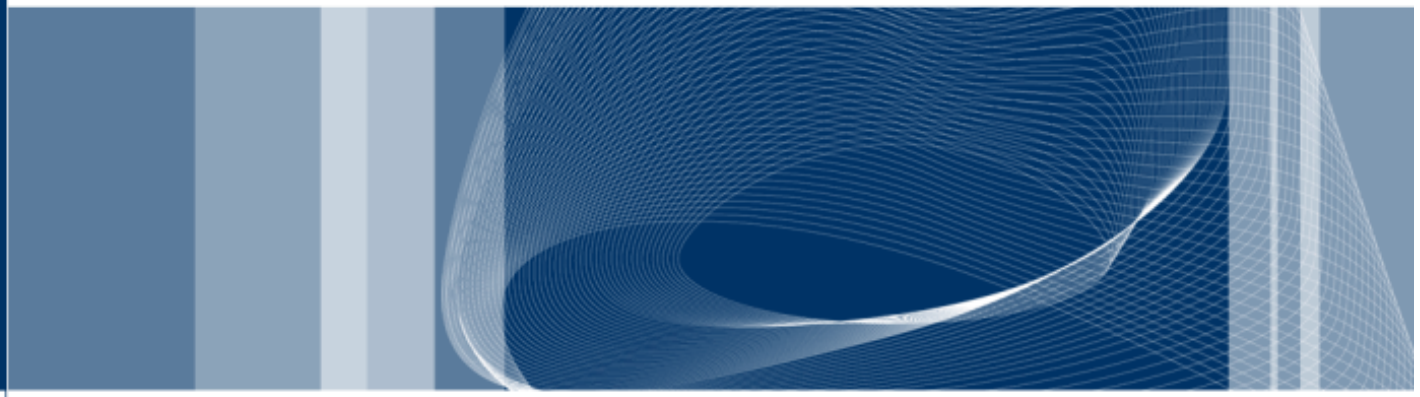
POLITECNICO DI MILANO



PERFORMANCE PREDICTION OF GPU-BASED DEEP LEARNING APPLICATIONS

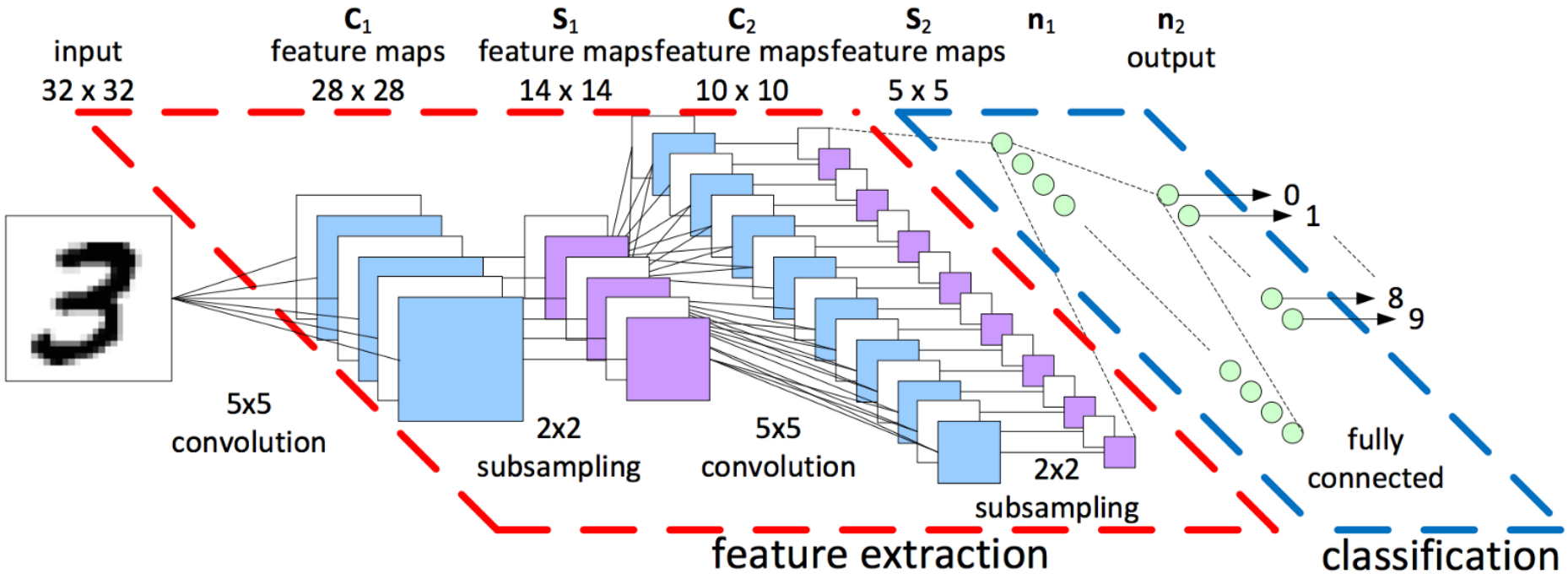
Eugenio Gianniti[1], Li Zhang[2], and **Danilo Ardagna**[1]

