Structured Bayesian Pruning via Log-Normal Multiplicative Noise

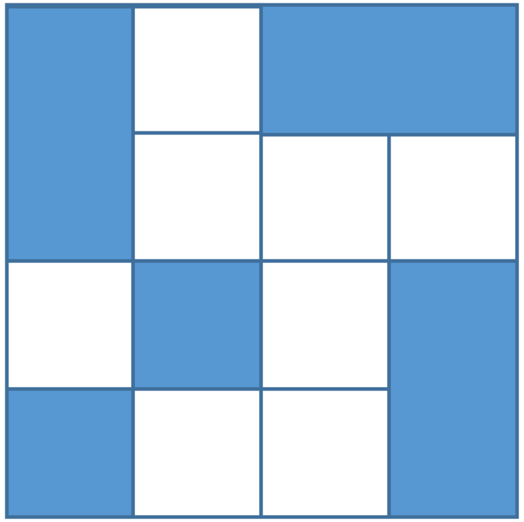
Kirill Neklyudov, Dmitry Molchanov, Arsenii Ashukha,

Dmitry Vetrov

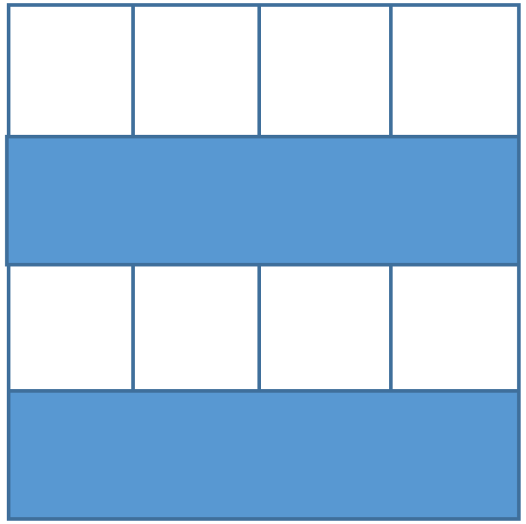
Why do we want structured sparsity?

- Acceleration of DNN
- Efficient compression
- Small representations

Unstructured



Structured



Many related papers

- Model selection and estimation in regression with grouped variables (Ming Yuan et al., 2007)
- A sparse-group Lasso (Jerome Friedman et al., 2010)
- Pruning Filters For Efficient Convnets (Hao Li et al., 2016)
- Fast ConvNets Using Group-wise Brain Damage (Vadim Lebedev et al., 2015)
- Learning Structured Sparsity in Deep Neural Networks (Wei Wen et al., 2016)
- Learning the Number of Neurons in Deep Networks (Jose M Alvarez et al., 2016)

One common idea

Group Lasso:

$$\min_{w} \frac{1}{n} \left\| t - \sum_{l} X^{(l)} w^{(l)} \right\|_2^2 + \sum_{l} \lambda_l \|w^{(l)}\|_2 + \ldots$$

As always we have

$$p(w|X,t) = \frac{p(t|X,w)p(w)}{\underbrace{\int p(t|X,w)p(w)dw}_{\text{intractable}}}$$

Variational Inference

$$\mathrm{KL}(q_\varphi(w)||p(w|X,t)) \to \min_\varphi$$

$$\mathrm{KL}(q_\varphi(w)||p(w|X,t)) = \int q_\varphi(w) \log \frac{q_\varphi(w)}{p(w|X,t)} dw =$$

$$\boxed{\mathrm{KL}(q_\varphi(w)||p(w)) - \mathbb{E}_{w\sim q_\varphi(w)} \log p(t|X,w)} + \underbrace{\log p(X,t)}_{\mathrm{const}(\varphi)}$$

Objective

Stochastic Variational Inference

$$\mathbb{E}_{w\sim q_\varphi(w)} \log p(t|X,w) \simeq \frac{n}{m}\sum_{k=1}^{m} \log p(t_{i_k}|X_{i_k}, w = f(\varphi, \varepsilon_k))$$

