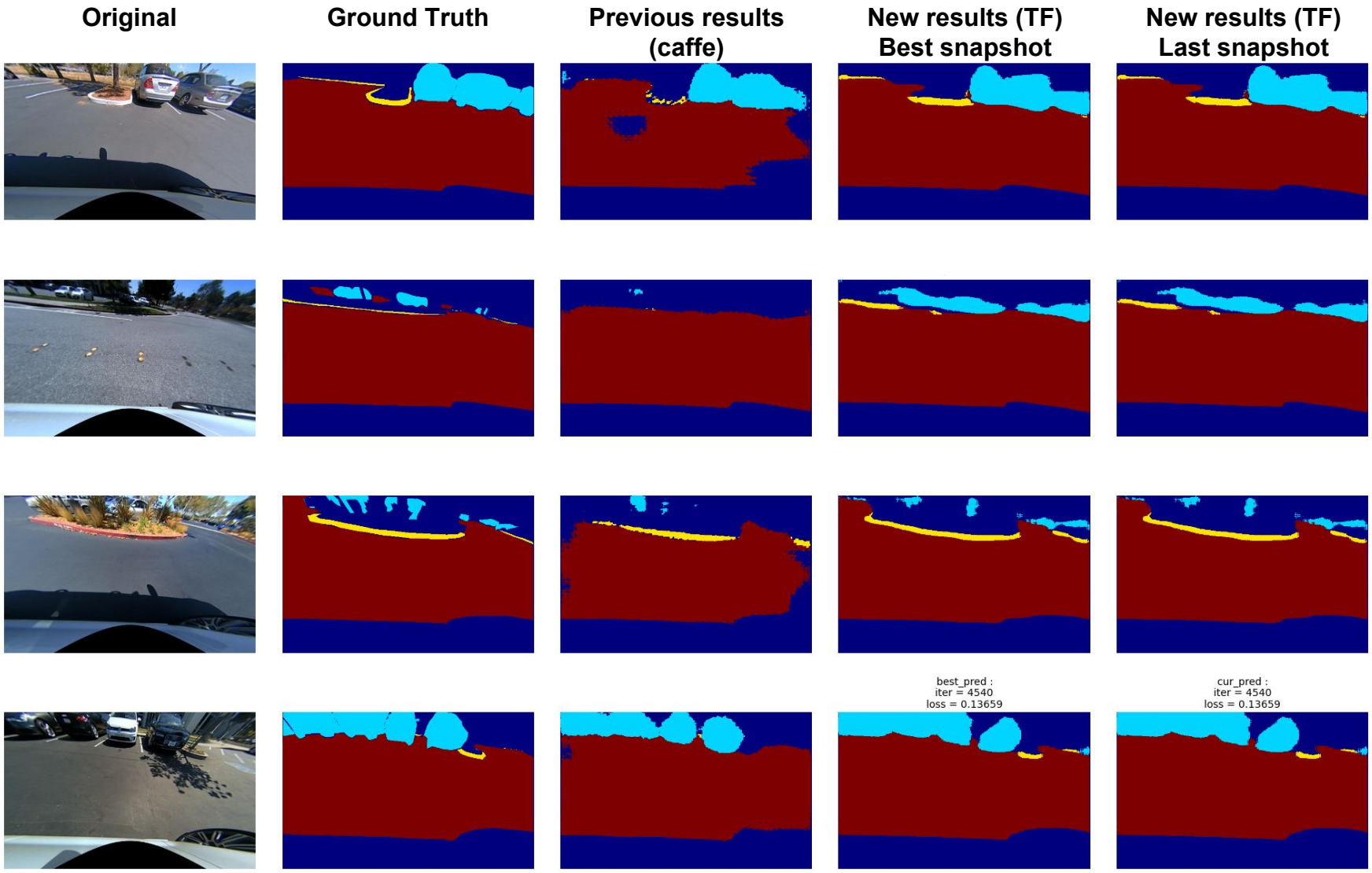
ERL PML
Deep Learning meeting
07/15/2016

Artem Molchanov

Overview

- **Architectures**
  - *Inception V1*: TF from scratch
    (baseline to compare to SqueezeNet)
  - *SqueezeNet:*
    - Design choices
    - Architecture
    - Training vs Inception V1
- **Compression**
  - *Pruning. Can it help ?*
    - Convolutions in a nutshell
    - Sparse convolutions vs Sparse-dense vs conv2d. Results
    - Discussion
  - *Quantization*
    - Overview of the toolkit
    - 8-bit quantization results
  - *Smaller SqueezeNets*
    - Architectures
    - Training results
  - *Distillation: SNet to SNetX8*
  - *Next steps*