[1]Politecnico di Milano, [2]IBM Research

Introduction & Motivation

- Deep learning is widely applied across industries
- Model learning greatly benefits from GPUs
- GPU as a service: $200-million market in 2016 with a 30% CAGR
- Goal: performance models with good generality

Convolutional Neural Networks



CNNs are characterized by their architecture and hyper-parameters

Contributions

- Per Layer Models
- End to End Models

Layer Computational Complexity

Layer	Forward	Backward
Conv	$H_{\mathrm{f}} W_{\mathrm{f}} C_{\mathrm{in}} C_{\mathrm{out}}$	$\left(2 H_{\mathrm{f}} W_{\mathrm{f}} C_{\mathrm{in}} + 1\right) C_{\mathrm{out}}$
FC	$H_{\mathrm{in}} W_{\mathrm{in}} C_{\mathrm{in}} C_{\mathrm{out}}$	$2 H_{\mathrm{in}} W_{\mathrm{in}} C_{\mathrm{in}} C_{\mathrm{out}}$
Loss	$4 C_{\mathrm{out}} - 1$	$C_{\mathrm{out}} + 1$
Norm	$5 C_{\mathrm{out}} + C_{\mathrm{n}} - 2$	$8 C_{\mathrm{out}} + C_{\mathrm{n}} - 1$
Pool	$H_{\mathrm{f}} W_{\mathrm{f}} C_{\mathrm{out}}$	$\left(H_{\mathrm{f}} W_{\mathrm{f}} + 1\right) C_{\mathrm{out}}$
ReLU	$3 C_{\mathrm{out}}$	$4 C_{\mathrm{out}}$

