Bootstrapping Deep Neural Networks from Approximate Image Processing Pipelines

Kilho Son,
Jesse Hostetler,
Sek Chai

February 17, 2019. EMC2 Workshop

SRI International®

SRI's Mission: World-changing solutions to make people safer, healthier, and more productive.


First telerobotic surgical system

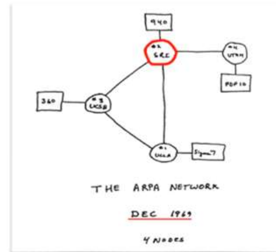

New drug for lymphoma

.com
.gov
.org

First assigned domain names


First computer mouse


Ultrasound for medical diagnostics



First ARPANET and internetworking nodes


U.S. Dept. of Education 2010 technology plan

Treatment for drug-resistant malaria


Emmy Awards for HDTV and more

Created Siri (acquired by Apple)

SRI International

The team

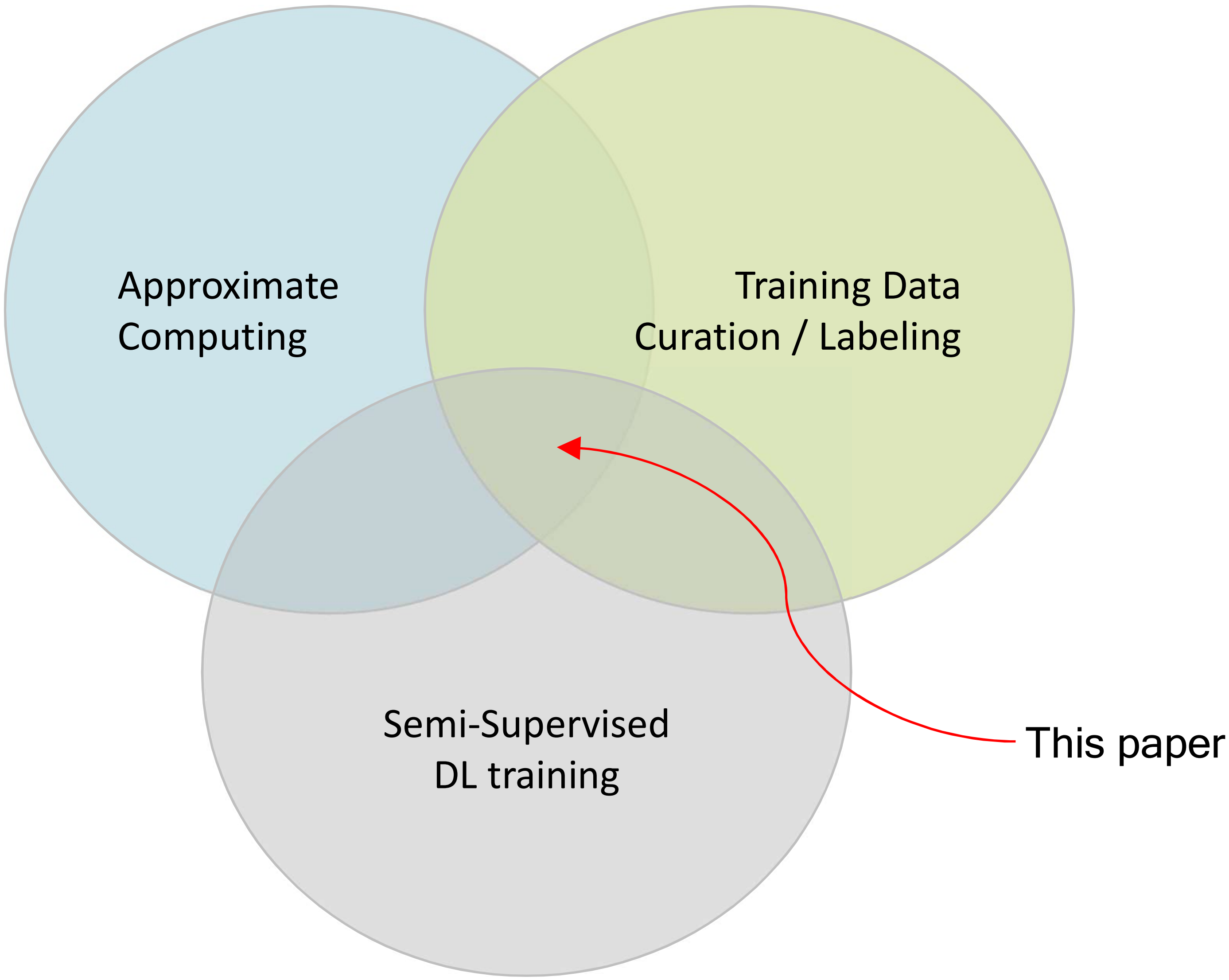
Kilho Son

Jesse Hostetler

Acknowledgments

- Special thanks to our research sponsor.

