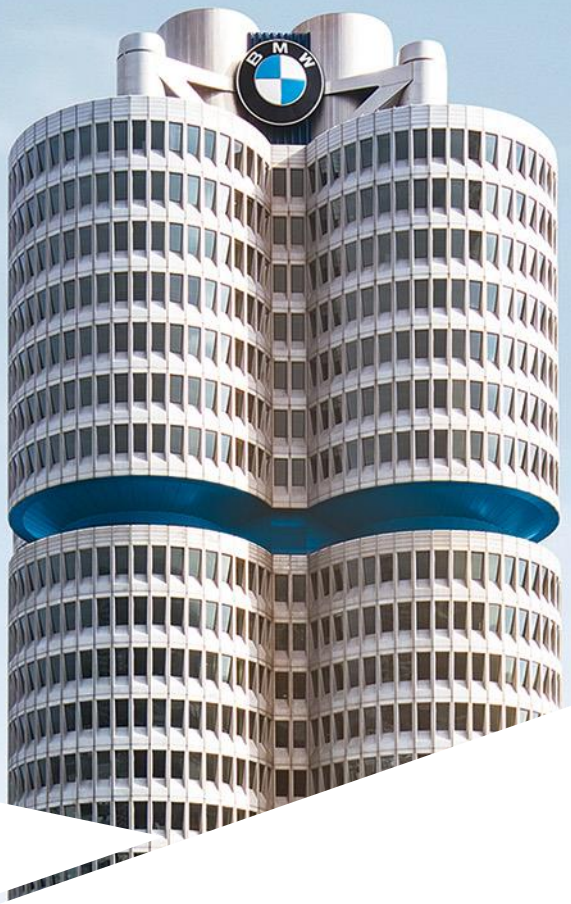
DEEP LEARNING IN THE FIELD OF AUTONOMOUS DRIVING

AN OUTLINE OF THE DEPLOYMENT PROCESS FOR ADAS AND AD



Alexander Frickenstein, 3/17/2019

BMW GROUP THE NEXT 100 YEARS

AUTONOMOUS DRIVING AT BMW

- BMW Autonomous Driving Campus in Unterschleißheim (Munich), established in 2017
- 1400 Employees incl. Partners (Sensor-processing, Data-Analytics, ML, Driving-Strategy, HW-Architecture)
- 81 Feature teams (incl. Partners), working in 2 weekly sprints (LESS)
- 30 PhDs

‘Raw data are good data’ -Unknown Author-

- BMW AD research fleet consist of 85 cars collecting 2TB/h per car
  → High resolution sensor data, like LIDAR, Camera

►**Insight into three PhD-projects, which are driven by the AD strategy at BMW**

CONTENT

- **Introduction:** Design Process of ML-Applications for AD
- **Exemplary projects;** which are driven by the AD strategy at BMW

1. Fine-Grained Vehicle Representations for AD
2. Self-Supervised Learning of the Drivable Area of AD
3. CNN Optimization Techniques for AD

DESIGN PROCESS OF ML-APPLICATIONS FOR AD

DESIGN PROCESS OF ML-APPLICATIONS FOR AD

- An rough outline of the deployment* process for ADAS and AD

- Inspired by **Gajski-Kuhn chart** (or *Y diagram*) [1]
- Design of real-world applications include:
  - Multiple domains (structural, modelling, optimization)
  - Abstraction levels
  - Knowledge sharing is essential for the drive of inovation (e.g. Car manufactures, technology companies)

*Presented projects gives an academic insight of PhD-candidates

*Datasets shown here are not used for commercial purpose

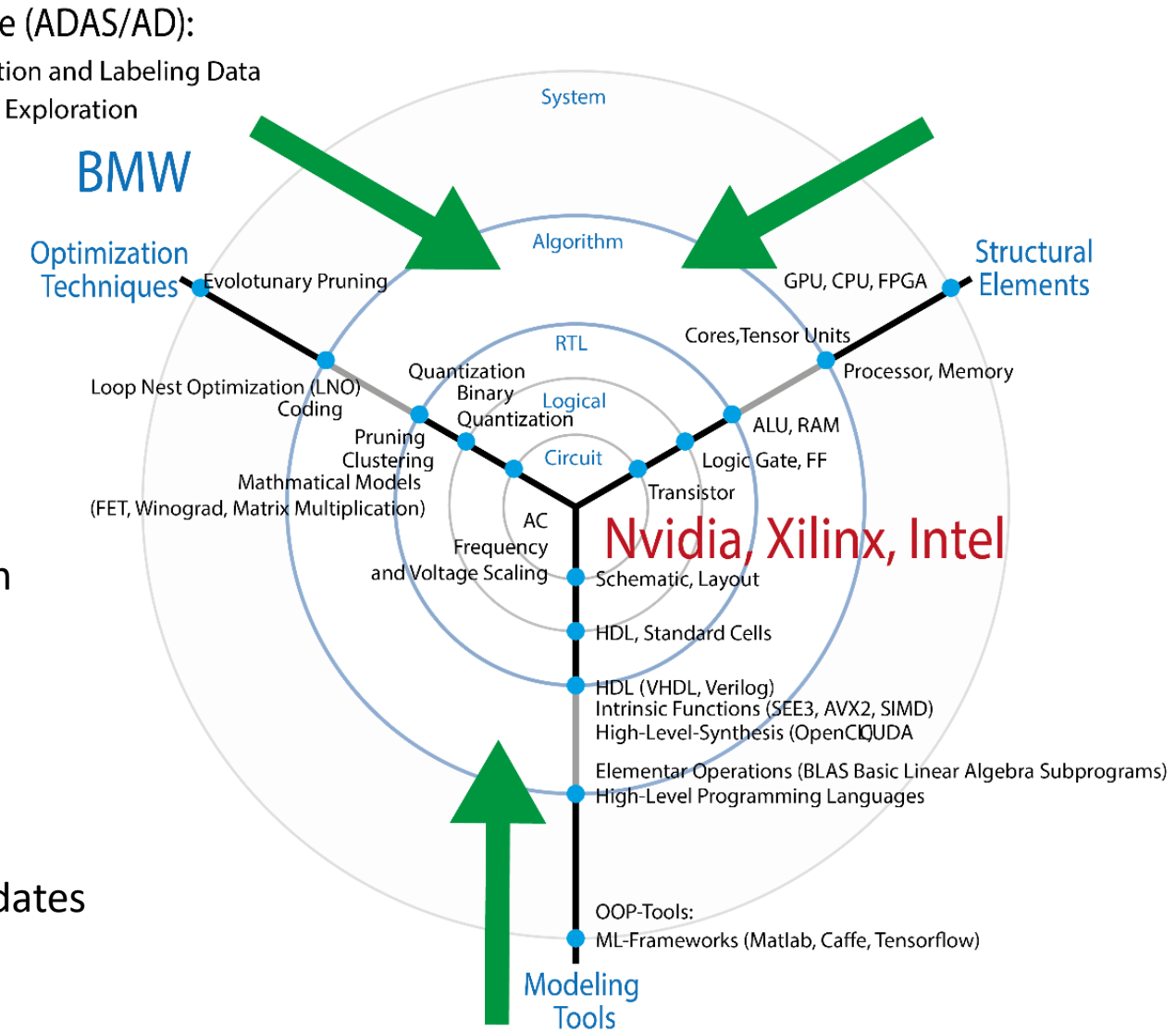


Fig. 1: Design Process of AD-Applications.