- Derived from architecture and hyper-parameters

Per Layer Models

$$t_l = \beta_{0l} + \beta_{1l} c_l + \varepsilon_l$$

- Learn linear regression models per layer

$$\hat{t}^{\mathrm{CNN}} = I \sum_{l \in L} \hat{t}_l$$

- Sum terms according to CNN architecture

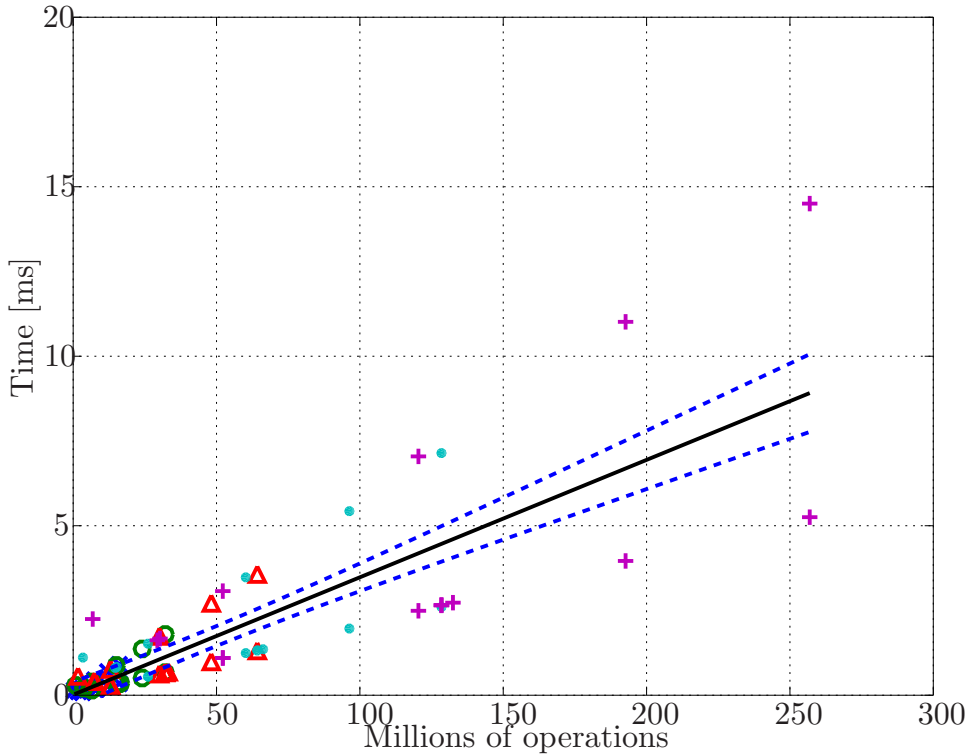
End to End Models

- Extract from historical data the execution time of the network in its entirety
- Vary batch sizes and number of iterations
- Apply linear regression
- Affine dependency

Experimental Settings

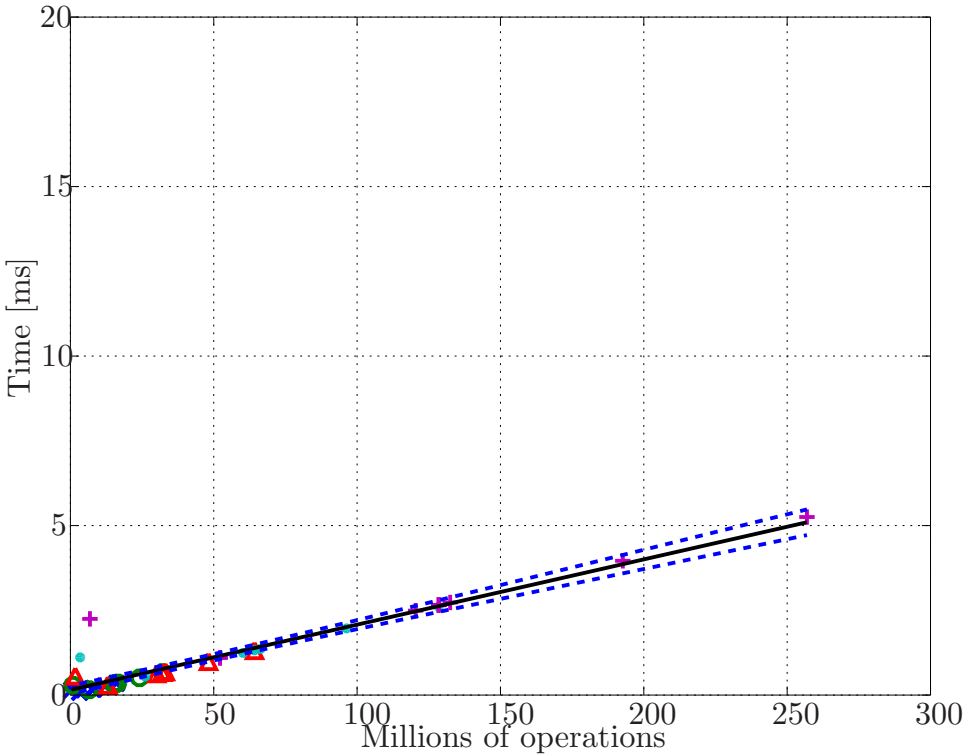
- Several well-known convolutional neural networks
  - AlexNet
  - GoogLeNet
  - VGG-16
- ImageNet dataset (ILSVRC12)
- Intel Xeon 10-core processor and NVIDIA Quadro M6000
  - 3,072 CUDA cores
  - 12 GB dedicated RAM, 317 GB/s bandwidth
  - 7.0 TFLOPS in single precision