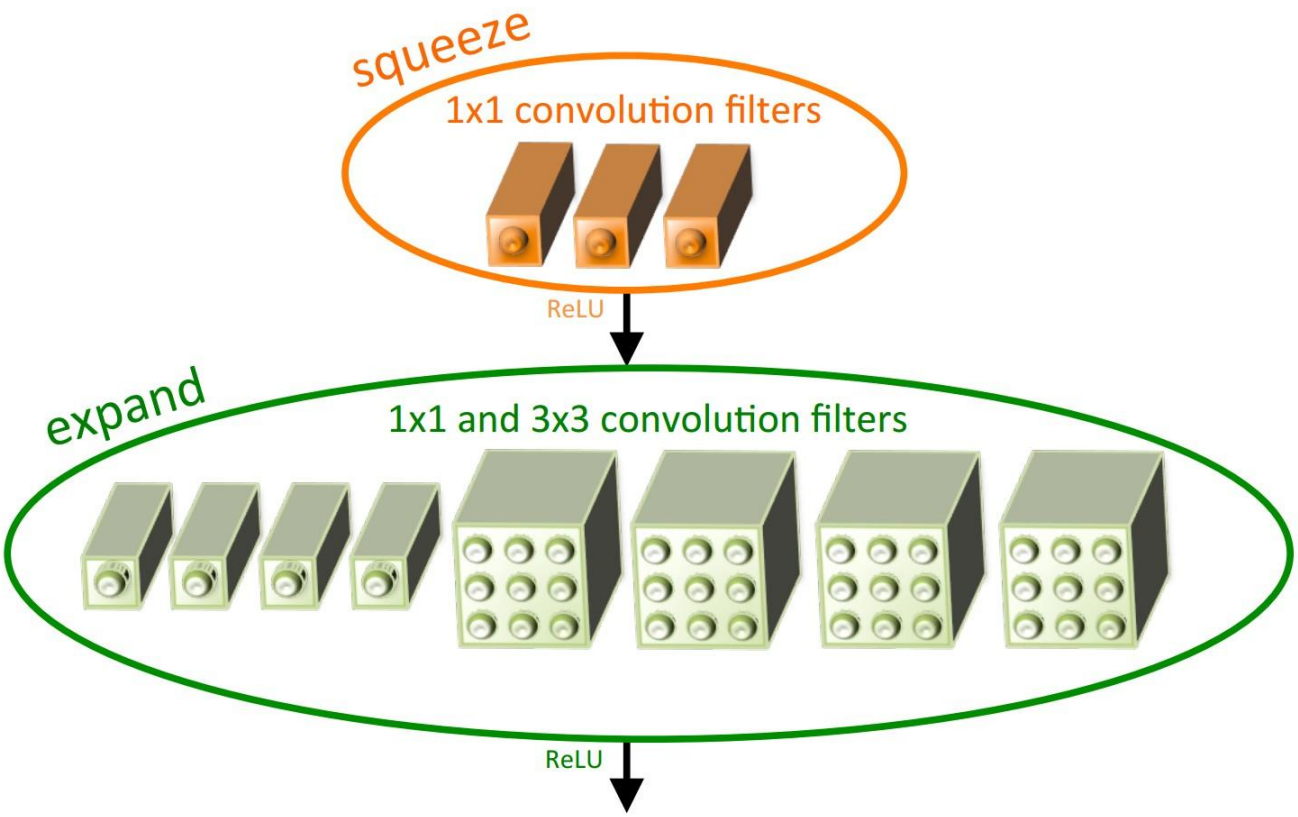
Inception V1: TF training from scratch



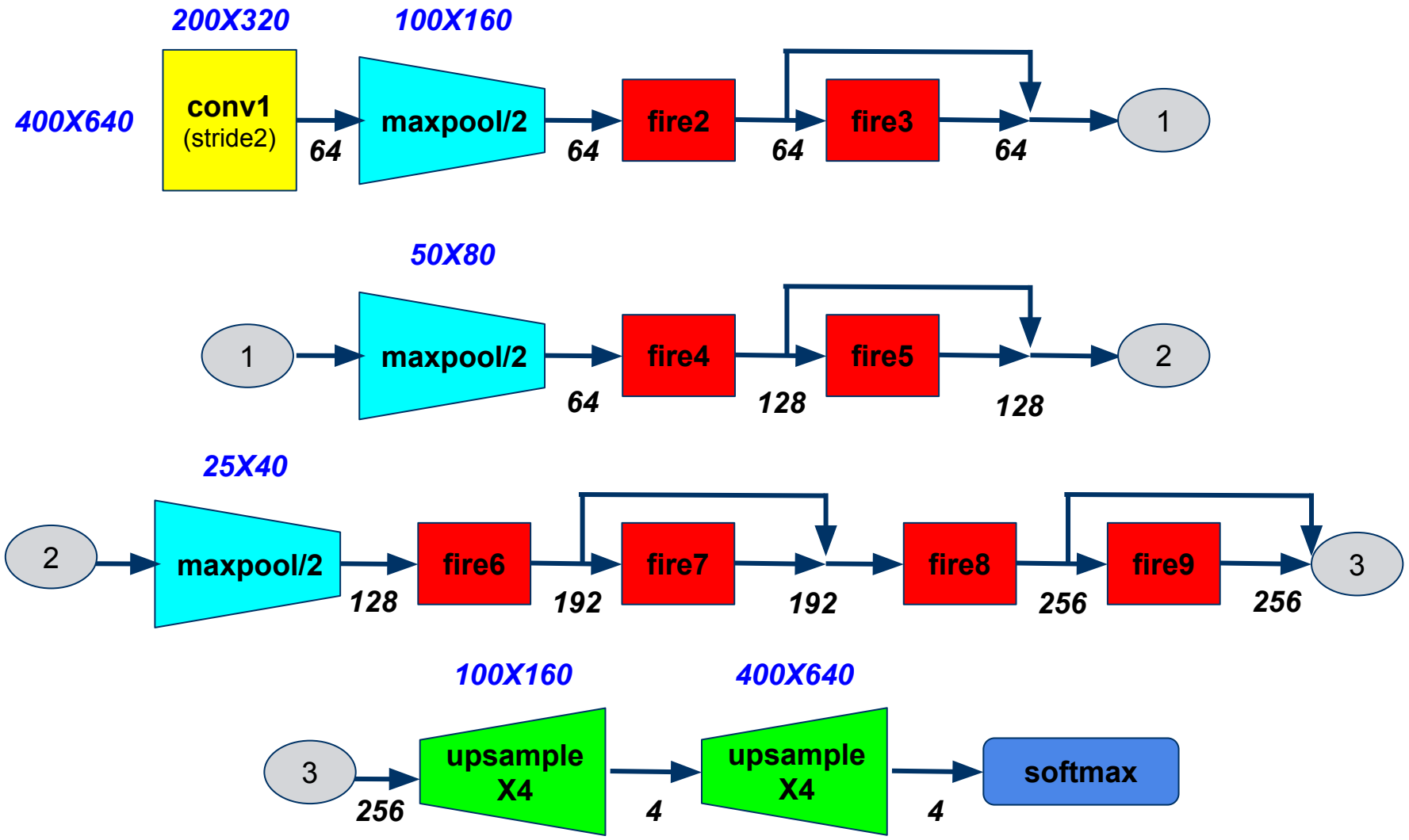
SqeezeNet: Design choices. Fire modules

Main ideas:

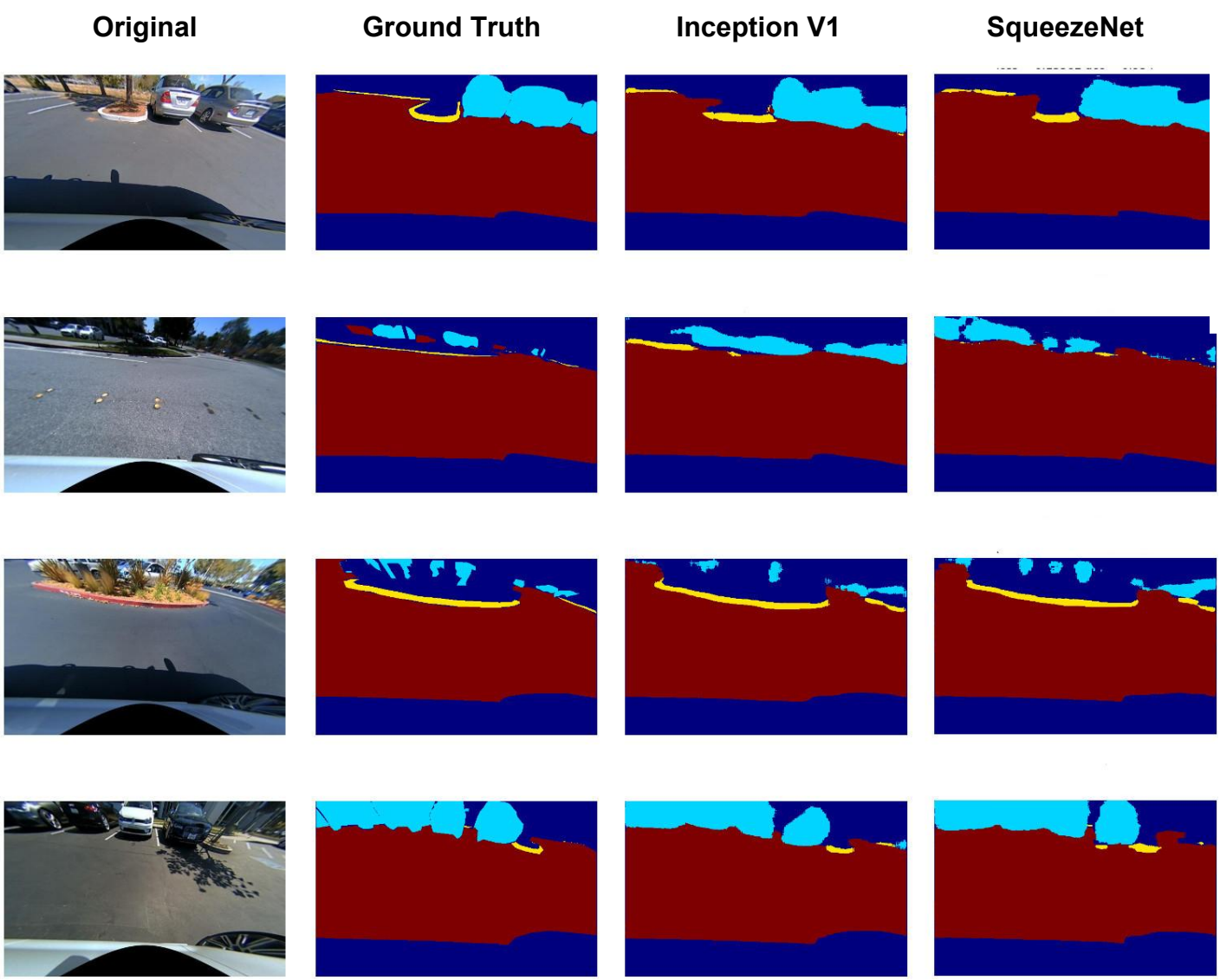
- Replace 3x3 filters with 1x1 filters
- Decrease number of input channels (**squeeze**)
- Postpone downsampling (convolve with stride 1)