01

FINE-GRAINED VEHICLE REPRESENTATIONS FOR AD

BY THOMAS BAROWSKI, MAGDALENA SZCZOT AND SEBASTIAN HOUBEN

FINE-GRAINED VEHICLE REPRESENTATIONS FOR AD

BY THOMAS BAROWSKI, MAGDALENA SZCZOT AND SEBASTIAN HOUBEN

- **Motivation**: a detailed understanding of complex traffic scenes
  - State and possible intentions of other traffic participants
  - Precise estimation of a vehicle pose and category
  - Be aware of dynamic parts, e.g. Doors, Trunks
  - React fast and appropriate to safety critical situations

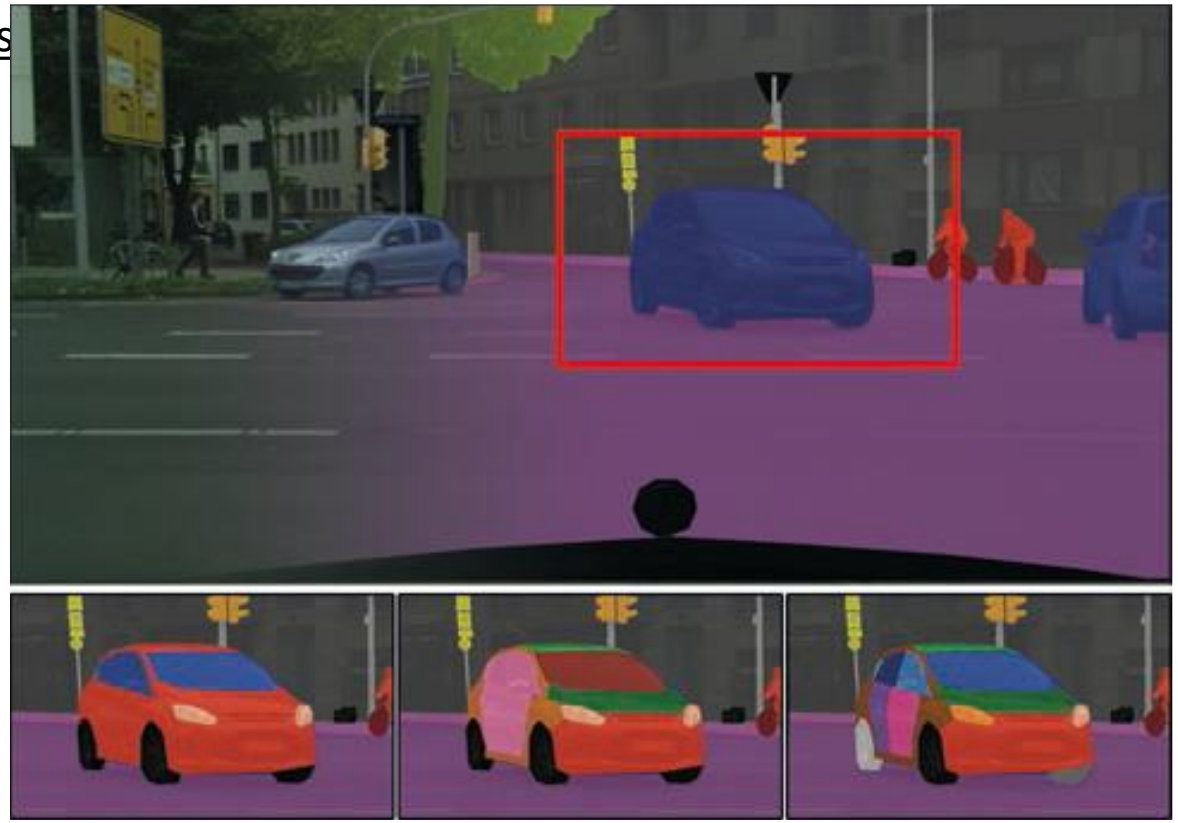


Fig. 2: Exemplary 2D visualization of fragmentation levels in the Cityscapes[3] segmentation benchmark.

Thomas Barowski, Magdalena Szczot and Sebastian Houben: Fine-Grained Vehicle Representations for Autonomous Driving, ITSC, 2018, 10.1109/ITSC.2018.8569930.

FINE-GRAINED VEHICLE REPRESENTATIONS FOR AD

- **Goal:**
  - Learn new vehicle representations by semantic segmentation
  - Three vehicle fragmentation levels (Course → Fine → Full):
    - Dividing vehicle into part areas, based on materials and fun
    - Embedding pose information
    - Annotating representations on CAD-Models
  - Empirically examined on VKITTY[4], Cityscapes[3]
- Core idea is to extend an existing image-dataset by manual labe
- Data generation pipeline is an adaption of the semi-automated method from Chabot et al. [5]
- Instead, annotation is done on a set of models (3D Car Models)

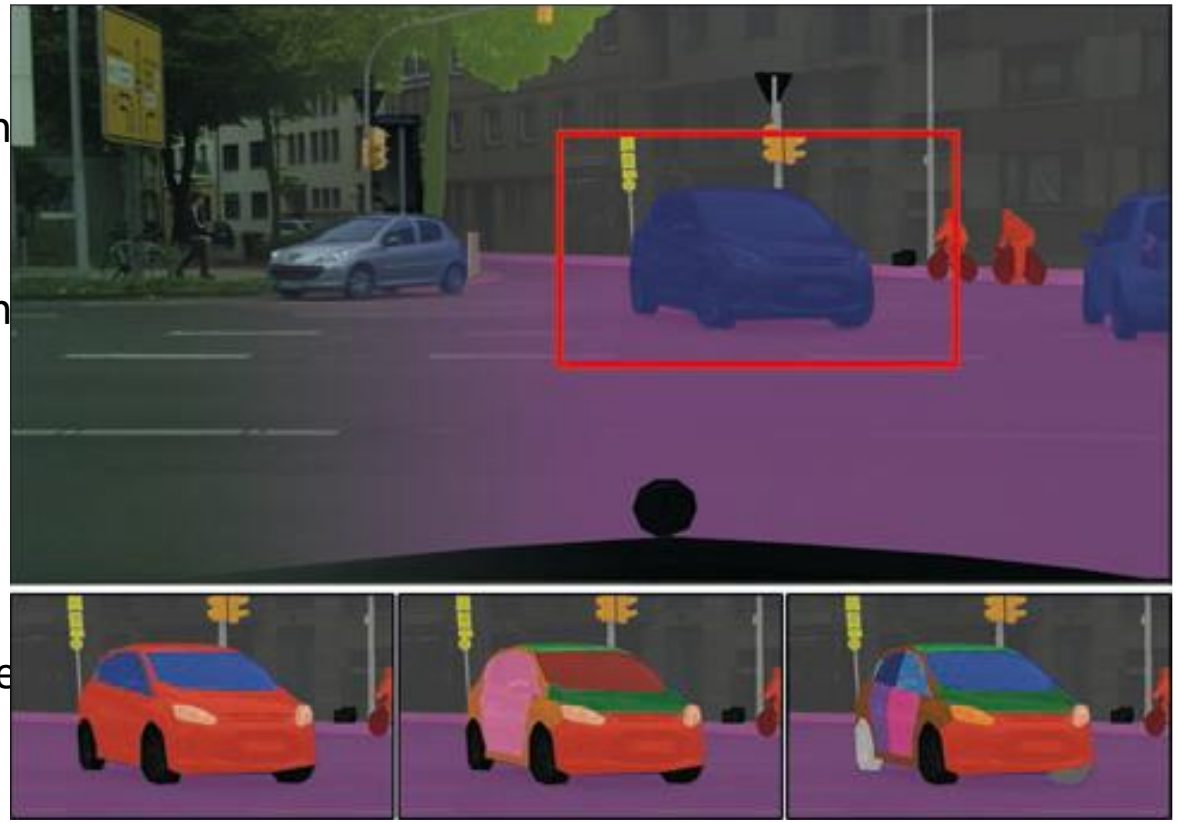


Fig. 2: Exemplary 2D visualization of fragmentation levels in the Cityscapes[3] segmentation benchmark.