Background

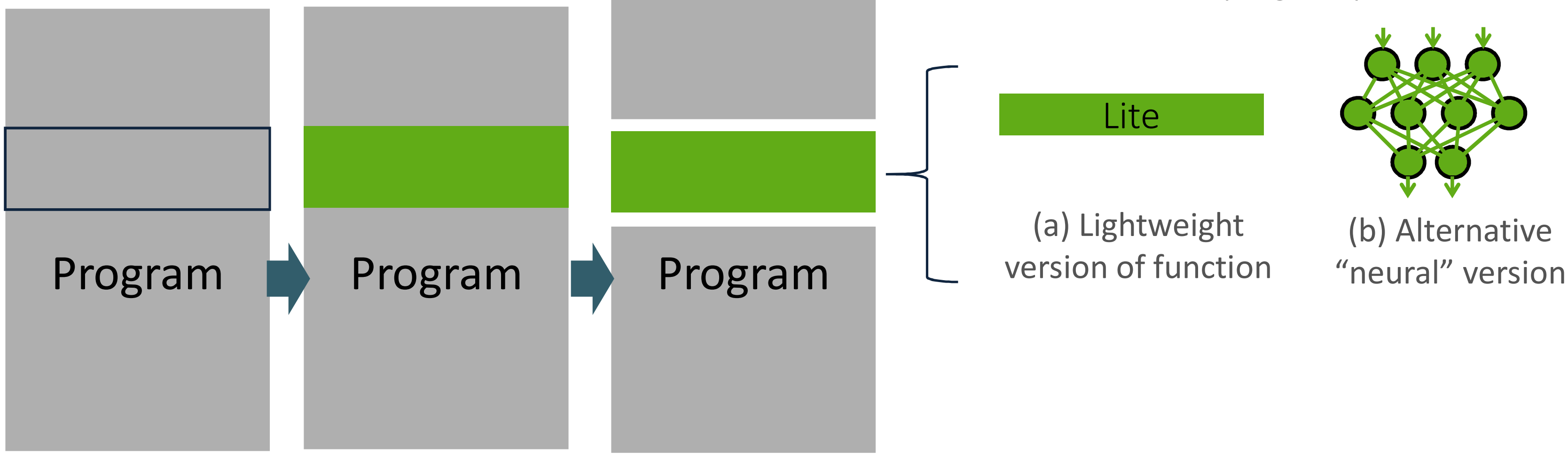


SRI International

Motivation (Computation Efficiency)

Approximate Computing

We use alternative version of functions (a or b) without loss of program performance.

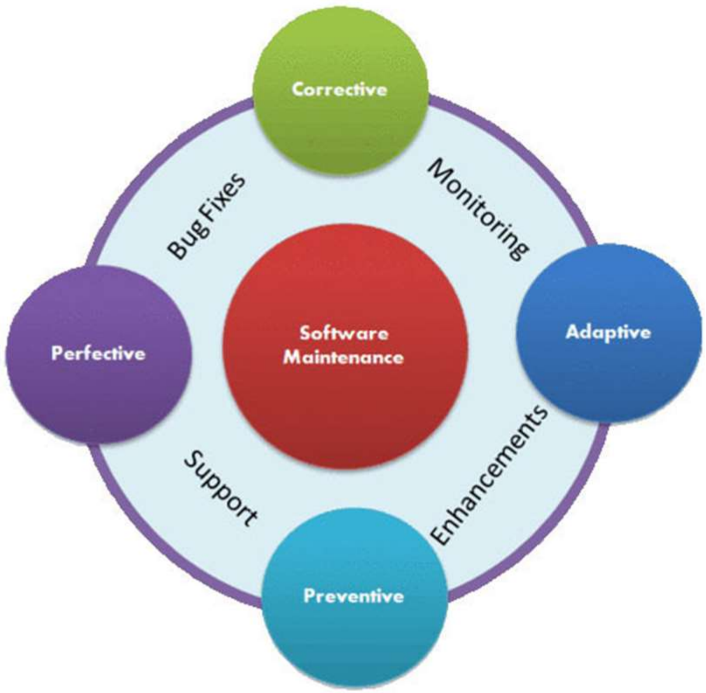


Basic Premise

- Can we leverage new deep learning accelerators by structurally approximate functional modules in software programs?

Motivation (Software Maintenance)