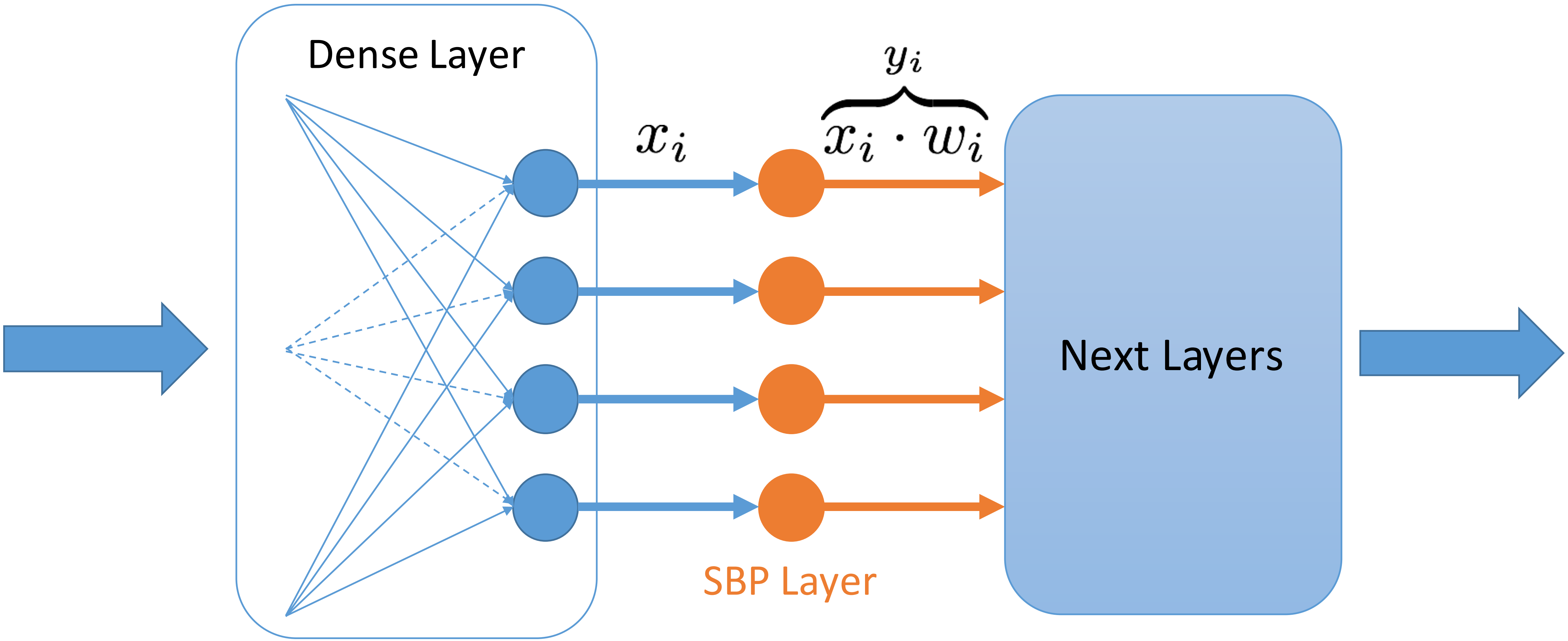
$$-\frac{n}{m}\sum_{k=1}^{m} \log p(t_{i_k}|X_{i_k}, w = f(\varphi, \varepsilon_k)) + \mathrm{KL}(q_\varphi(w)||p(w)) \to \min_{\varphi}$$