SEMI-AUTOMATED LABELING PIPELINE (1)

- Vehicle Fragmentation Levels from ShapeNet (3D Model Repository):
  - Different car models including WorldNet synsets (>4000)
  - Three fragmentation levels (Coarse (4) – Fine (9) – Full (27))
  - Including classes for: Body, windows, lights, wheels, doors, roof, side, trunk, wheels, windshiels
  - In finer grained representations: model needs to solve challenging task of separation between parts that share visual cues but vary in position, e.g. individual doors
  - Identify parts with small local visual context: representation becomes suitable for pose estimation with high occlusion or truncation

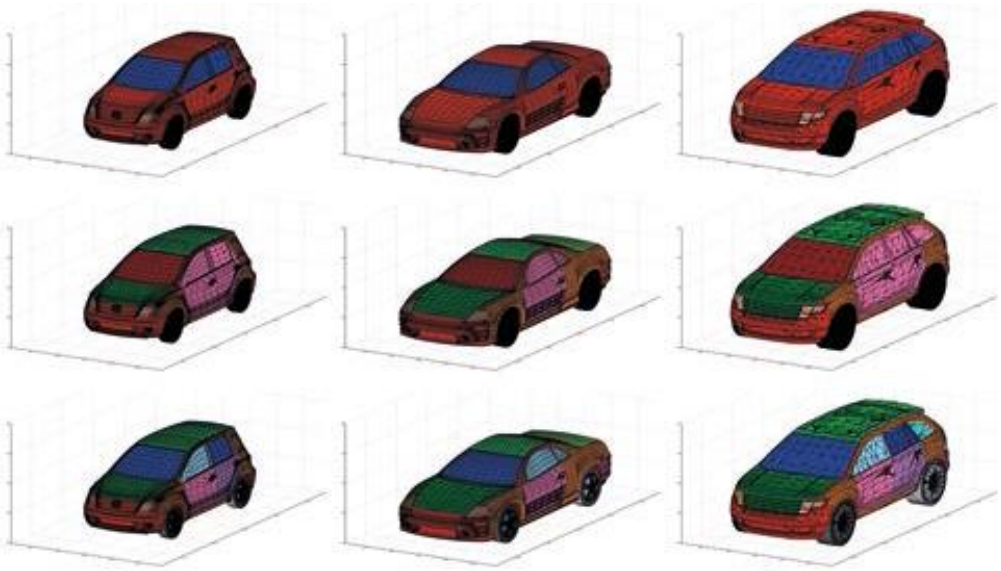


Fig. 3: Visualization of the annotated CAD models [5] .

SEMI-AUTOMATED LABELING PIPELINE (2)

1. Apply well overlapping 3D bounding boxes to raw images
2. Suited model is selected based on the vehicle type or dimensions of the model (L1-distance)
3. Mesh of the 3D-model is resize to fit the bounding box and aligned to 3D space
4. Mesh is projected on the image plane:

   → Resulting in a segmentation map containing fragmentation level information of the vehicle

5. Only pixels labeled as vehicle in the respective dataset are propagated to the image

   → To overcome projection errors

→ **Results in fine-grained dense representations**

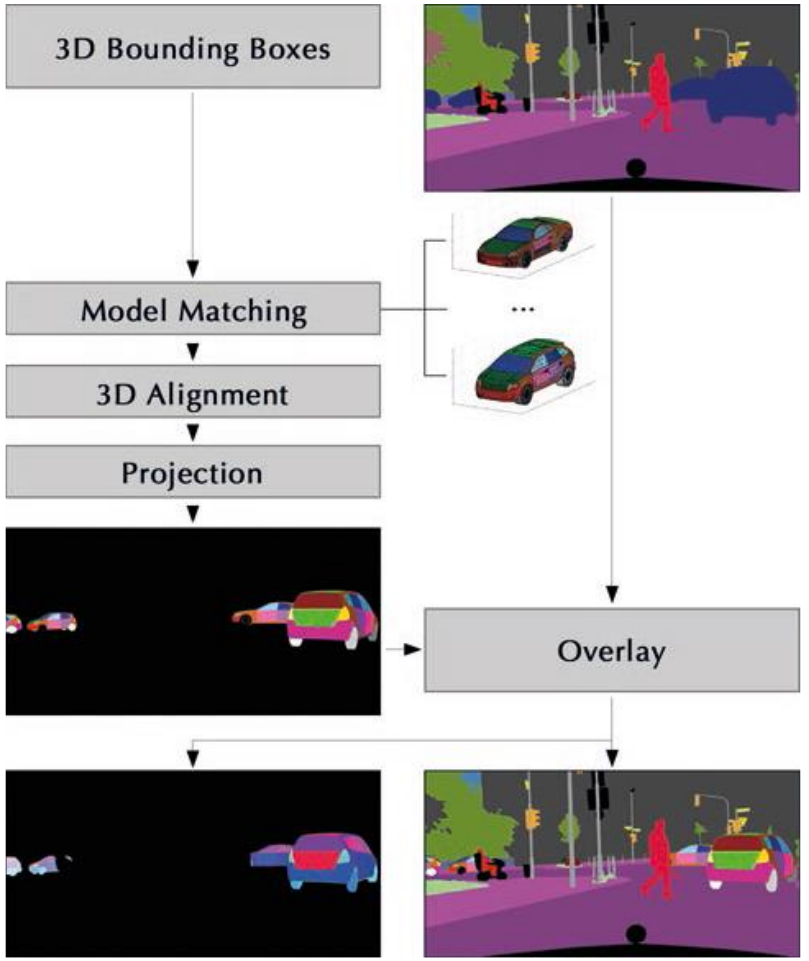


Fig. 4: Semi-automated labeling pipeline.

FCN MODEL EXPLORATION

- Reimplemented FCN8 [6] and VGG16 [7] as backbone
- End to end training, using cross entropy loss
- Trained on 4-27 classes (based on fragmentation level)
- Plus classes of datasets
- Multi-GPU training (Kubernets and Horovod on DGX1)

  → Full fragmentation level → High resolution input images
- **Aim:** not loosing significant accuracy in non vehicle-related background classe

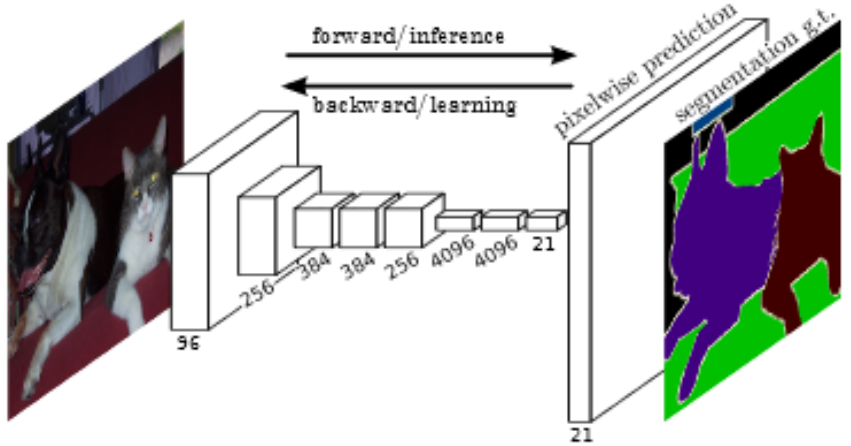


Fig. 5: FCN8 [6] with VGG16[7] backbone.

FCN MODEL EXPLORATION

Experiment		IoUclass	IoUnon-parts	IoUparts
VKITTY Baseline		68.77	**68.77**	-
VKITTY ShapeNet [15]	Coarse	61.05	66.49	**63.64**
	Fine	56.93	66.31	44.73
	Full	36.67	58.22	27.44
Cityscapes ShapeNet [15]	Coarse	49.56	48.81	**52.96**
	Fine	48.63	50.88	44.20
	Full	33.50	50.78	21.98

Tab. 1: Segmentation results for the three fragmentation levels, performed on VKITTY and Cityscapes using FCN8.

FCN MODEL EXPLORATION – VKITTY AND CITYSCAPES

Coarse: Fine: Full:

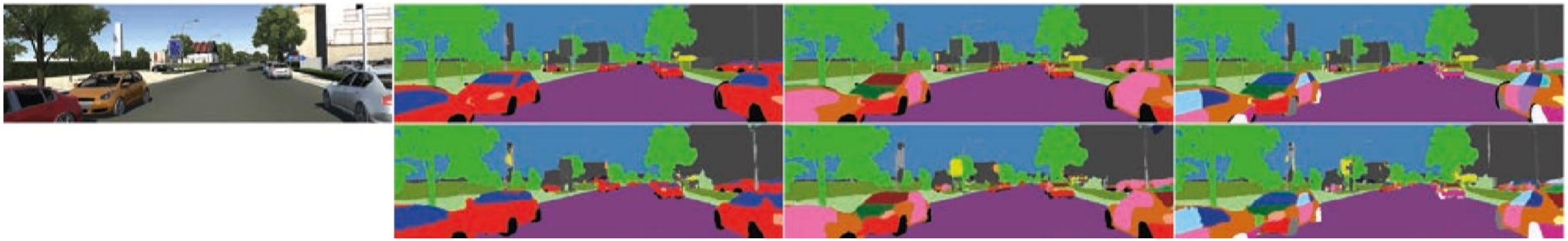


Fig. 6a: Qualitative results on VKITTY dataset for the three fragmentation levels.