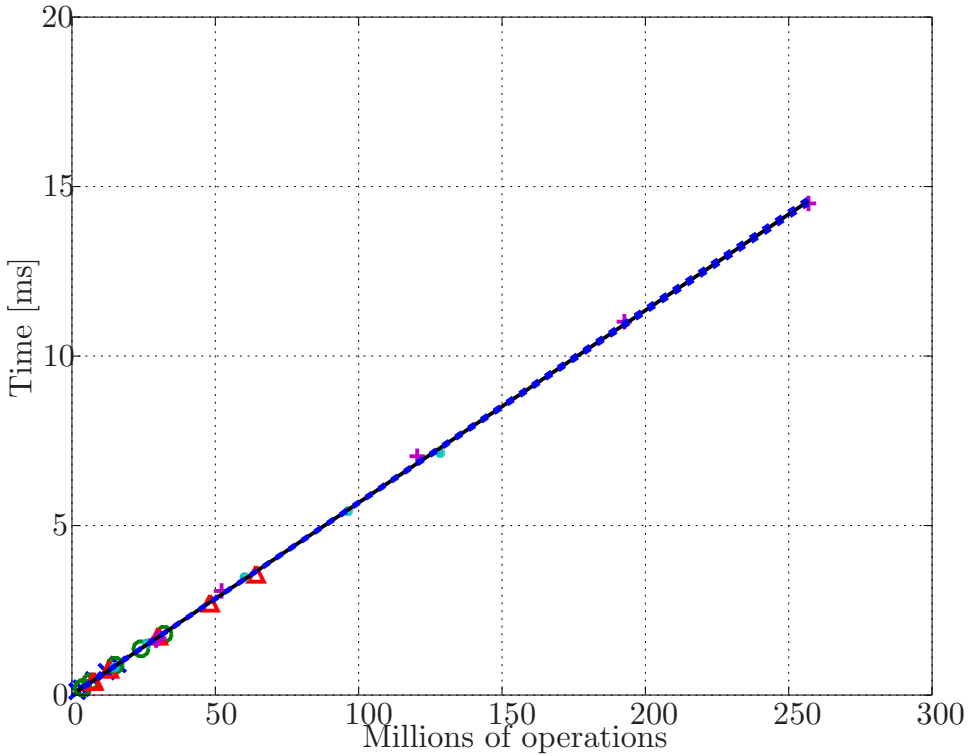
GoogLeNet Pooling Layers



Large errors with only one model for pooling layers

GoogLeNet Pooling Layers





$S = 1$

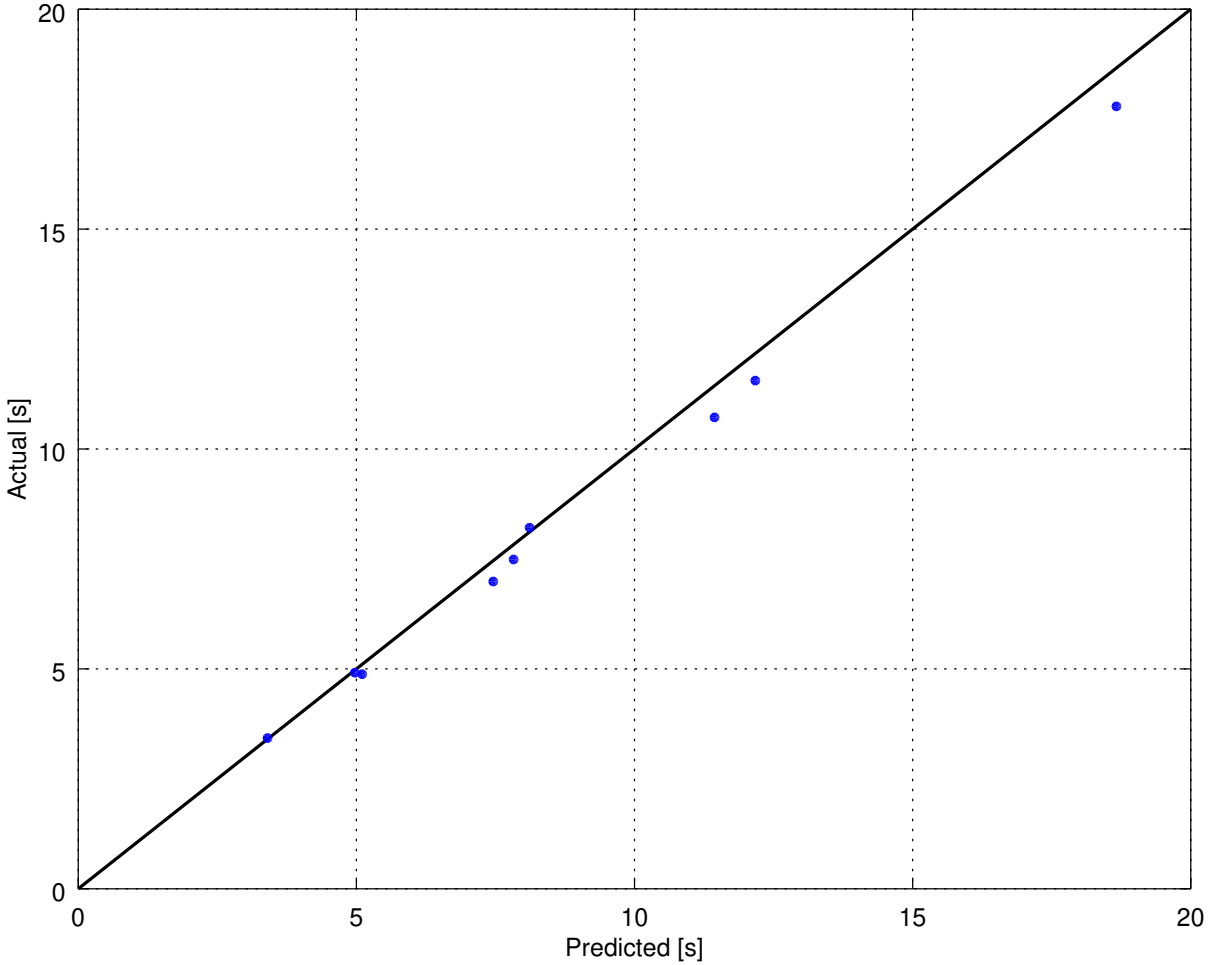
$S > 1$

Linear Regression Coefficients

Category	$\hat{\beta}_0^{\text{fw}}$ [ms]	$\hat{\beta}_1^{\text{fw}}$ $\left[\frac{\text{ms}}{\text{op}}\right]$	$\hat{\beta}_0^{\text{bw}}$ [ms]	$\hat{\beta}_1^{\text{bw}}$ $\left[\frac{\text{ms}}{\text{op}}\right]$
Conv/FC	1.83E−1	3.43E−10	3.15E−1	3.65E−10
Norm	1.64E−2	7.11E−9	1.01E−1	6.87E−9
Pool $S = 1$	2.23E−2	1.27E−8	1.52E−1	1.93E−8
Pool $S > 1$	1.44E−2	1.45E−8	6.11E−3	5.67E−8
ReLU/Drop	8.91E−3	1.17E−8	1.18E−2	1.33E−8

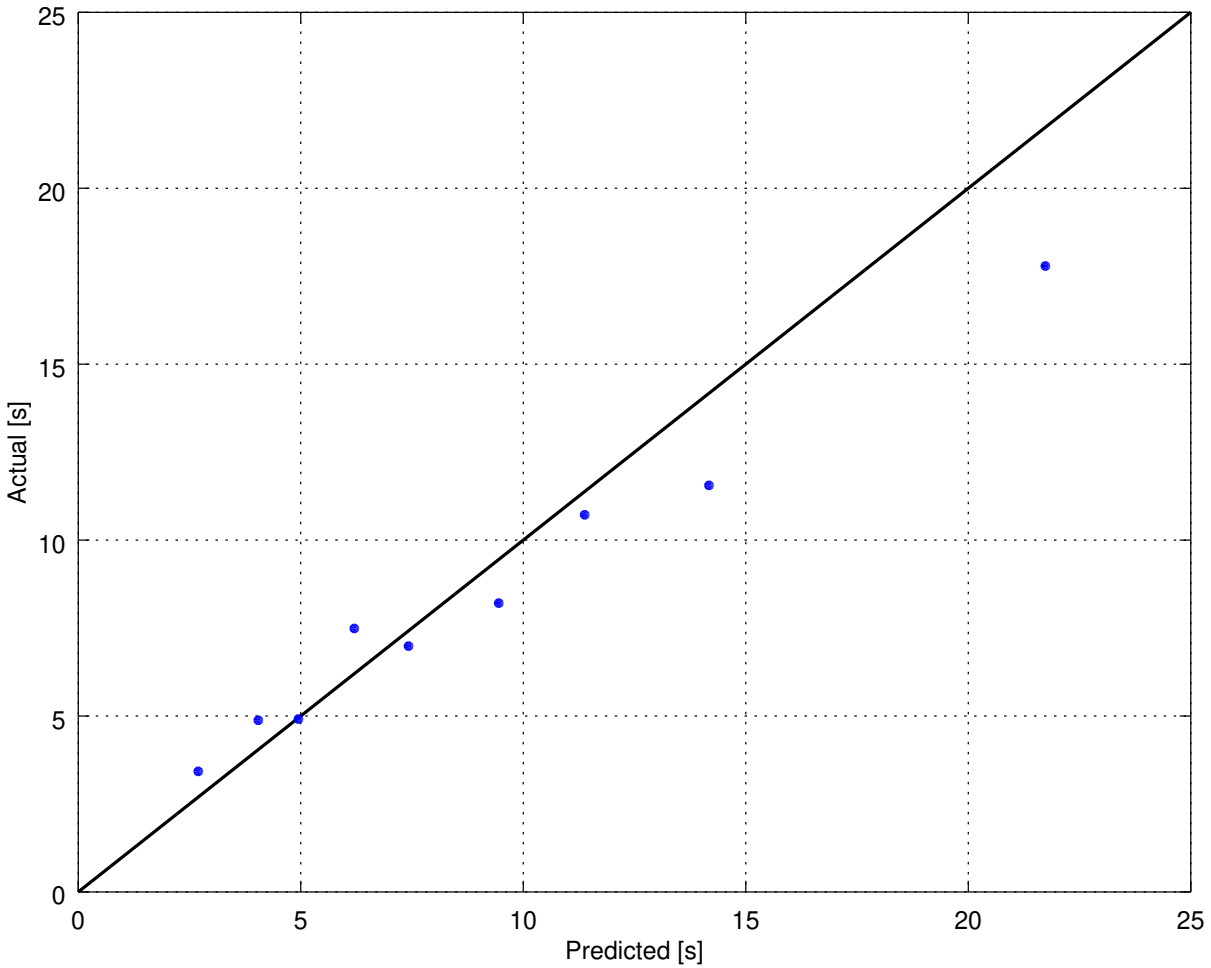
- Models obtained from GoogLeNet timing data