More general

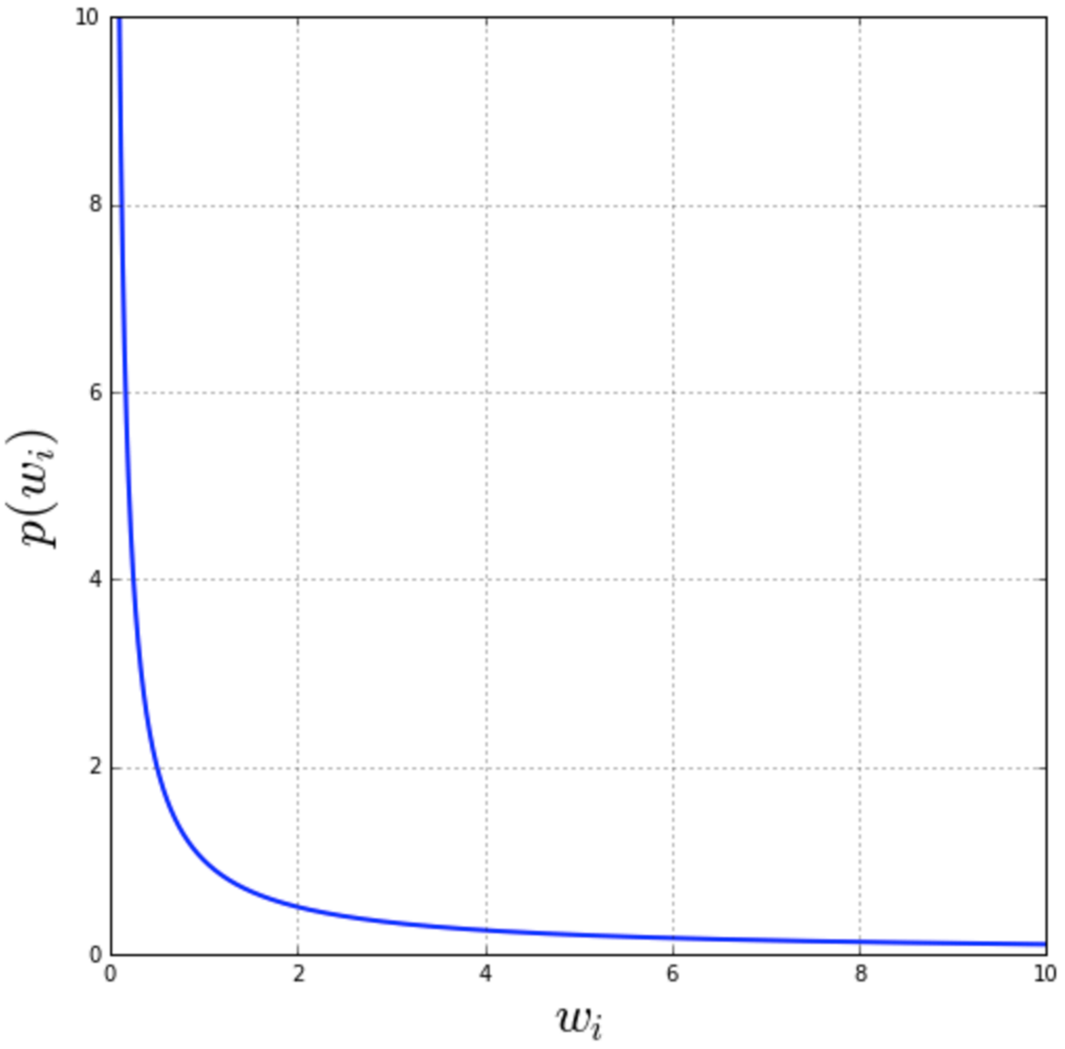
$$-\frac{n}{m}\sum_{k=1}^{m}\log p(t_{i_k}|X_{i_k}, w = f(\varphi, \varepsilon_k), W) + \mathrm{KL}(q_\varphi(w)||p(w)) \to \min_{\varphi, W},$$

where W – non-random parameters, e.g. weights of a DNN

Structured Bayesian Pruning Model



Prior



$$p(w) = \prod_i p(w_i)$$

$$p(w_i) = \mathrm{LogU}_\infty(w_i) \propto \frac{1}{w_i}, \quad w_i > 0$$

Variational Distribution

$$q_\varphi(w) = \prod_i q(w_i|\mu_i, \sigma_i) = \prod_i \mathrm{LogN}(w_i|\mu_i, \sigma_i^2)$$

$$w_i \sim \mathrm{LogN}(w_i|\mu_i, \sigma_i^2) \iff \log w_i \sim \mathcal{N}(\log w_i|\mu_i, \sigma_i^2)$$

Motivation

- No “prior gap”

$$\mathrm{LogN}(x|\mu,\sigma^2) \xrightarrow[\sigma\to\infty]{} \mathrm{LogU}_\infty(x)$$

- Only positive multiplications
  - Gaussian dropout: $\theta \sim \mathcal{N}(1,\alpha)$
  - Train: $y = x \cdot \theta$
  - Test: $y = x \cdot \mathbb{E}\theta$
- KL can be computed analytically

$$\mathrm{KL}(\mathrm{LogN}(x|\mu,\sigma^2)||\mathrm{LogU}_\infty(x)) = C - \log\sigma$$

Problems with improper prior

$$\mathrm{KL}(\mathrm{LogN}(x|\mu_i, \sigma_i^2)||\mathrm{LogU}_\infty(x)) = \mathrm{KL}(\mathcal{N}(x|\mu_i, \sigma_i^2)||\mathrm{U}(x))$$

$$\mathrm{KL}(\mathcal{N}(x|\mu_i, \sigma_i^2)||\mathrm{U}(x)) = C - \log\sigma, \quad C = +\infty$$