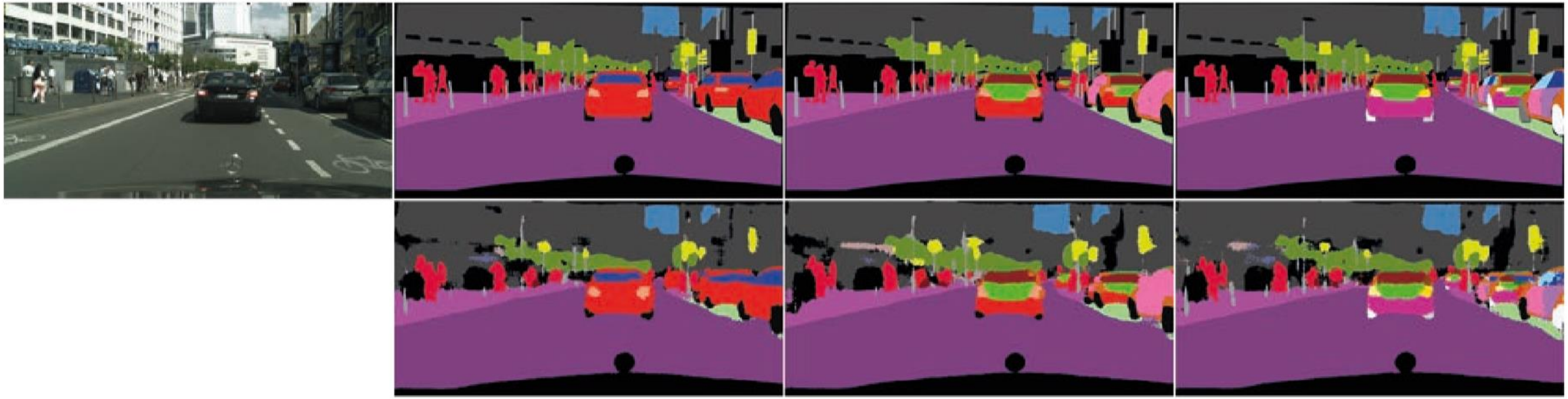


Fig. 6b: Qualitative results on Cityscapes dataset for the three fragmentation levels.

02

SELF-SUPERVISED LEARNING OF THE DRIVABLE AREA OF AD

BY JAKOB MAYR, CHRISTIAN UNGER, FEDERICO TOMBARI

SELF-SUPERVISED LEARNING OF THE DRIVABLE AREA OF AD

- **Motivation:**
  - Automated approach for generating training data for the task of drivable area segmentation → Training Data Generator (TDG)
  - Acquisition of large scale datasets with associated ground-truth still poses an expensive and labor-intense problem

Jakob Mayr, Christian Unger, Federico Tombari: Self-Supervised Learning of the Drivable Area for Autonomous Driving, iROS, 2018.

- Deterministic stereo-based approach for ground-plane detection:

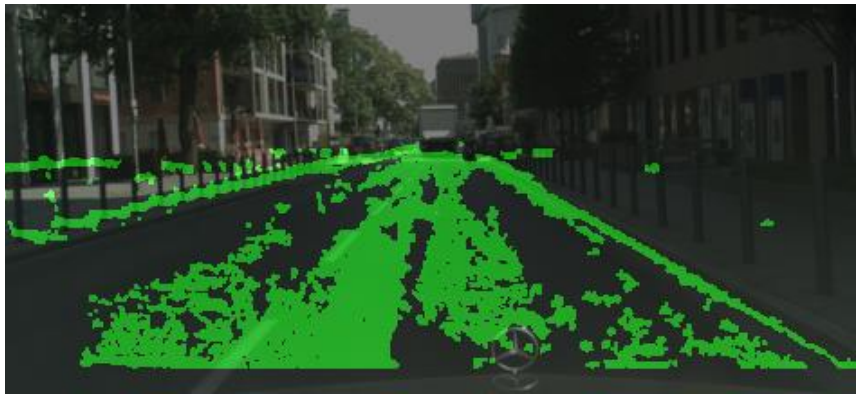


Fig. 7a: Automated generated data of TDG.

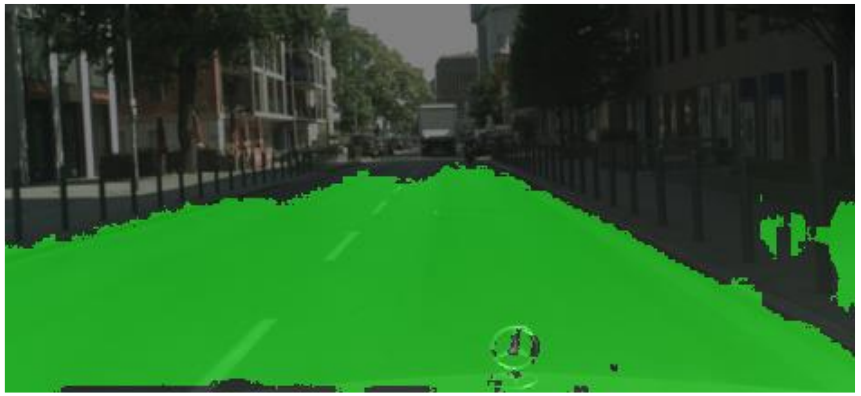


Fig. 7b: Segmentation of DNN trained on TDG.

WHY GROUND-PLANE DETECTION?

- Important aspect is the planning of safe and comfortable driving maneuvers
- Knowledge about the environment of the vehicle
- especially drivable areas (important role in ADAS and AD)
- e.g. road ahead/ drivable area is blocked by obstacles
- Parallel processing of GPUs allow frame based semantic segmentation
- Why Automated Data-Labeling?
  - Pace and cost pressure
  - Labeling is expensive
  - Existing datasets do not suit the desired application:
    - Technical aspects: e.g. field of view, mounting position, camera geometry
    - Environmental conditions: e.g. weather condition, time, street types

Technical Aspect of Cityscapes: images show part of the hood, initialization of the ground-plane model including non-ground plane disparity is necessary!

AUTOMATED LABELING PIPELINE

- Based on so-called v-disparity map [8]:
  - Different use cases
  - No fine tuning of existing models required

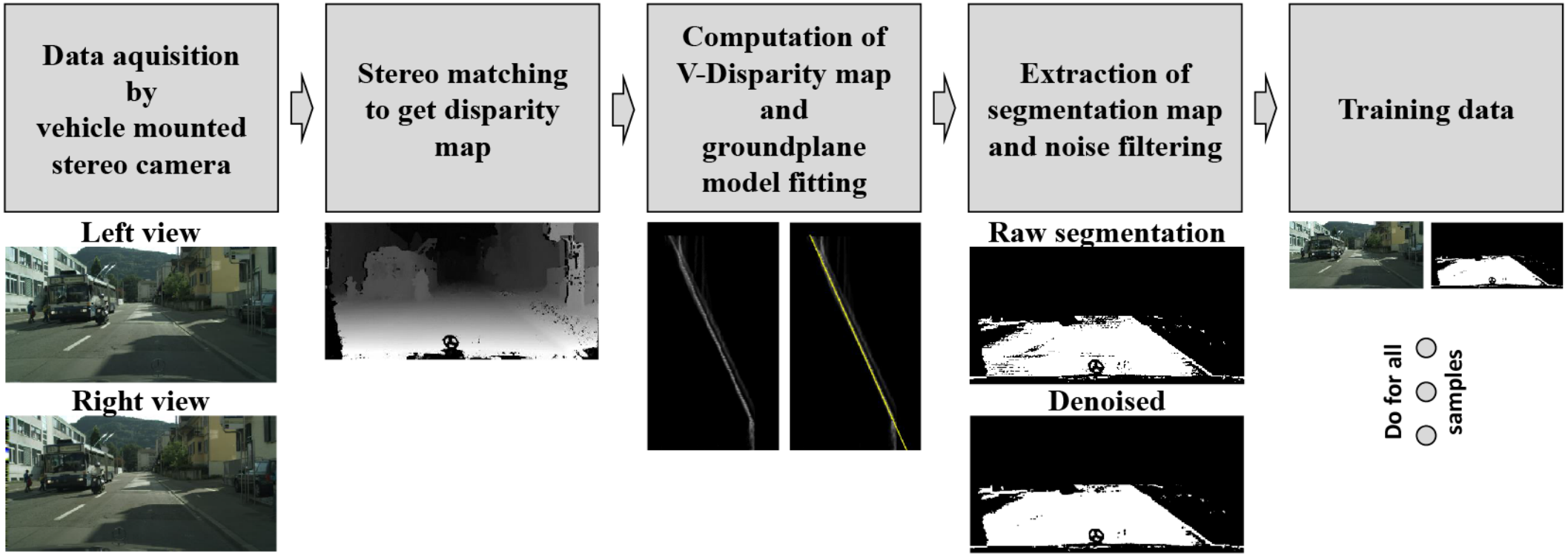


Fig. 8: Automated labeling pipeline.