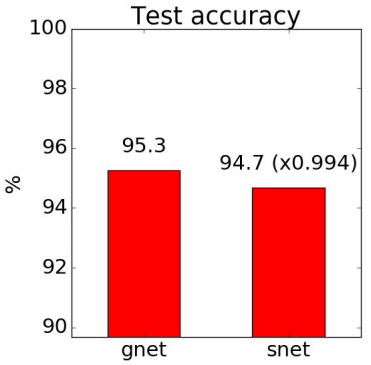


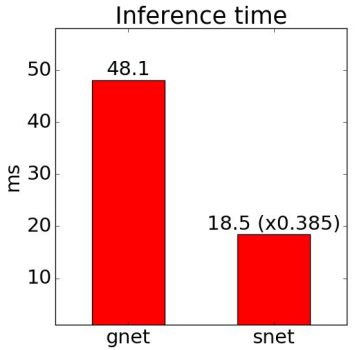
SqeezeNet: Architecture

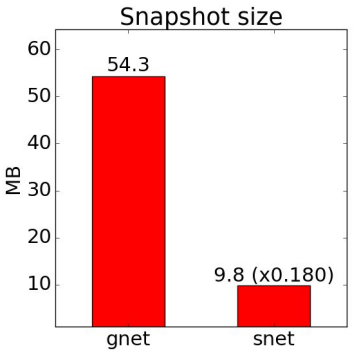


SqeezeNet: Training









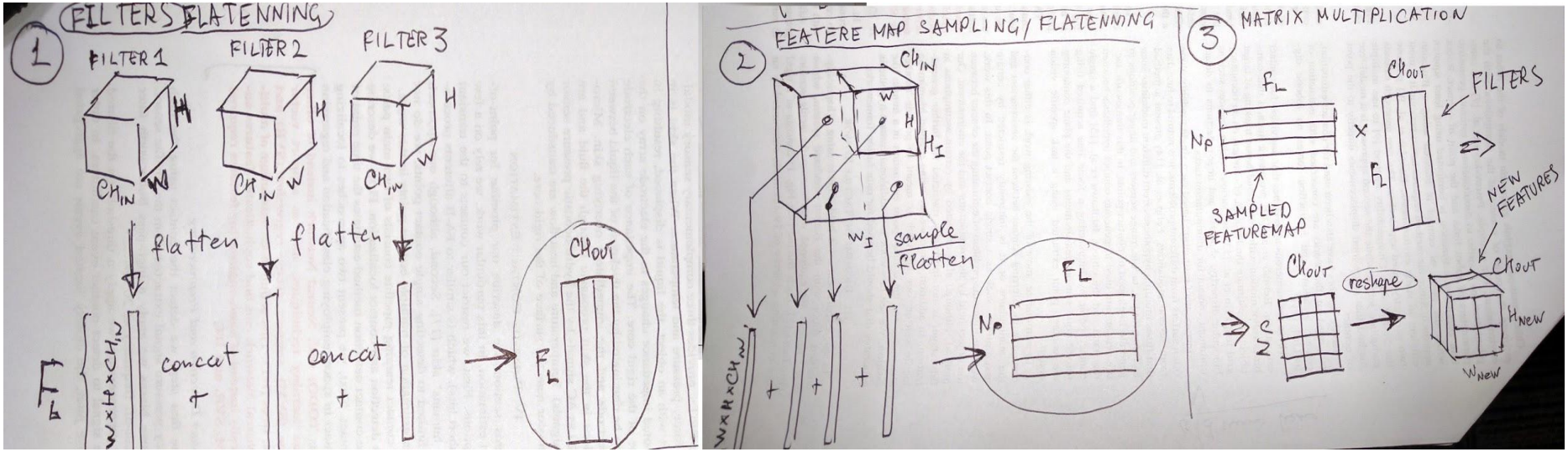
Convolution in a nutshell

What conv2d() do:

- **Flattens the filter to a 2-D matrix** with shape *[filter_height * filter_width * in_channels, output_channels].*

- **Extracts image patches** from the input tensor to form a virtual tensor of shape *[batch * out_height * out_width, filter_height * filter_width * in_channels].*

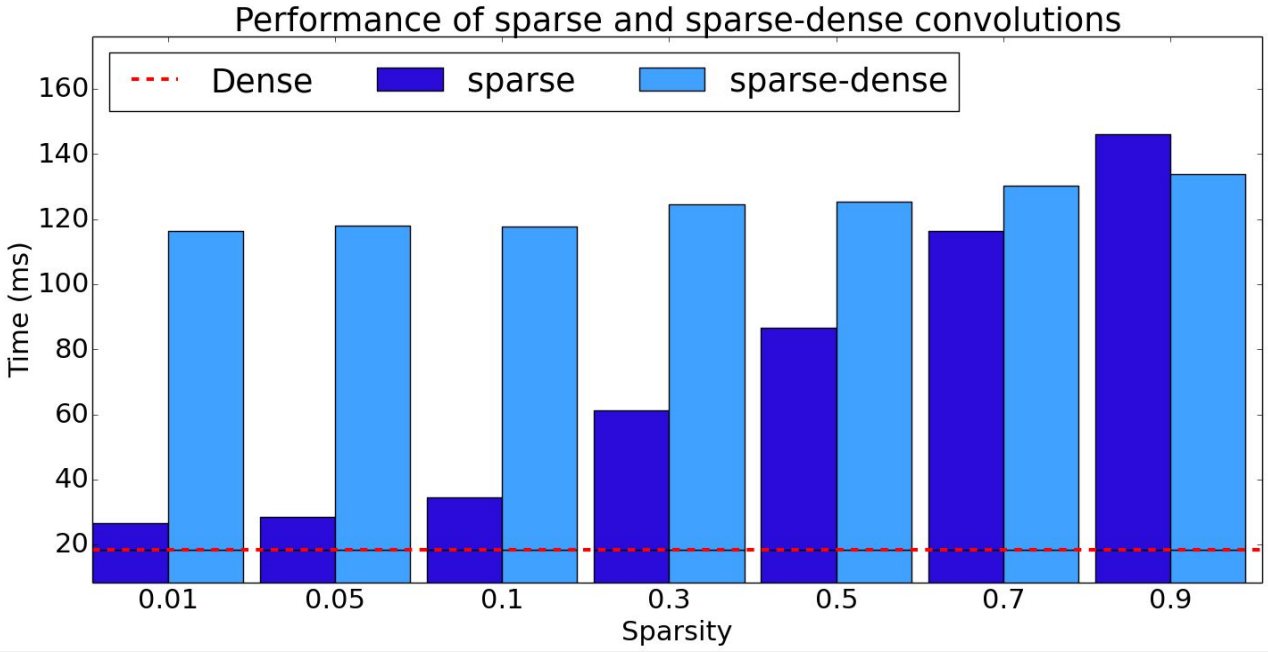
- For each patch, **right-multiplies the filter matrix** and the image patch vector.