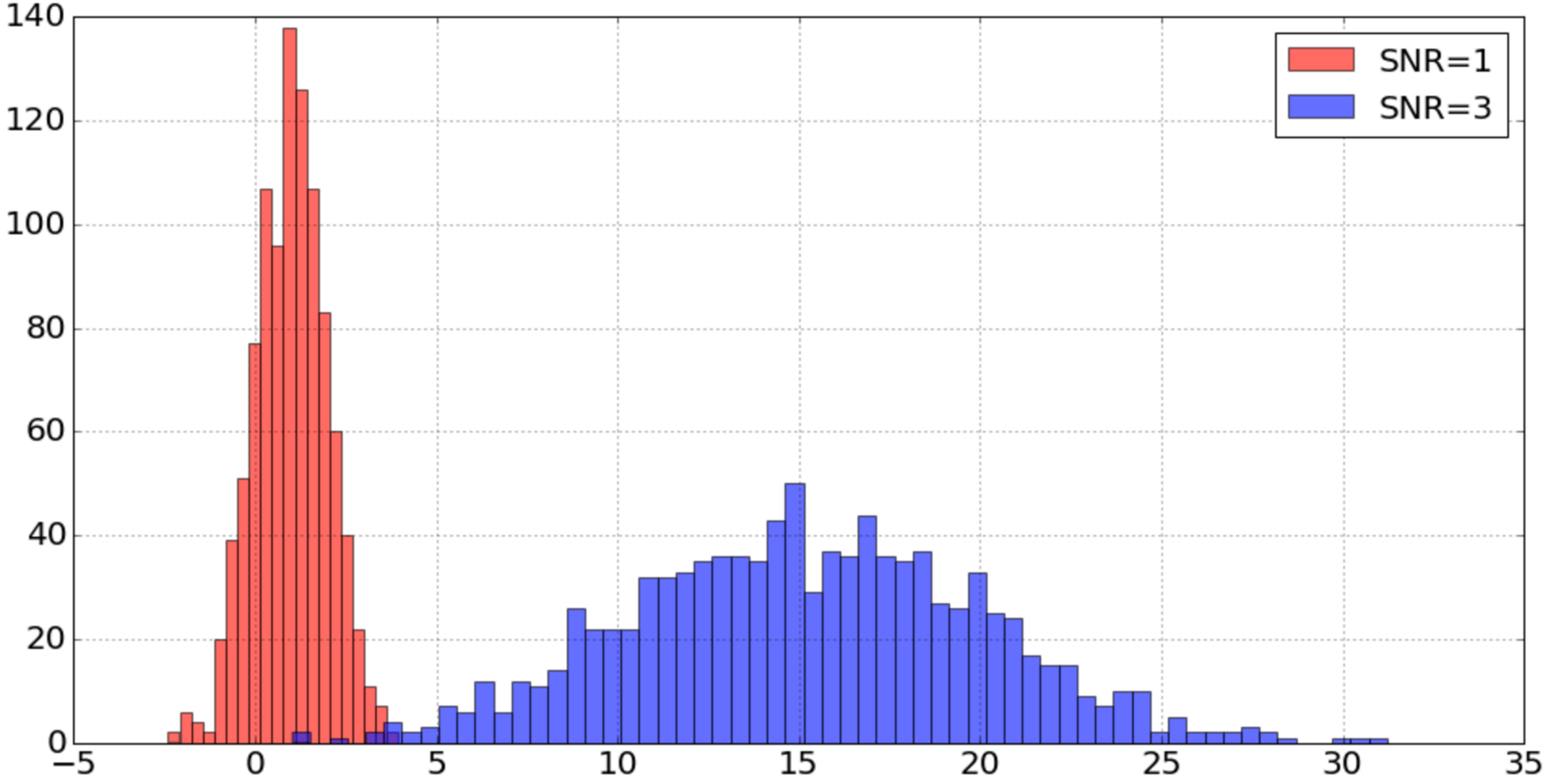
$$\arg\min_{\sigma_i} \mathrm{KL}(\mathcal{N}(x|\mu_i, \sigma_i^2)||\mathrm{U}(x)) = +\infty$$