CNN-MODEL EXPLORATION (1)

- Automatically generated data are used to train Unet and SegNet → low resolution inputs (512*256 and 480*360)
- Models are trained only on automatically generated datasets
- Evaluation is performed by using human labeled ground-truth data, e.g. Cityscape [3], Kitty [2]
  → Drivable (road, parking) and non-drivable area (side walks, pedestrians)
- **Observations:**
  - Low detection in lateral direction
  - Noisy data of TDG → generate robust CNN model
  - Dynamic objects are detected reliably

Fig. 9a: SegNet segmentation.

Fig. 9b: U-Net segmentation.

CNN-MODEL EXPLORATION (1)

- **Robustness of model - Flipping images upside down:**
  - SegNet [9] → Not capable of detecting ground-plane
  - UNet [10] → Detects ground-plane

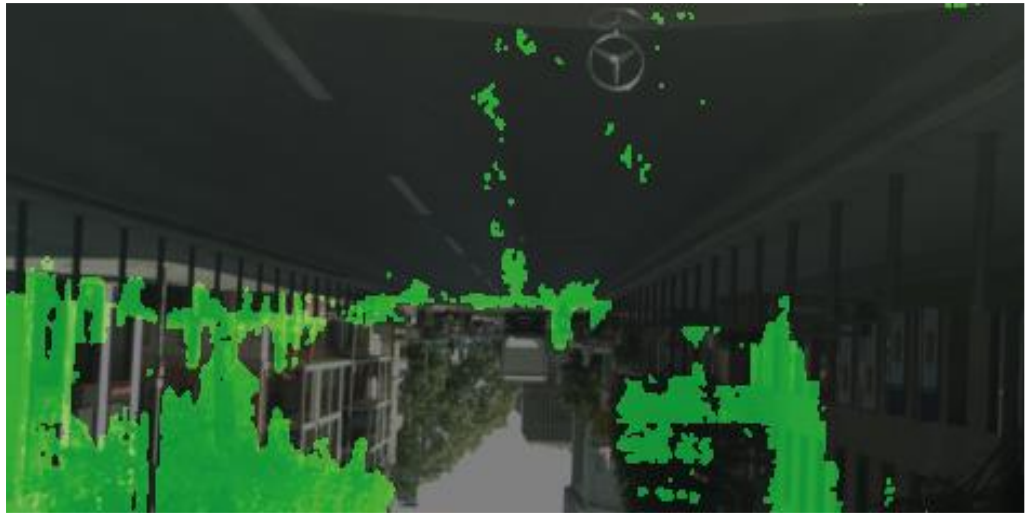


Fig. 9c: Flipped SegNet segmentation.

Fig. 9d: Flipped U-Net segmentation.

CNN-MODEL EXPLORATION (2)

	Cityscapes				KITTY			
Approach	**Rec**	**Prec**	**Acc**	**IoU**	**Rec**	**Prec**	**Acc**	**IoU**
Training Data Generator (TDG)	70.84	91.49	85.35	66.46	81.87	58.07	86.58	51.45
Unet trained on auto. TDG labels	85.29	**92.83**	**91.27**	**80.01**	87.25	**81.35**	**94.31**	**72.70**
SegNet trained on auto. TDG labels	**85.75**	76.10	85.29	67.56	**90.96**	70.01	91.35	65.45

Tab. 2: Segmentation results for the TDG, performed on Cityscapes [3] and Kitty [2] using U-Net [10] and SegNet [9].

- **Performance:**
  - Data generation:
    - Hand labeling Cityscape [3] 19d
    - Using automated labeling 3.5h (CPU) not parallelized on GPU yet
  - U-Net [10] : 10 fps on Titan X
  - ResNet [9].:4.4fps on Titan X

→ Optimization of DNNs comes into account

→ Out of the box CNNs come along with substantial drawbacks

03

CNN OPTIMIZATION TECHNIQUES FOR AD

BY ALEXANDER FRICKENSTEIN, MANOJ VEMPARALA, WALTER STECHELE

CNN OPTIMIZATION TECHNIQUES FOR AD

- Running example: Quantization of CNNs:
  - Normally, floating-point PEs is **10× less** energy efficient compared to fixed point math.
  - The step-size between two numbers could be dynamic using floating-point numbers. This is useful feature for different kinds of layers in CNN.
  - Closing gap between CNN compute demand and HW-accelerator is important

    → Trend to specialized HW-accelerator, e.g. Tesla T4

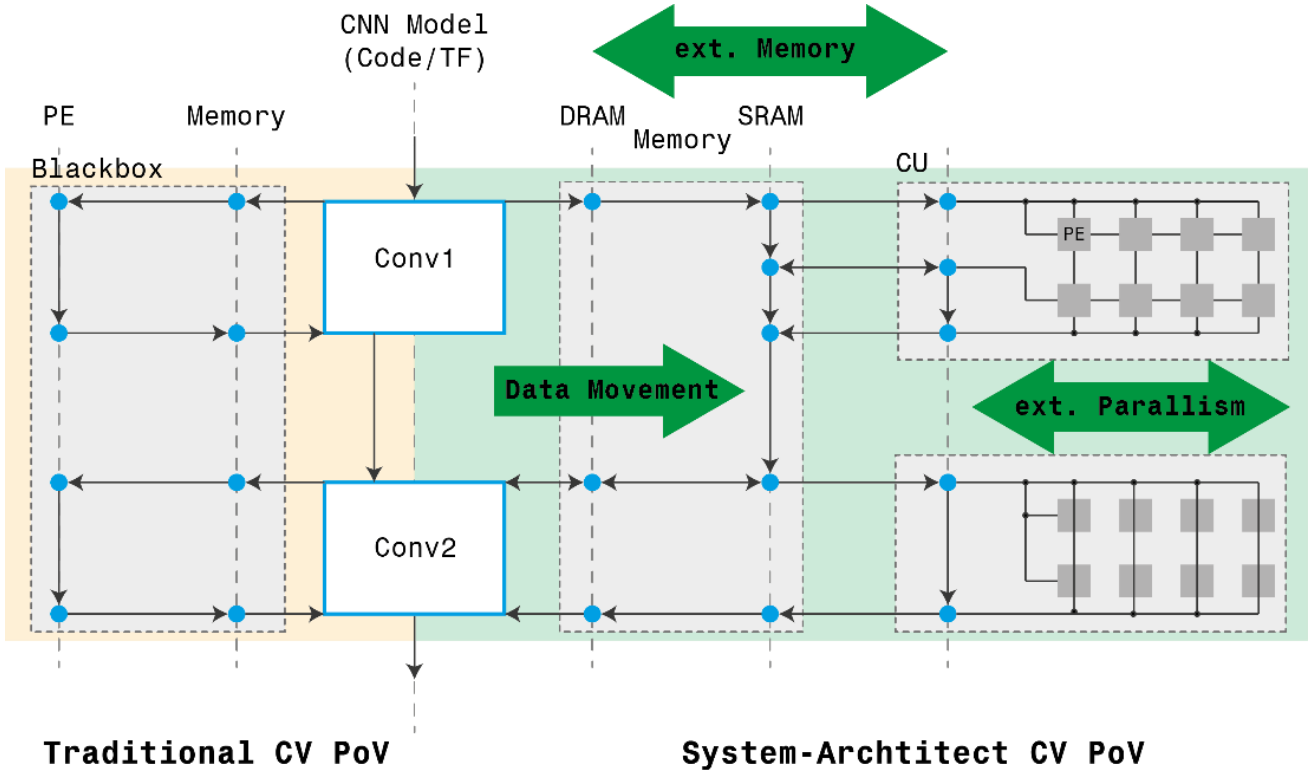


Fig. 10: How deployment of DNNs can be seen differently.