Software Lifecycle*

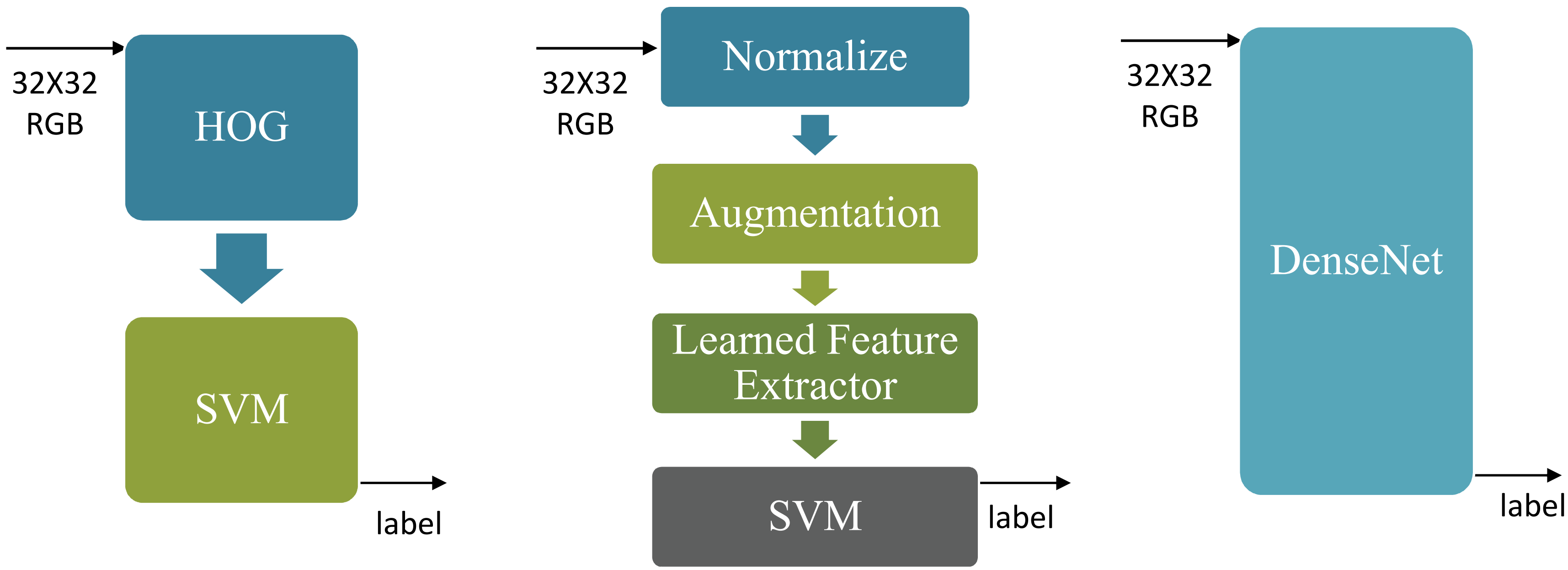


- Software modules are developed through a combination of engineering and expertise (e.g. parameter tuning, compute optimization, etc.).
- Revisiting the pipeline to make changes or improvements requires thorough understanding by domain experts or developers.
- Systematic upkeep of code can be costly, especially if the system is old and engineering knowledge has been lost.

* Source: https://www.indiamart.com/adhrittechnologies/software-maintenance-services.html