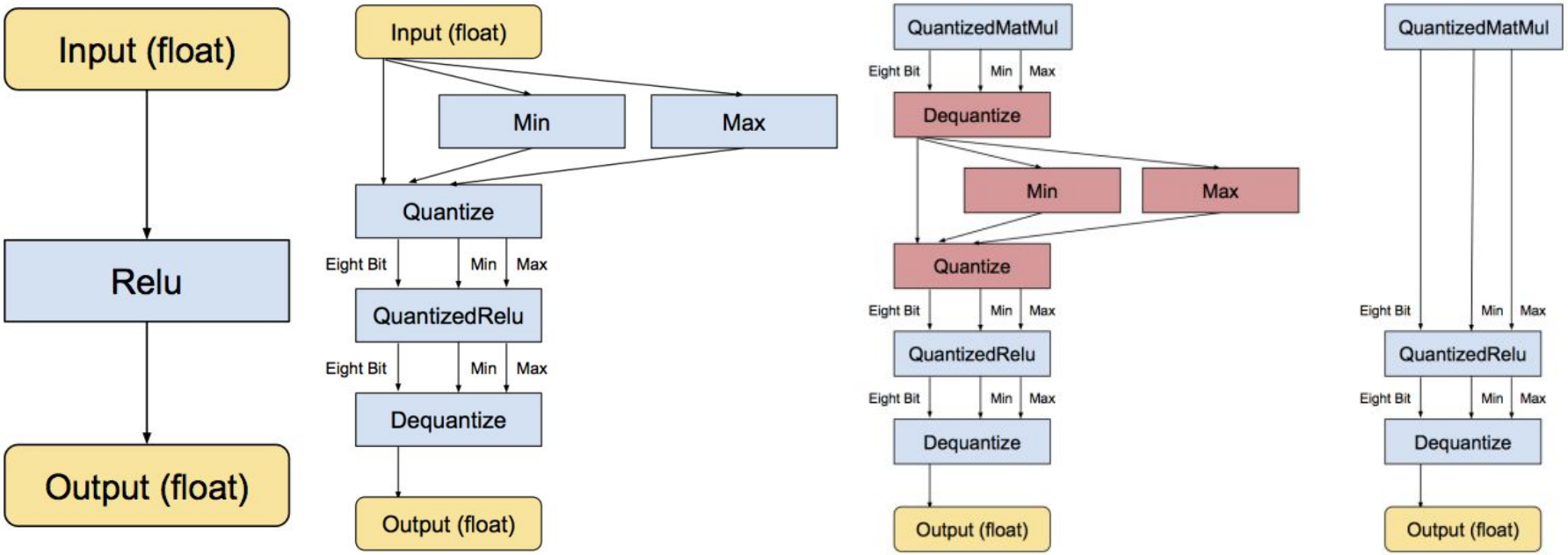
Convolutions: Sparse vs Sparse-Dense

Sparse vs. Sparse-Dense vs Dense (conv2d()):
differ by the TYPE OF MULTIPLICATION

- **Sparse:** SPARSE x DENSE
  tf.sparse_tensor_dense_matmul(filter, patches_mx, adjoint_b=True)
  Where ***filter*** is a **sparse matrix**
- **Sparse-Dense:** DENSE x DENSE with **SPARSE OPTIMIZATION:**
  *tf.matmul(patches_mx, filter, **b_is_sparse=True**)*
  Where ***filter*** is a **dense matrix**



Quantization: Overview



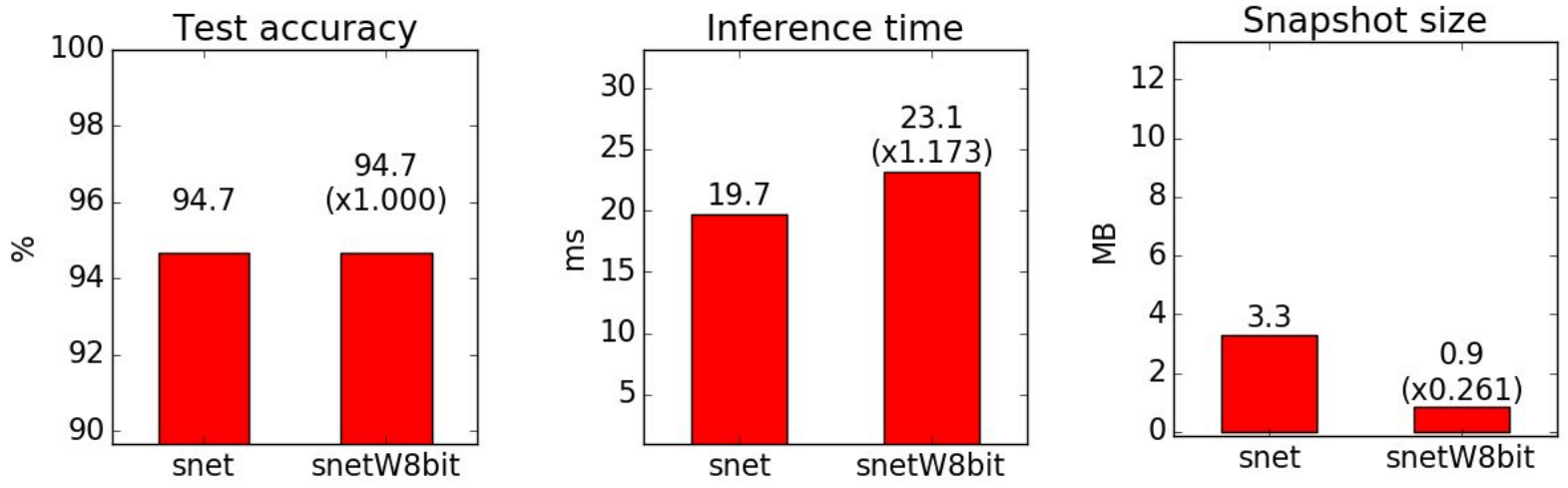
- Weights quantized uniformly between Min/Max values
- Min/Max values (range) are defined in an “optimal” way, rather than plain min/max floats to retain proper resolution