CNN OPTIMIZATION TECHNIQUES FOR AD

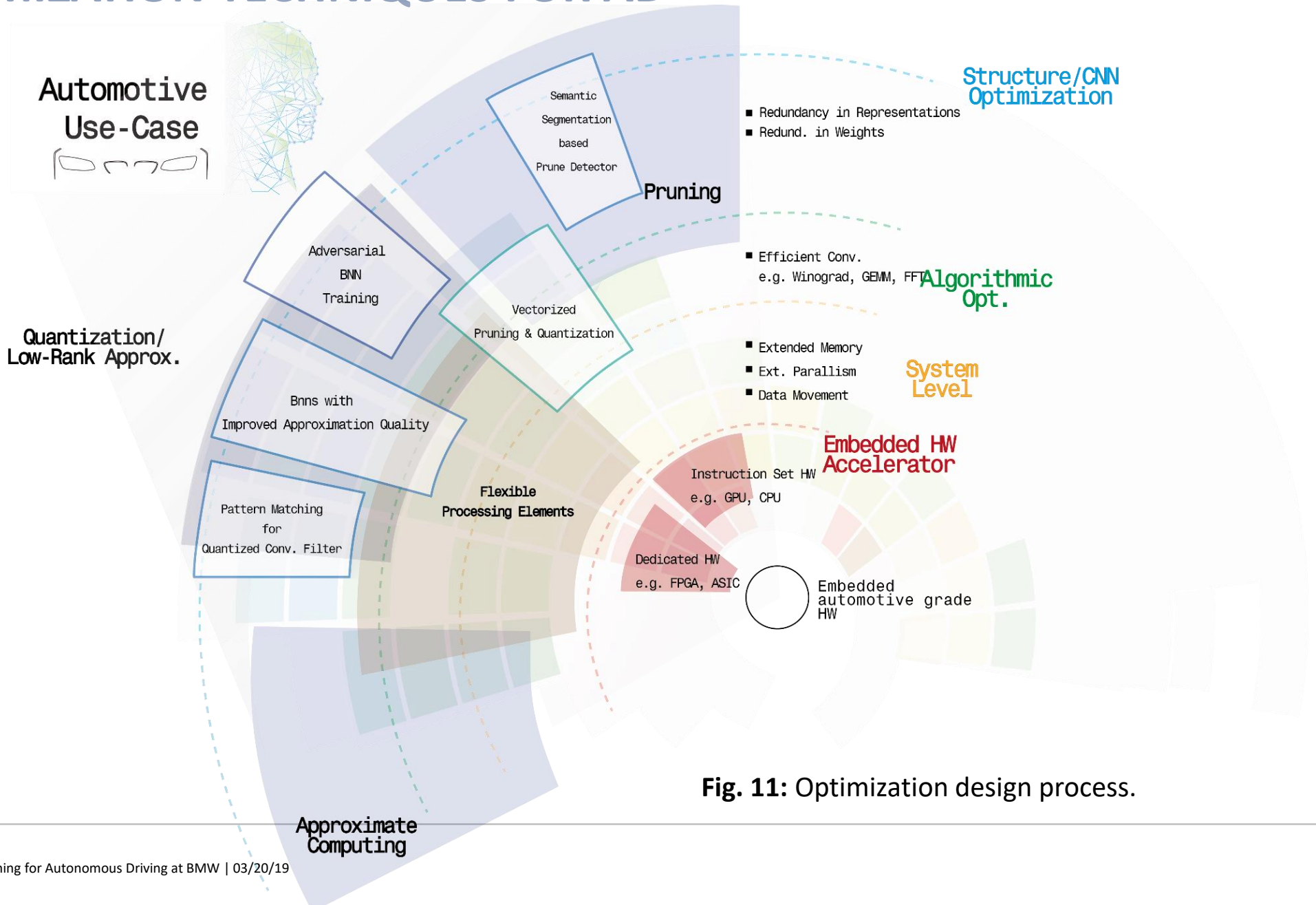


Fig. 11: Optimization design process.

RESOURCE AWARE MULTICRITERIAL OPTIMIZATION

- **Motivation:**
  - Out of the box DNNs require high performance HW-accelerator:
    - YOLO [11] or SSD300 [12] require an Nvidia Titan X to run in real-time
      → Showing the high compute demand of those models
    - **No**, SqueezeNet [13] is really not out of the box!
      → An 18,000 GPU super-computer is used for the model exploration
  - Deploy DNNs on memory, performance and power-consumption constraint embedded hardware is commonly time consuming

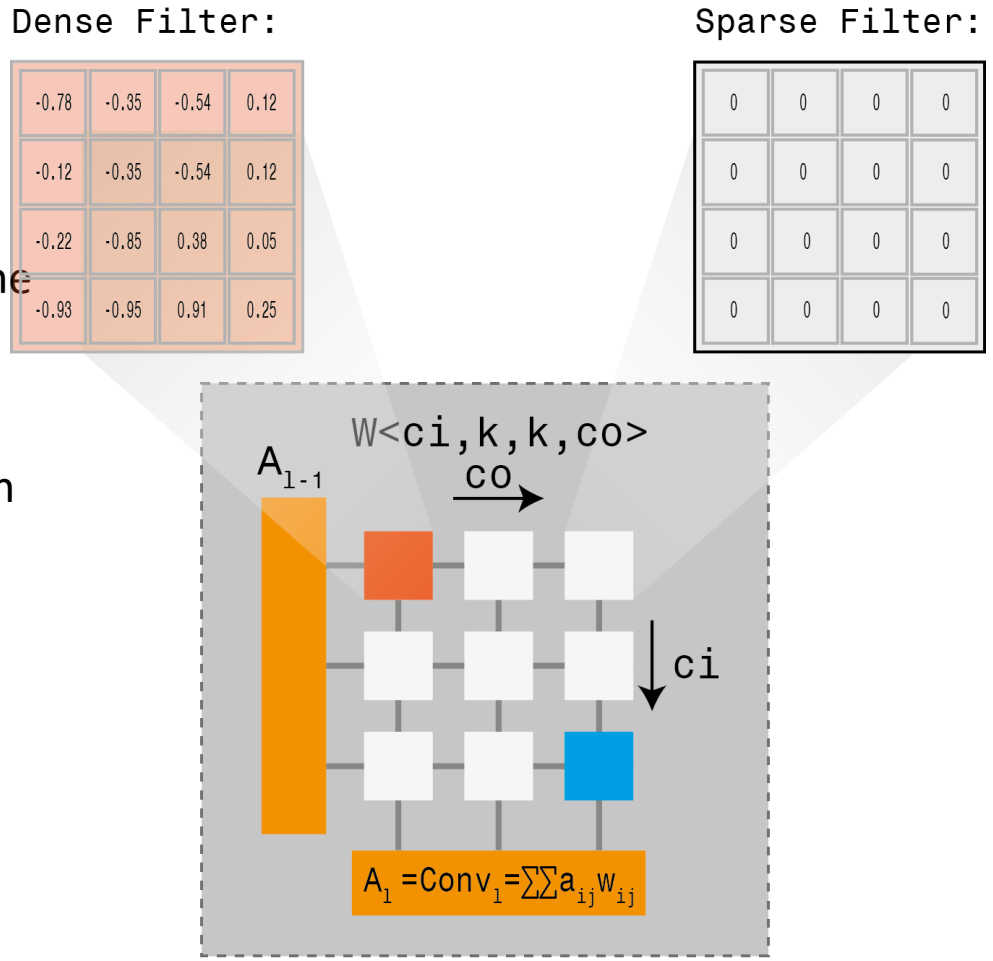


Fig. 11: Filter-wise pruning of convolutional layer.

WHY RESOURCE-AWARE MULTICRITERIAL OPTIMIZATION?

- Optimization techniques are going hand in hand (Filter-wise pruning and Quantization)
- CNN optimization depends on system, algorithmic and system level in the design process
- DNNS need to be highly compressed to fit the HW for AD
  → Automotive rated memory is expensive

- Good data locality is essential for low-power applications
  → Extreme temperatures in cars (Siberia → Death Valley)
  → Active cooling obligatory?