Example Processing Pipeline

Training data

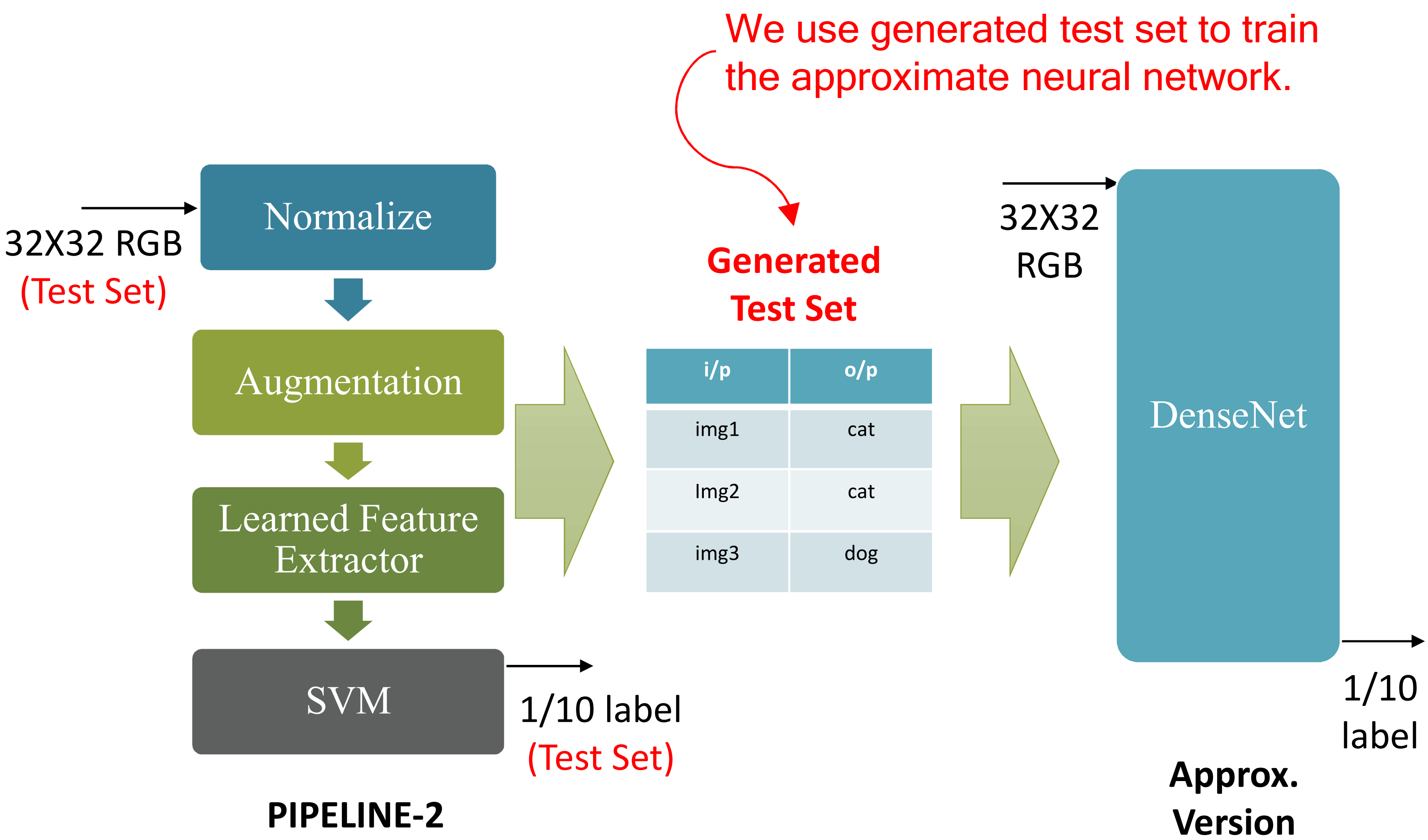


PIPELINE-1
Achieves ~50% accuracy

PIPELINE-2
~80% accuracy

PIPELINE-3
~95% accuracy

Generating Training Data for Approximation



DNN Computation Examples

Image Classification

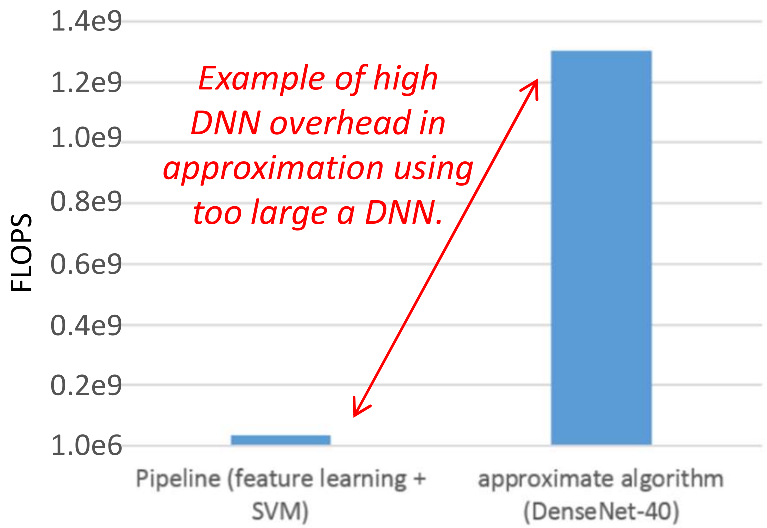
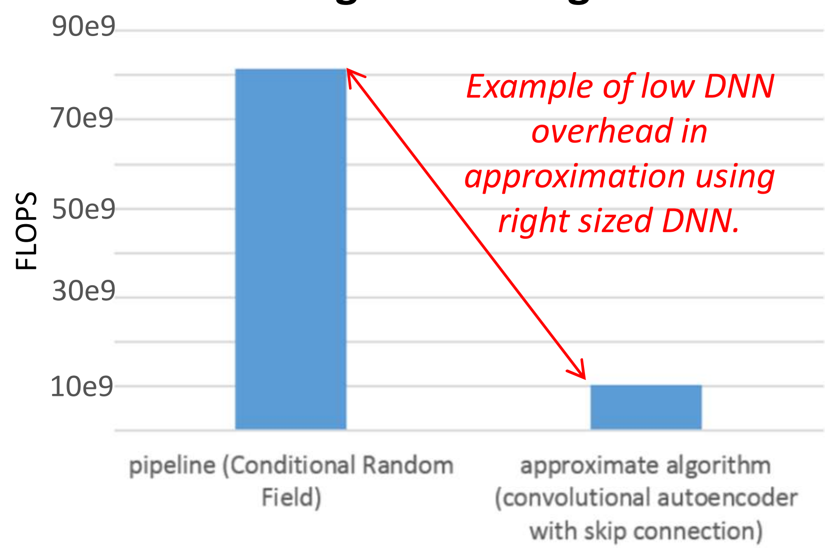
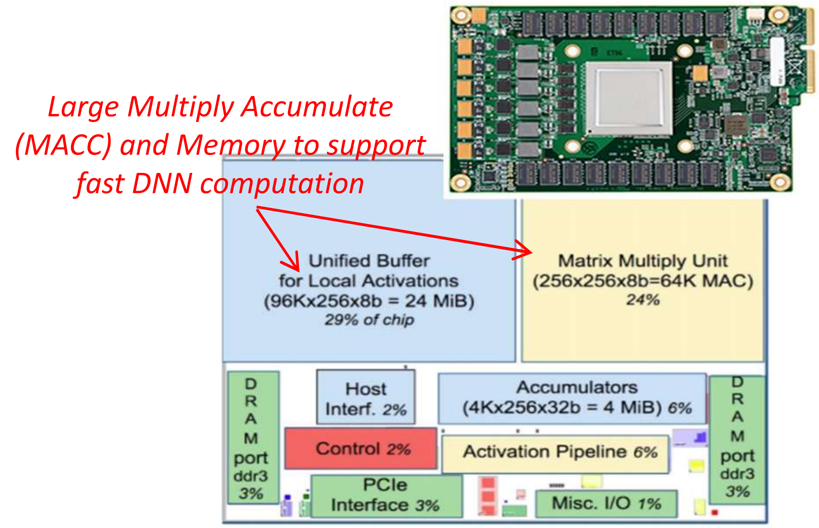