Actual vs Fitted Plot – Per Layer



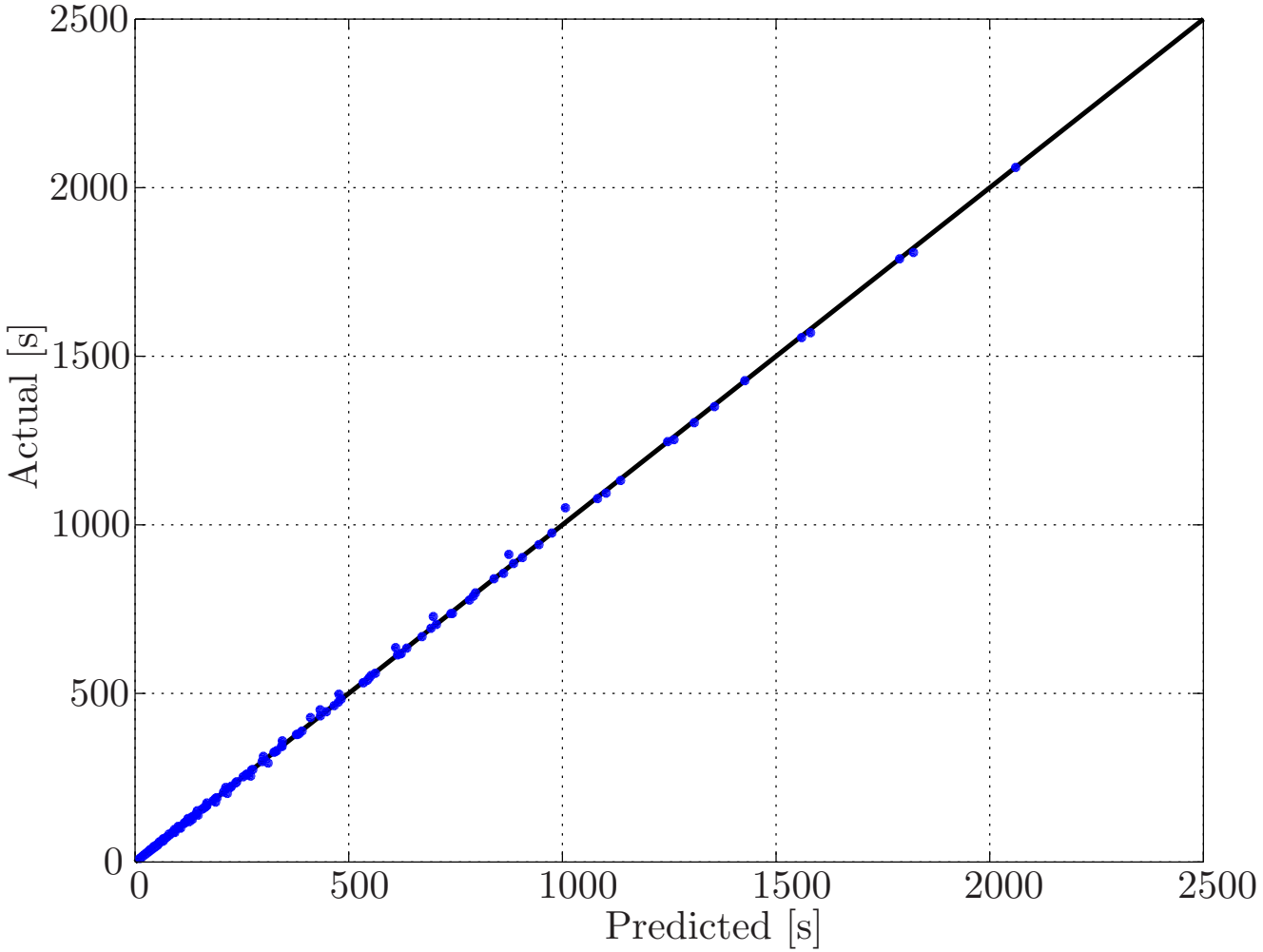
AlexNet model, AlexNet prediction

Actual vs Fitted Plot – Per Layer



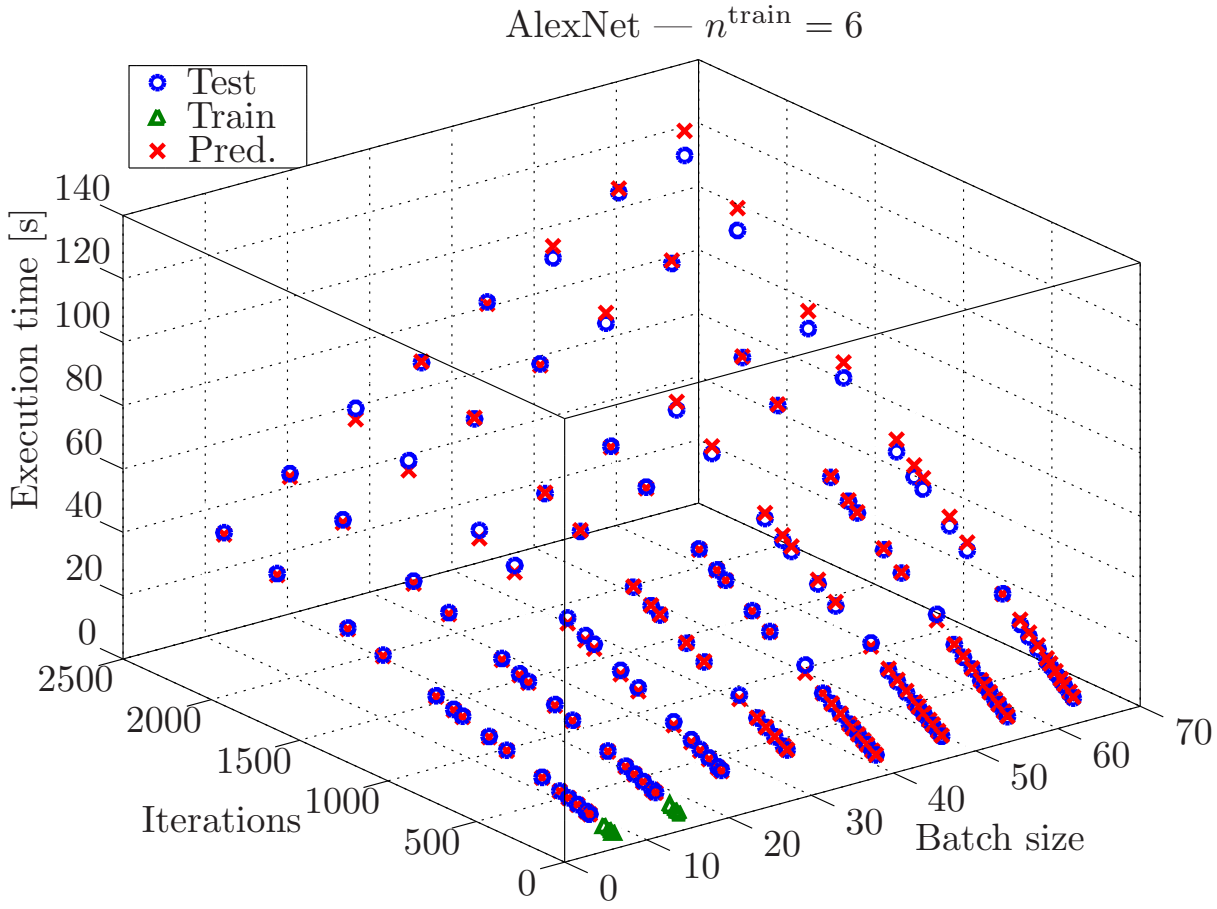
GoogLeNet model, AlexNet prediction

Actual vs Fitted Plot – End to End



VGG model, VGG prediction

End to End Models Extrapolation Capabilities



Conclusions & Future Work

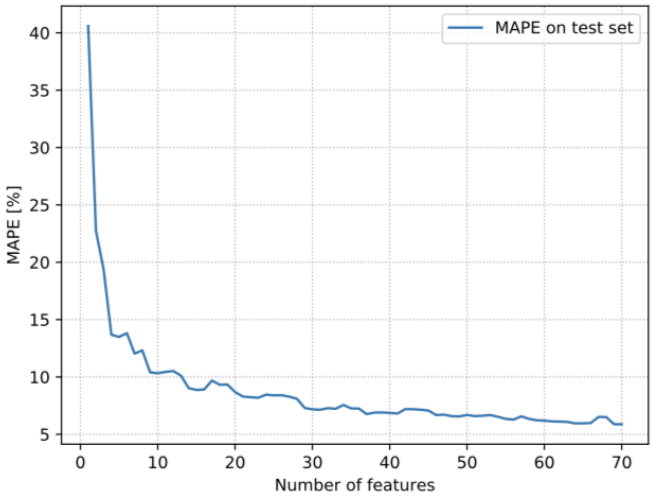
- Accurate per layer model with good generalization capability
- End to end model linked to the specific CNN, resulting in different accuracy/generality trade-offs
- Multi-GPU, multi-framework modeling
- Capacity planning based on performance prediction