- Fast deployment time is a crucial aspect for agile SW deployment

→ Proposing a Pruning and Quantization scheme for CNNs

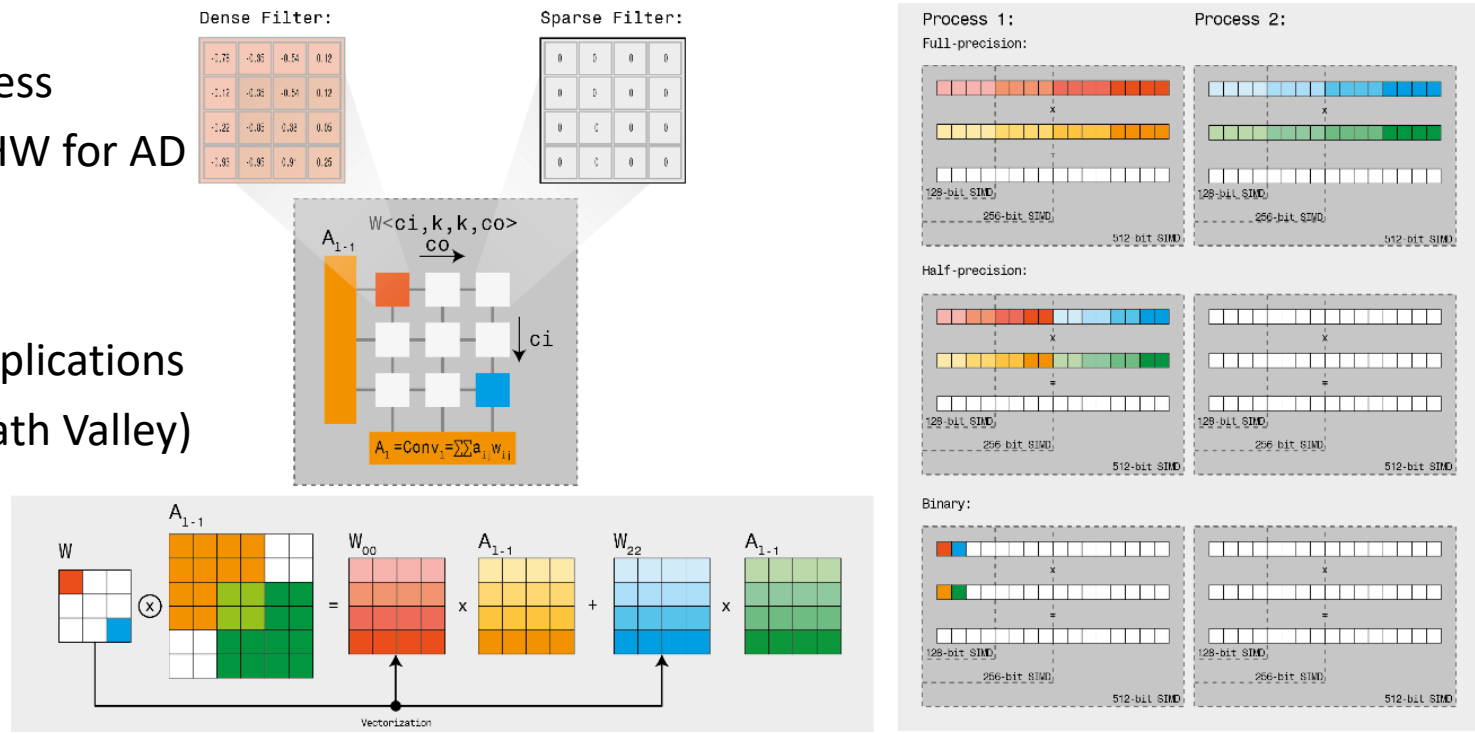


Fig. 12: Pruning and quantization for efficient embedded Applications of DI

METHOD -PRUNING

- Fast magnitude based [14] pruning scheme → removing unimportant filter
- Based on a half-interval search (log2(n))

  → Explore optimal layer-wise pruning rate
- Varying pruning order to generate an optimized model either with respect to the memory demand or execution time
- **Process of removing weight-filter of a layer:**
  1. Identify Cost (L1-distance) of all weight-filter of a layer
  2. Based on the half-interval search remove filter, which cost is below a threshold
  3. Threshold is identified by half-interval search
  4. Retrain model (SGD with momentum) with small learning rate

     → Momentum should be available before pruning
  5. As accuracy of the CNN is maintained increase the pruning rate (half-interval search)
  6. Pruning is always applied to the layer which fits the desired optimization goal best

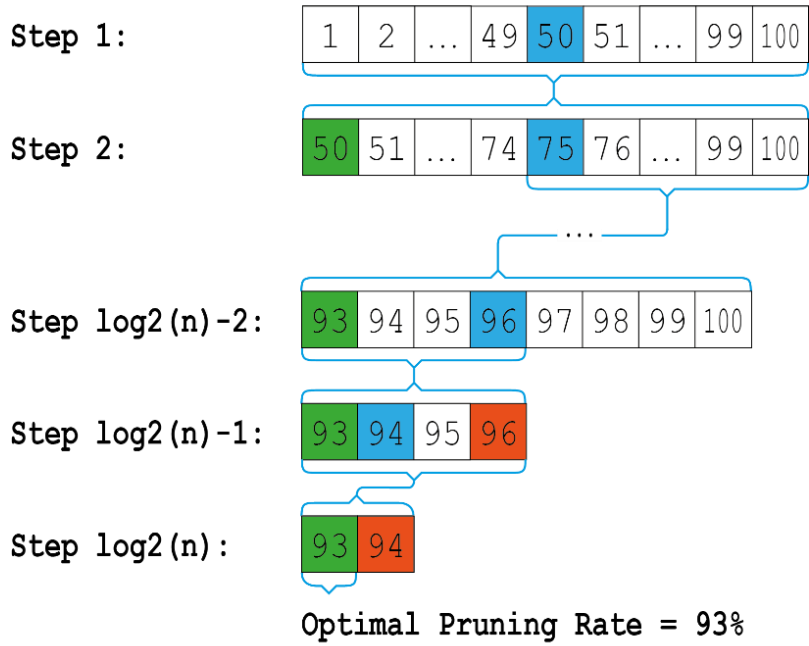


Fig. 13: Binary-search applied to prune CNN.

METHOD - QUANTIZATION

- Quantization leads to a hardware friendly implementation
- Reducing the footprint of HW-components
- Lowering the memory bandwidth
- Improving the performance
  → Floating-point PE is **10x** less efficient
  compared to fixed-point unit

- Weight and activations are brought into the fixed-point format with the notation <S,IB,FB>
  - S: Sign bit
  - IB: Integer bit
  - FB: Fractional bit
- Stochastic rounding is used for approximation

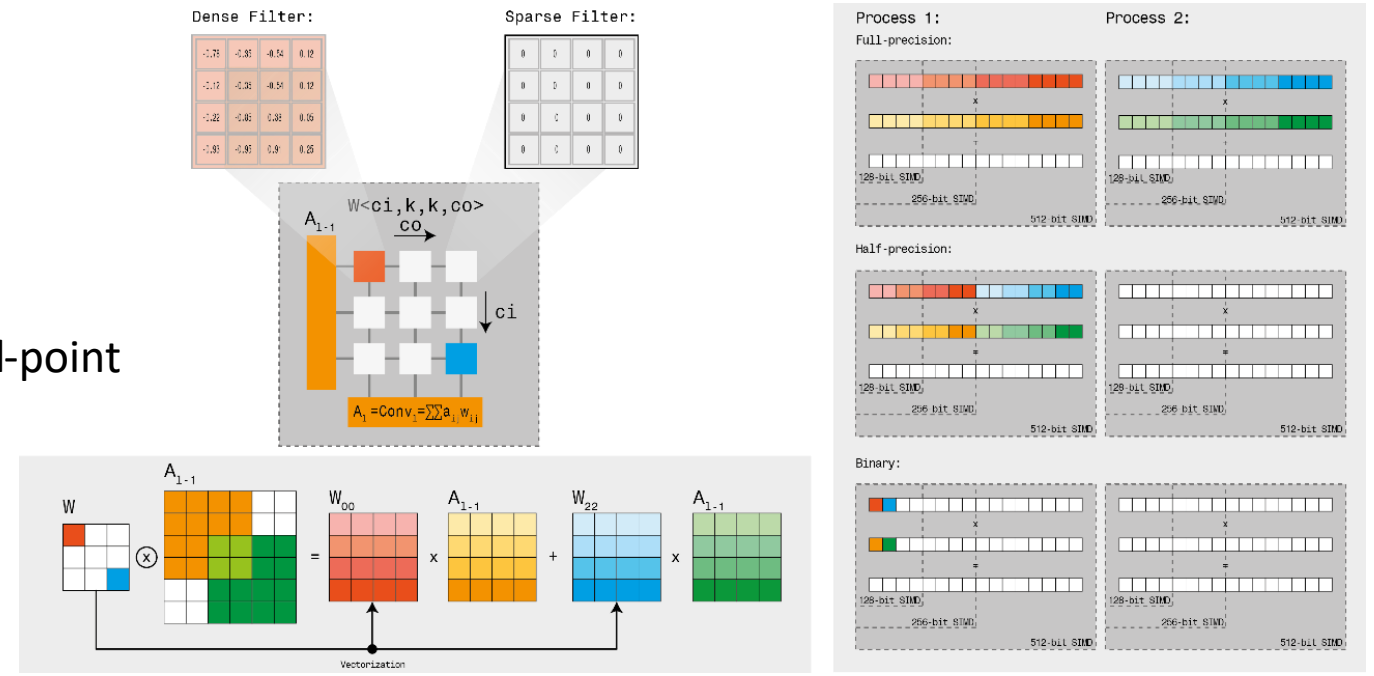


Fig. 14: Pruning and quantization applied to CNN.