Image Denoising



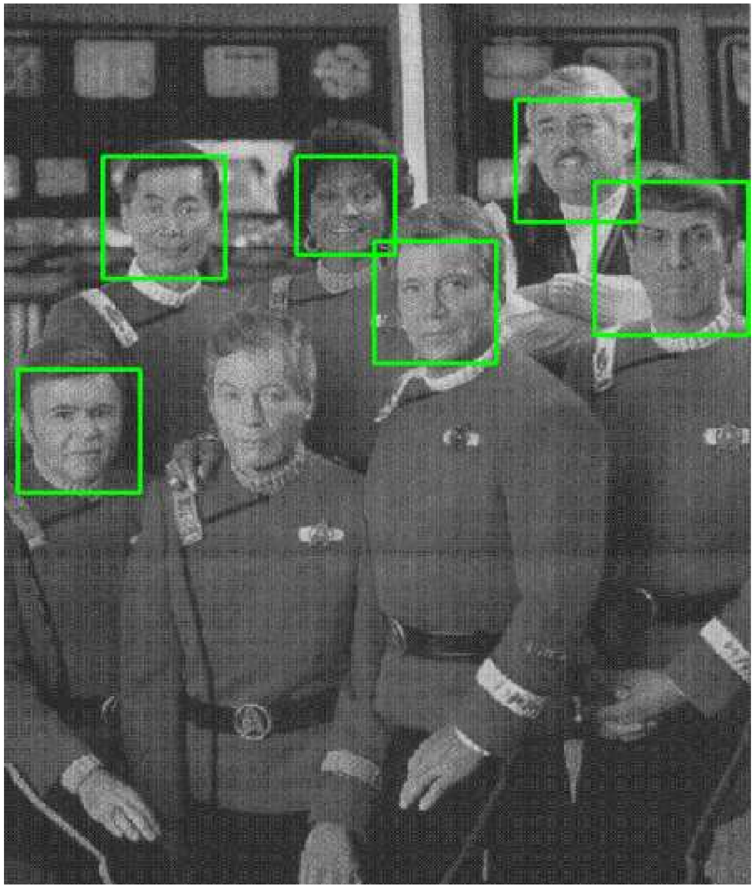


Google TPU for DNN
40Watt (28nm 700MHz)

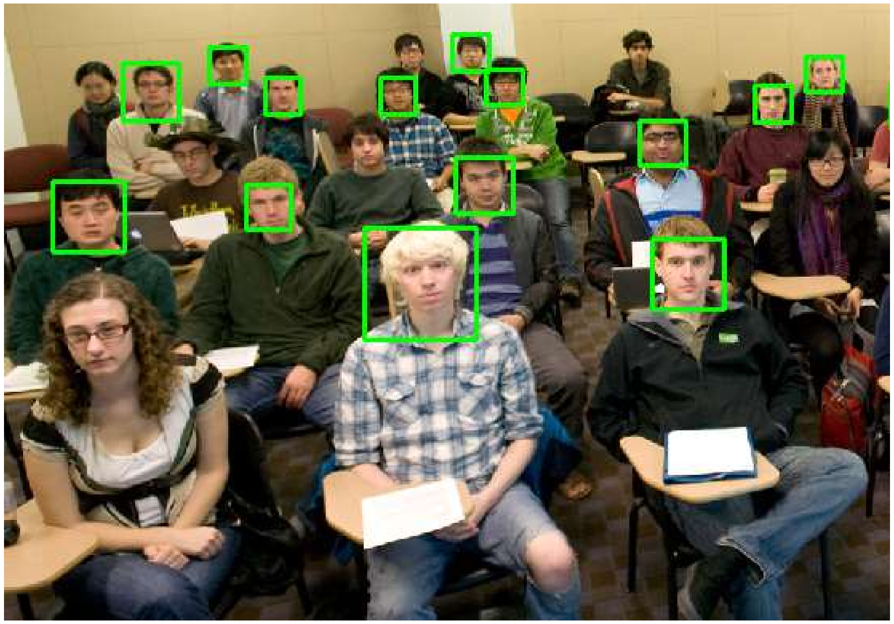
Insights:

1. We can train Approximate DNNs, but we must also select an appropriately sized DNN.
2. Approximate DNNs make processor operation more uniform (e.g. mostly MACCs) and thus hardware can be more efficient.
(reference: 80x vs CPU, 30x vs GPU)

Noisy Labels Examples



http://cs.brown.edu/courses/cs143/2011/results/proj4/hangsu/



http://cs.brown.edu/courses/cs143/2011/results/proj4/hangsu/

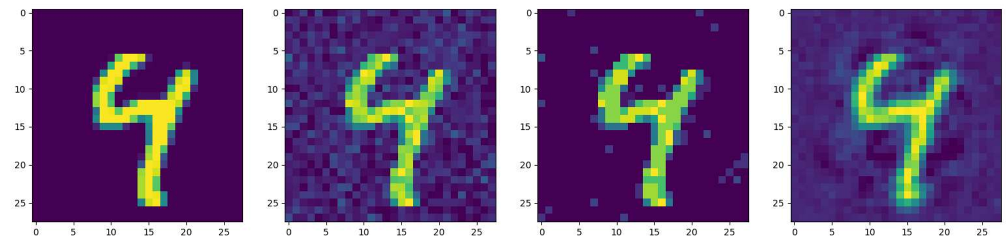
Generated labels can be noisy due to:

1. Low accuracy of conventional pipeline.
2. Operational settings that alter the performance of the function.
3. Coding errors.

Example Evaluation Benchmarks

Image Denoising

Image Denoising Average SSIM	Evaluation	
	Train	Test
Pipeline (MNIST)	78.8%	78.9%
Approximate	78.0%	78.2%



Groundtruth ssim

Gaussian noise added

Conventional CRF

Auto-encoder with CRF data

Image Denoising Average SSIM	Evaluation	
	Train	Test
Pipeline BSD300	53.9%	56.3%
Approximate	56.4%	59.4%

We are able to train an Approximate DNN that performs equally well on ground truth dataset vs. user-coded pipeline.