- Optimal job scheduling relying on performance models

The End

Thanks
for your attention!

Multi-GPU results

- Performance prediction goals:
- Extrapolation on batch size
- Exploitation of new hardware
- Training of new versions of a CNN



Comput. Power Extr.

Network	Framework	MAPE
AlexNet	PyTorch	8.28
	TensorFlow	5.08
ResNet-50	PyTorch	18.09
	TensorFlow	10.10

GPUs number Extr.

Network	Framework	GPU Type K80	M60	GTX 1080Ti
AlexNet	PyTorch	7.21	12.18	4.98
	TensorFlow	24.75	17.27	8.77
ResNet-50	PyTorch	5.11	9.04	11.76
	TensorFlow	24.58	18.29	6.54
VGG-19	PyTorch	12.20	15.98	24.13
	TensorFlow	8.84	13.52	13.65

Extr. Inner Modules number

Network	Framework	Max N. IMs	GPU Type M60 1	2	4
ResNet	PyTorch	4	23.51	27.95	17.40
ResNet	PyTorch	5	24.85	25.11	16.75
ResNet	PyTorch	6	26.76	20.40	16.63
ResNet	PyTorch	8	17.06	7.93	15.99

Batch Size Extr.

Network	Framework	GPU Type P600 1	2	K80 1	2	3	4	M60 1	2	3	4	GTX 1080Ti 1	2	4	8
AlexNet	PyTorch	11.12	7.85	1.74	3.33	1.81	0.66	6.19	3.49	6.58	0.75	0.43	1.62	1.15	4.16
	TensorFlow	9.83	10.04	2.30	2.61	4.28	2.82	7.19	6.36	6.91	6.96	4.06	5.36	1.14	1.12
ResNet-50	PyTorch	10.64	11.97	0.76	7.83	3.09	4.53	3.60	20.04	9.58	4.64	12.62	11.93	20.63	4.29
	TensorFlow	2.37	14.35	10.25	1.27	1.84	6.83	2.08	2.79	3.07	21.49	0.68	6.44	1.43	12.06
VGG-19	PyTorch	-	-	13.88	21.71	27.63	9.65	10.74	18.54	13.81	7.68	24.98	17.40	2.93	14.06