Handling improper prior

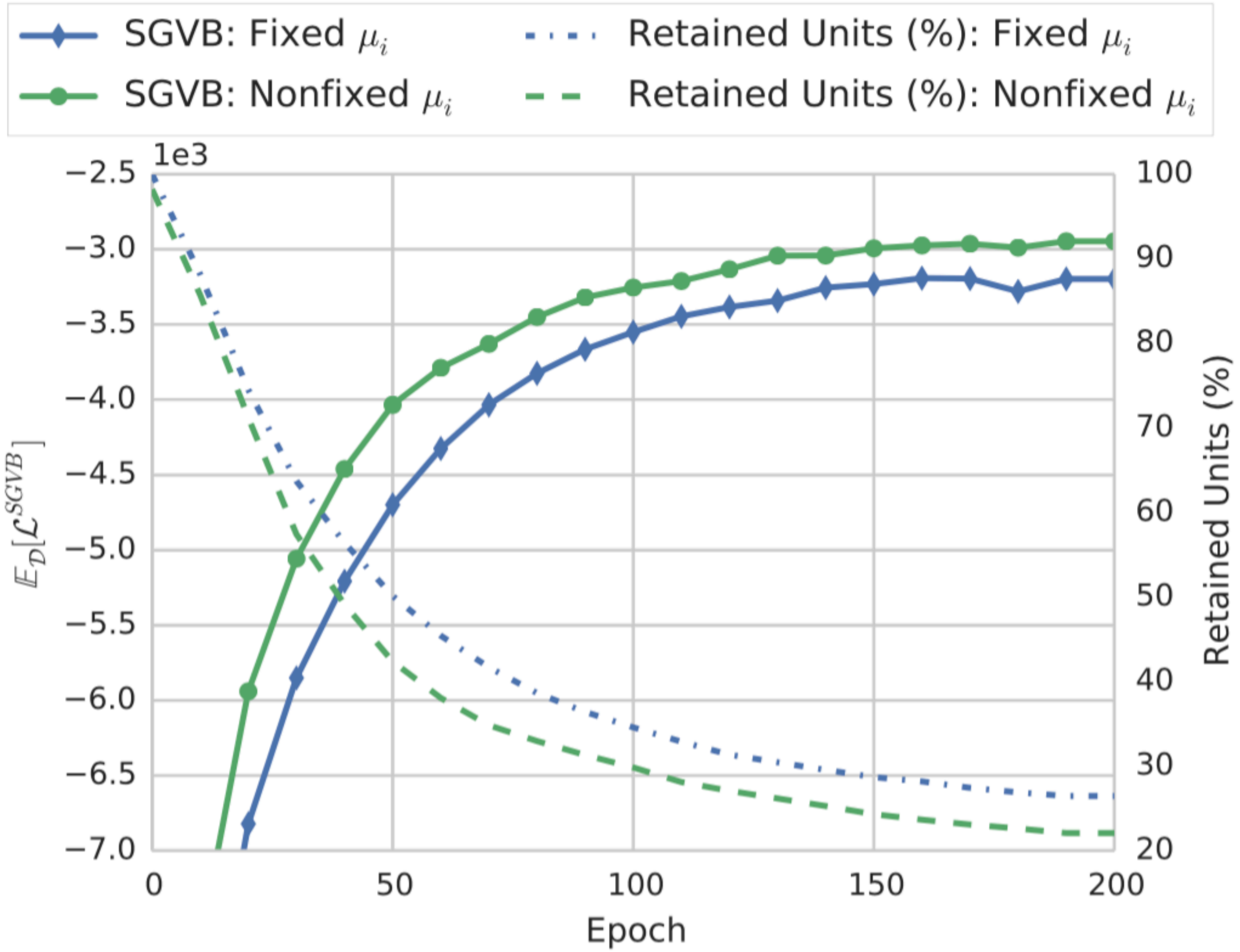
	Before	After
$p(w_i)$	$\mathrm{LogU}_{\infty}(w_i)$	$\mathrm{LogU}_{[a,b]}(w_i)$
$q_{\varphi}(w_i)$	$\mathrm{LogN}(w_i \| \mu_i, \sigma_i^2)$	$\mathrm{LogN}_{[a,b]}(w_i \| \mu_i, \sigma_i^2)$