SRI International

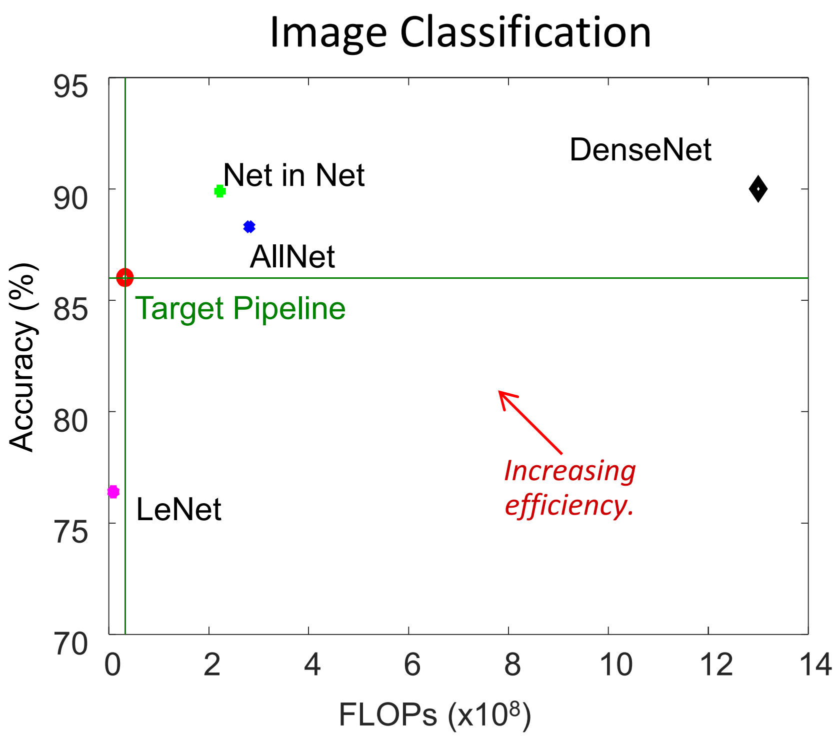
Example Evaluation Benchmarks

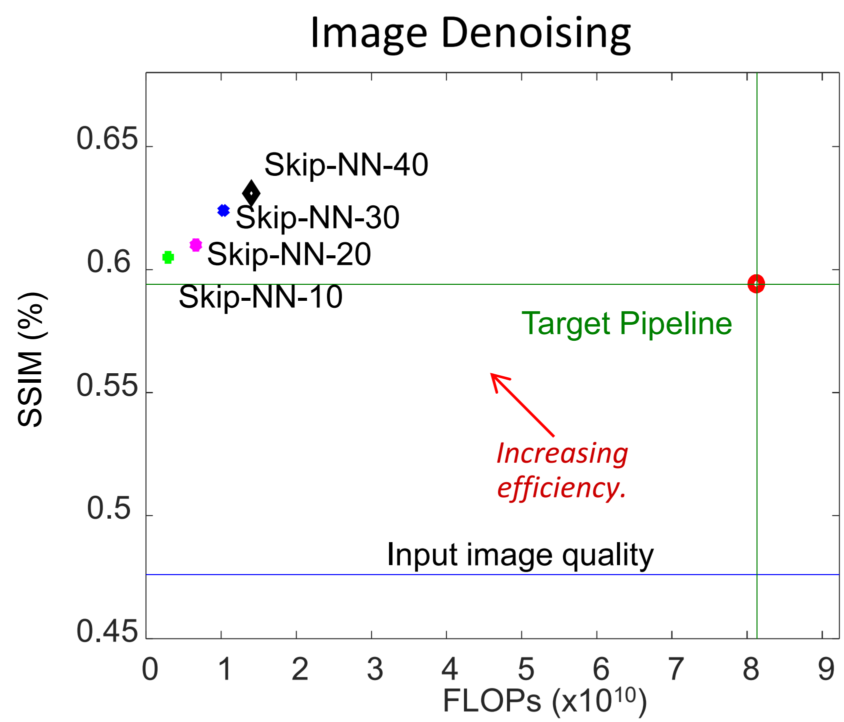
Image Classification

Image Classification, Average Detection	Evaluation	
	Train	Test
Pipeline CIFAR10	95%	86%
Approximate	95%	90%

Approximate DNN can generalize and outperform user-coded pipeline.

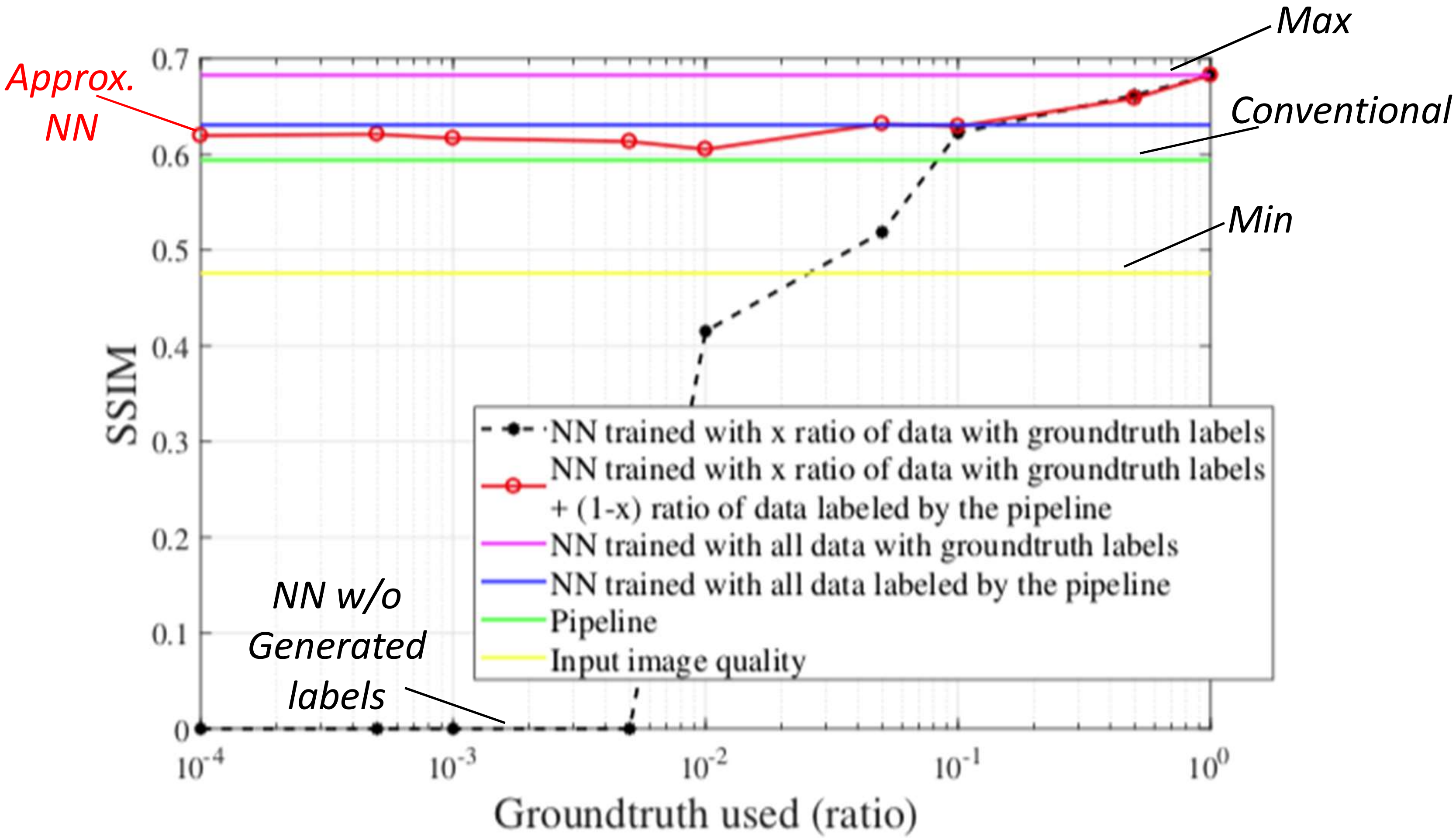
Performance Characterization





Approximation results on two benchmarks, showing achievable performance vs. computational complexity. We can select the appropriate DNN based on target power efficiency.