MEMORY AND PERFORMANCE BENCHMARK

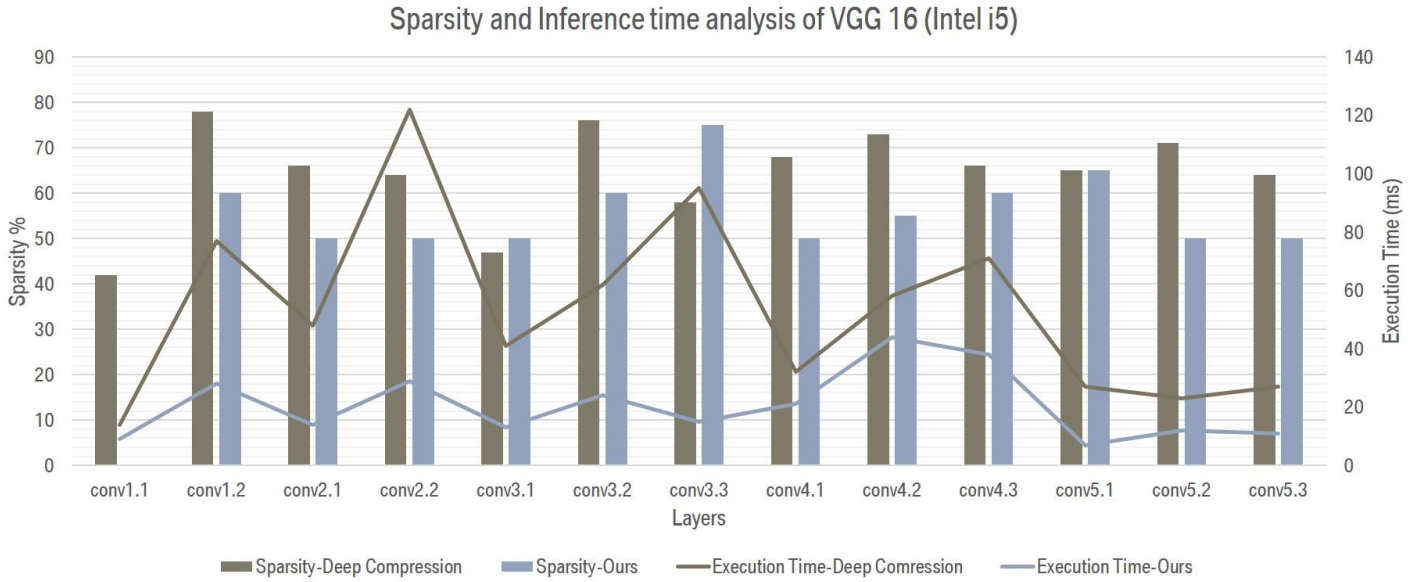


Fig. 15: Pruning rate and runtime of Deep Compression [14] and our approach.

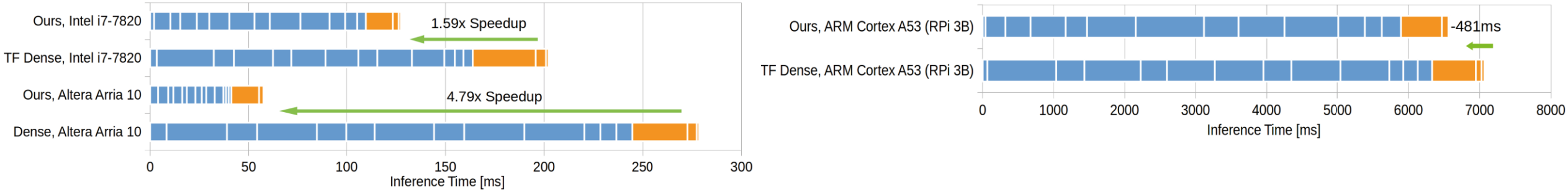


Fig. 16: Runtime of VGG16 [7] on different HW-Accelerator.

MEMORY AND PERFORMANCE BENCHMARK

Model	Top-1 acc.	Params.	Comp. Rate	MACs	Retrain [Epochs]	([Time])
LeNet reference	98.9%*	1,720kB		2.3M	~ 100	(21min)
DC [11]	99.2%	44kB	39×	415k	~ 160	(34min)
DNS [8]	99.1%	16kB	108×	91k	- - -	- - -
BC [29]	99.0%	10.5kB	361×	106k+	- - -	- - -
L-OBS [6]	98.7%	17kB	101×	78k	~ 8	(4min)
LeNet Mem. (Ours)	98.9%*	**4.4kB**	**385×**	210k	~ 112	(24min)
LeNet Perf. (Ours)	98.9%*	10.0kB	172×	**57k**	~ 112	(24min)
VGG16 reference	68.5%	138.4M		15.47B	~ 90	(7.5d)
LWC [13]**	68.7%	10.4M	13.1×	5.04B	~ 4,950	(412d)
L-OBS [6]**	62.7%	10.4M	13.1×	5.45B	~ 19	(3.2d)
DNS [8]**	36.6%+	10.4M	13.1×	- - -	~ 230	(19.2d)
VGG16 Mem. (Ours)**	68.4%	**9.5M**	**14.5×**	7.14B	~ 192	(16d)
VGG16 Perf. (Ours)**	68.5%	19.4M	7.1×	**3.54B**	~ 192	(16d)

* 5,000 images of the training set are used for validation. Otherwise, the reported accuracy is 99.1%.
** Optimization of VGG16 is limited to pruning

Tab. 3: Performance and memory benchmark of our method applied to VGG16.

REFERENCES

[1] Grout Ian: Digital Systems Design with FPGAs and CPLDs, 2008.

[2] Andreas Geiger, Philip Lenz, Raquel Urtasun, et al.: Are we ready for autonomous driving? The KITTI vision benchmark suite, CVPR, 2012.

[3] Marius Cordts, Mohamed Omran, Sebastian Ramos, et al.: The Cityscapes Dataset for Semantic Urban Scene Understanding, CVPR, 2016.

[4] Adrien Gaidon, Qiao Wang, Yohann Cabon, et al.: Virtual Worlds as Proxy for Multi-Object Tracking Analysis, CVPR, 2016.

[5] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, et al.: ShapeNet: An Information-Rich 3D Model Repository, 2015.

[6] Jonathan Long, Evan Shelhamer, Trevor Darrell: Fully convolutional networks for semantic segmentation, CVPR, 2015.

[7] Karen Simonyan, Andrew Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR, 2014.

[8] Raphael Labayrade,Didier Aubert,Jean-Philippe Tarel: Real time obstacle detection in stereovision on non flat road geometry through "v-disparity" representation, IVS, 2002.

[9] Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, IEEE Transactions on Pattern Analysis and Machine Learning, 2017.

[10] Olaf Ronneberger, Philipp Fischer, Thomas Brox: U-Net: Convolutional Networks for Biomedical Image Segmentation, Lecture Notes in Computer Science, 2015.

[11] Joseph Redmon, Santosh Divvala, Ross Girshick: YOLO: Real-Time Object Detection, CVPR, 2014.

[12] Wei Liu, Dragomir Anguelov, Dumitru Erhan: SSD: Single Shot MultiBox Detector, ECCV, 2016.

[13] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size, ICLR, 2017.

[14] Song Han, Jeff Pool, John Tran, William J. Dally: Learning both Weights and Connections for Efficient Neural Networks, CVPR, 2015.