Source link:
https://petewarden.com/2016/05/03/how-to-quantize-neural-networks-with-tensorflow/comment-page-1/#comment-99004

Quantization: Tool and Results

- **Modes:**
  - **Weights** - round, compress (dequantize at runtime)
  - **Quantize -** quantize weights and the operations (described in the tutorial)
  - **Weights_rounded** - round weights to buckets, but don't compress
- **There are a few bugs:** only **8 bit weight** quantization worked
- Requires preparation **special *.pb** files with frozen weights from *.meta files and checkpoints

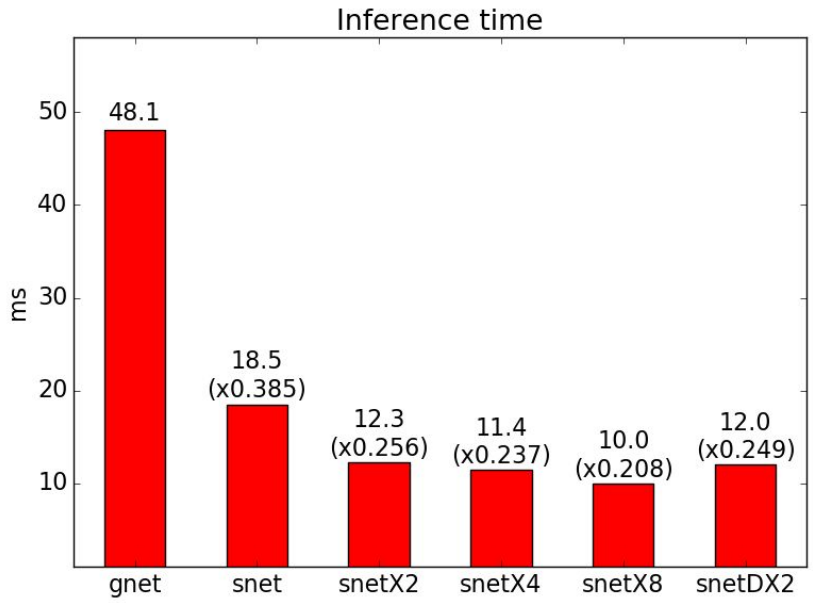
8bit weight quantization results

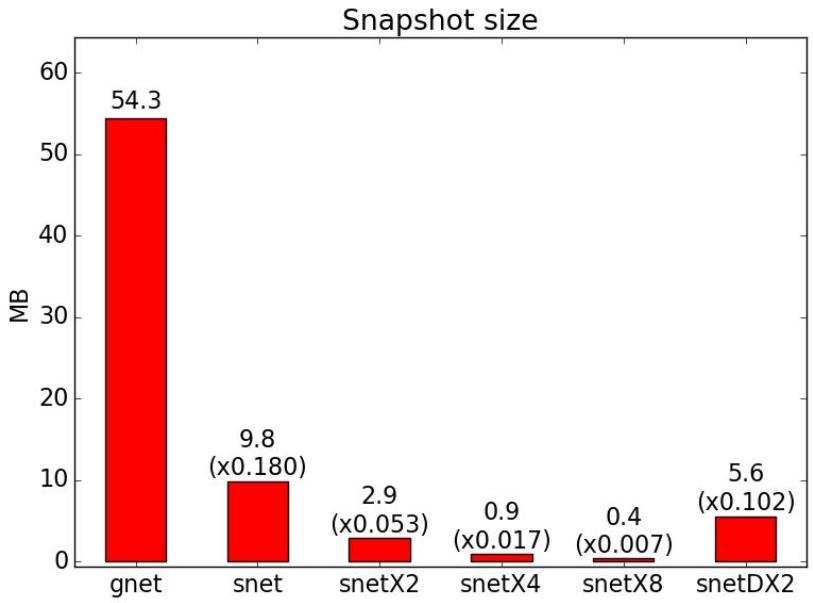


Smaller SqueezeNets: Architectures

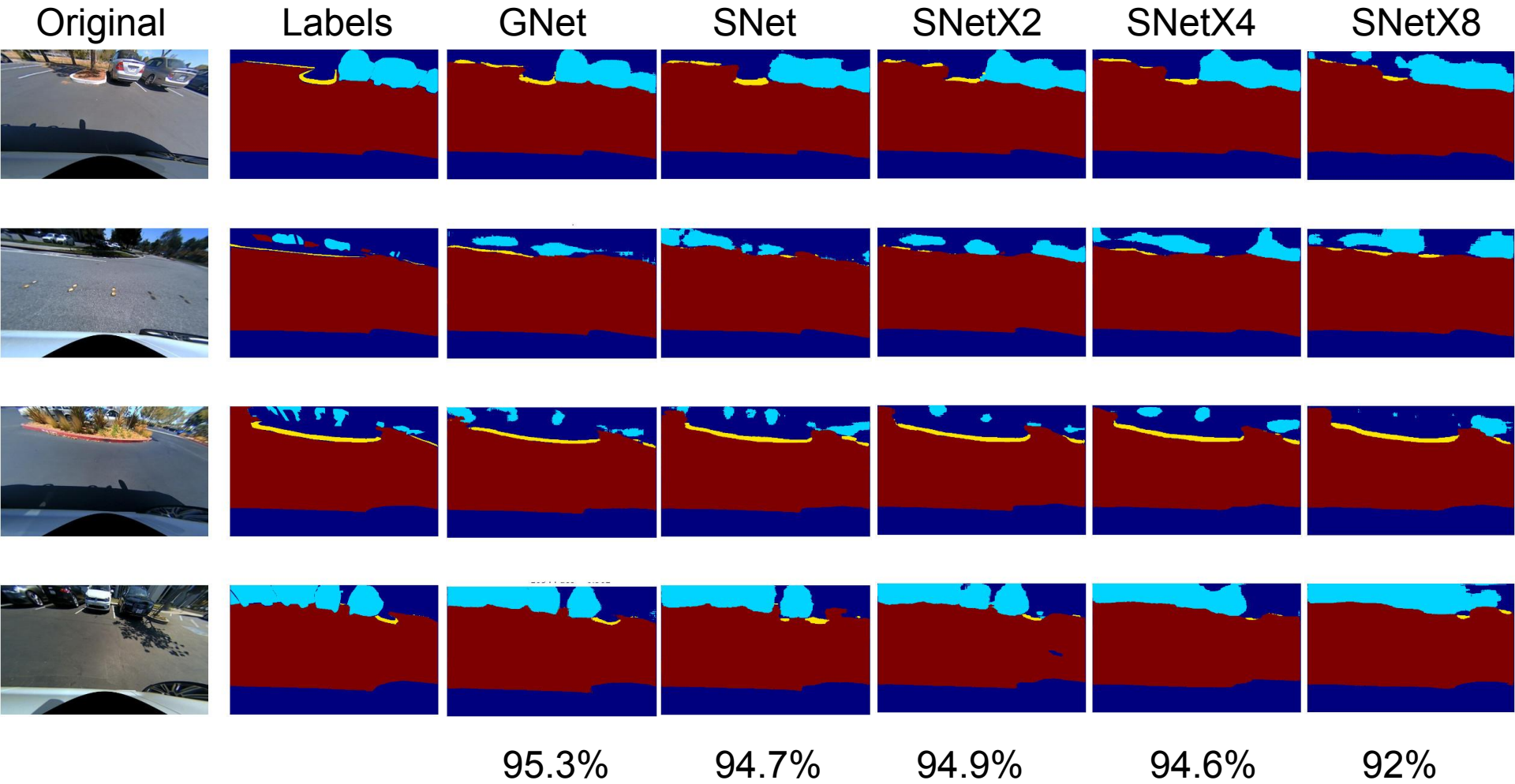
Architectures:

- **Width reduction** (reduce # of parameters in every layer):
  - snetX**N** - # of parameters reduced **~N** times
    - snetX2, snetX4, snetX8
- **Height reduction** (reduce # of parameters by stripping layers):
  - snetDX**N** - # of layers reduced **~N** times





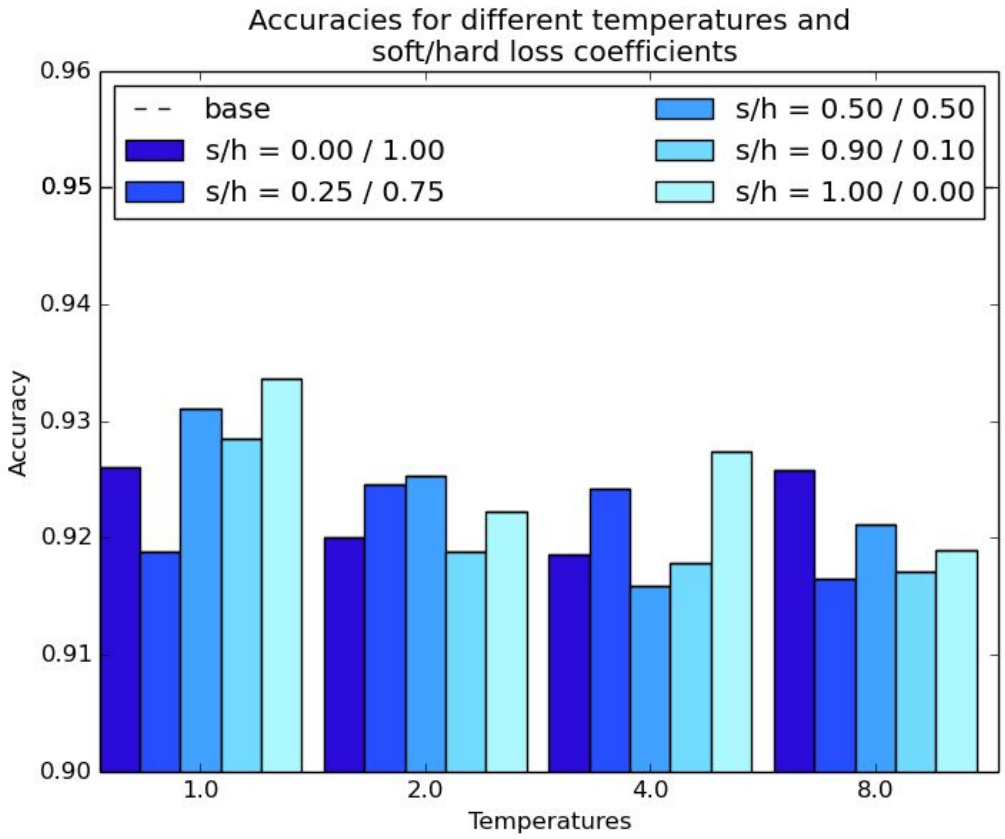
Smaller SqueezeNets: Training



Distillation: snet to snetX8

Legend:

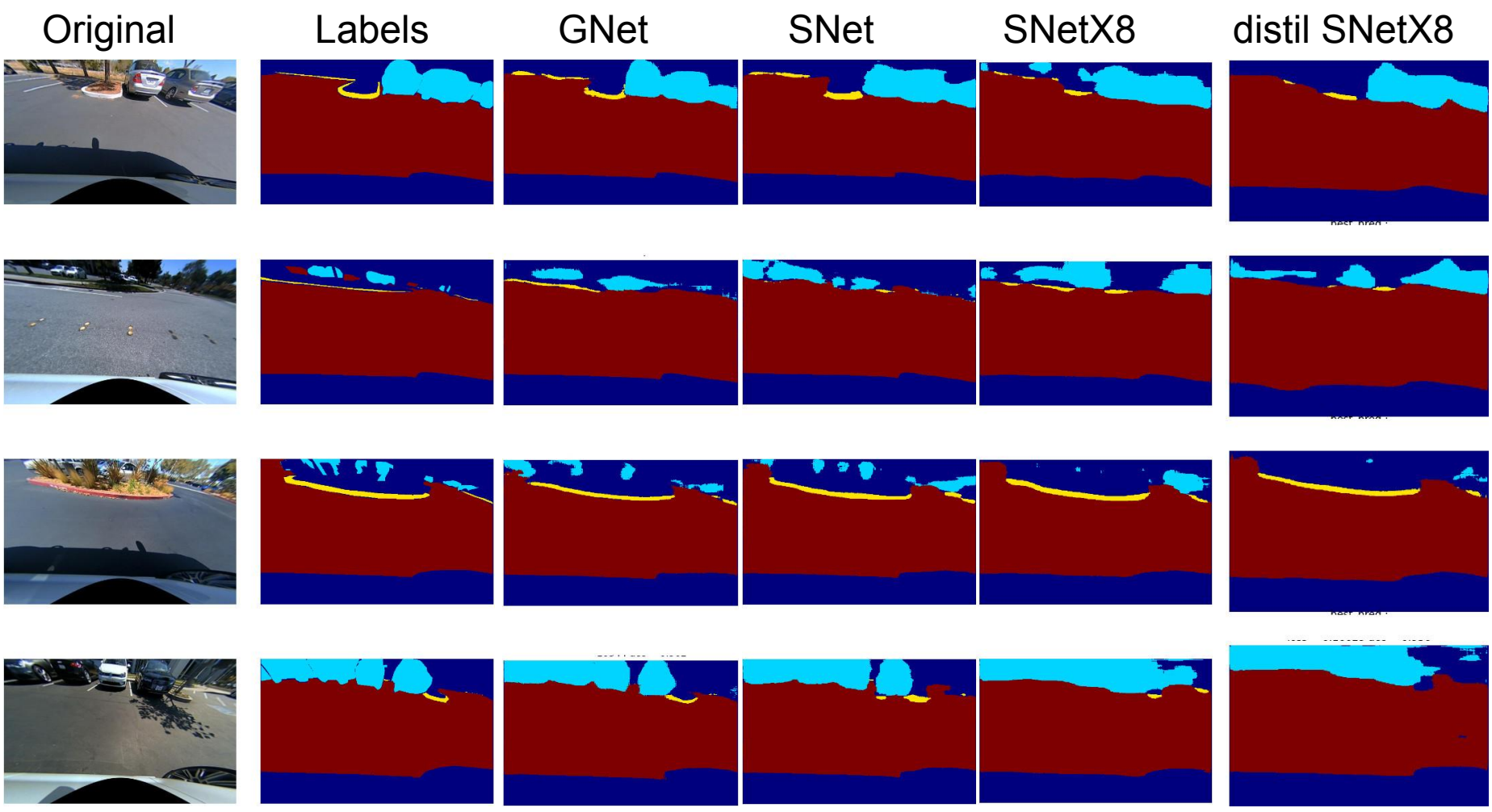
- **s/h =** weights of losses from soft/hard labels in the final weighted loss, i.e. Loss = s * Loss_softlabel + h * Loss_hardlabels



Conclusion:

Distillation may help, but there is no clear rule to pick up parameters

Distillation: snet to snetX8



Next steps:

- **Quantization:**
  - Does full 8-bit quantization give speed boost ?
- **Distillation:**
  - Incorporate intermediate feature maps into distillation
- **Sparse multiplication:**
  - Can we do better ?
- **Architectural changes to SqueezeNet:**
  - More graduate upsampling (4 stages instead of 2) +
  - Get features from earlier feature maps
- **Removing\Resetting correlated filters**

Discussion:

QUESTIONS ???

HDF5: file structure

- **Features and labels:**
  - /feat/.. - group containing different features
  - /label/.. - group containing different labels
- **Cross-validation indices** (for consistent comparison)
  - /crossval_indx/[index]/train
  - /crossval_indx/[index]/val
  - /crossval_indx/[index]/test
- **Names for cross-validations**
  (in case they have semantic meaning, like testing on different unseen objects)
  - /crossval_names