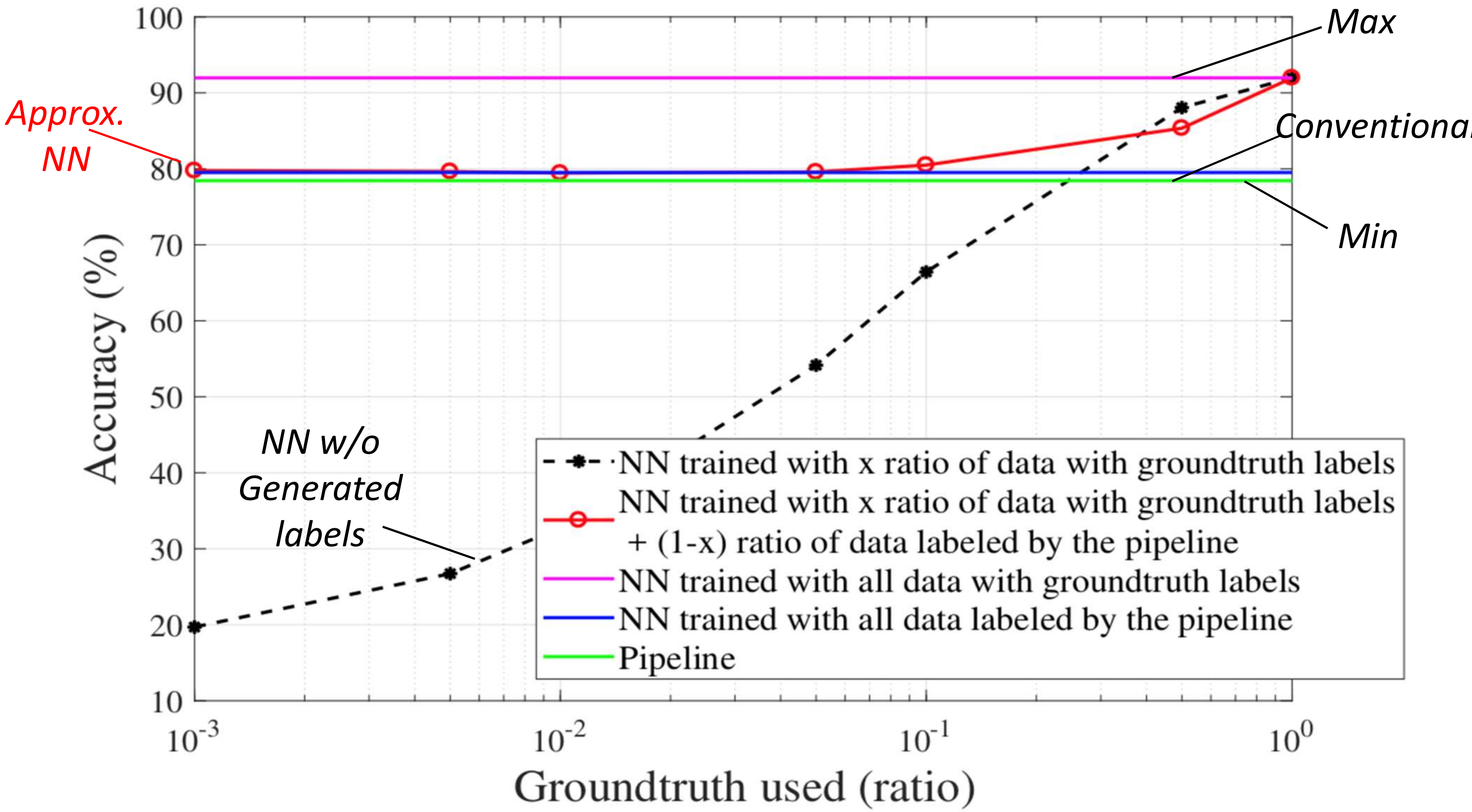
Training Data Characterization – Image Denoising



Takeaway –

- We can train NN with significantly less manual labeling of data -- e.g. use the conventional pipeline (green) to generate labels to train Approx NN (red).

Training Data Characterization – Image Classification



Summary

- Our result addresses ML with less manually curated labels.
- Helps future development of complex image processing and computer vision systems.
- Older deep networks (LeNet / AlexNet of yesteryear) are still useful to generate labels.

Looking ahead

- Generalization to other application domains
- Improve training convergence by characterizing the noise as outliers (removed from training set)
- Explore other uses, e.g. dynamic runtime selection based on performance/accuracy needs.