Signal-to-Noise Ratio

$$\mathrm{SNR} = \frac{\mathbb{E}}{\sqrt{\mathbb{D}}}$$



Unfixing mu

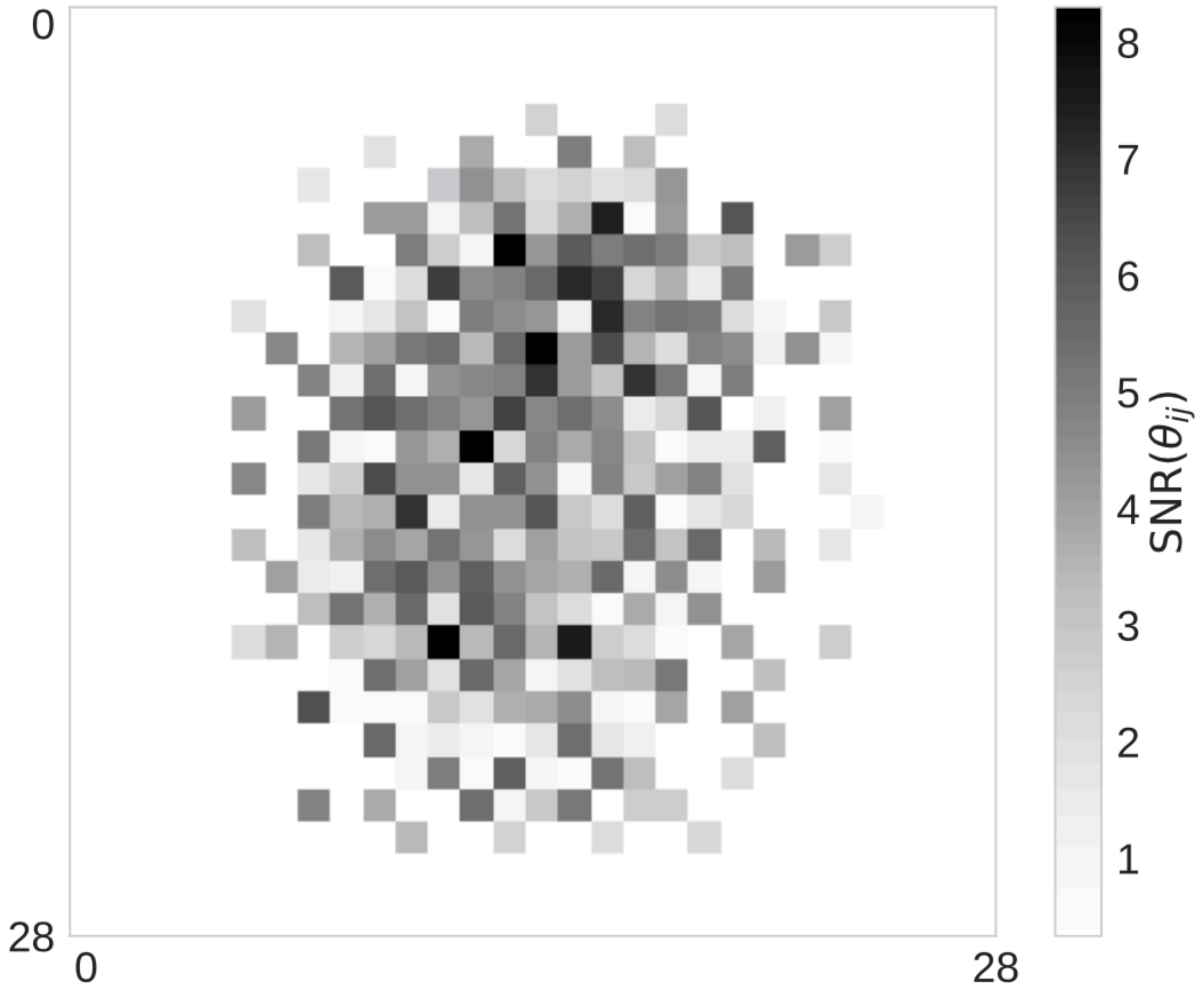


SGVB: Fixed μ_i
SGVB: Nonfixed μ_i
Retained Units (%): Fixed μ_i
Retained Units (%): Nonfixed μ_i
1e3
−2.5
−3.0
−3.5
−4.0
−4.5
−5.0
−5.5
−6.0
−6.5
−7.0
$\mathbb{E}_{\mathcal{D}}[\mathcal{L}^{SGVB}]$
100
90
80
70
60
50
40
30
20
Retained Units (%)
0
50
100
150
200
Epoch

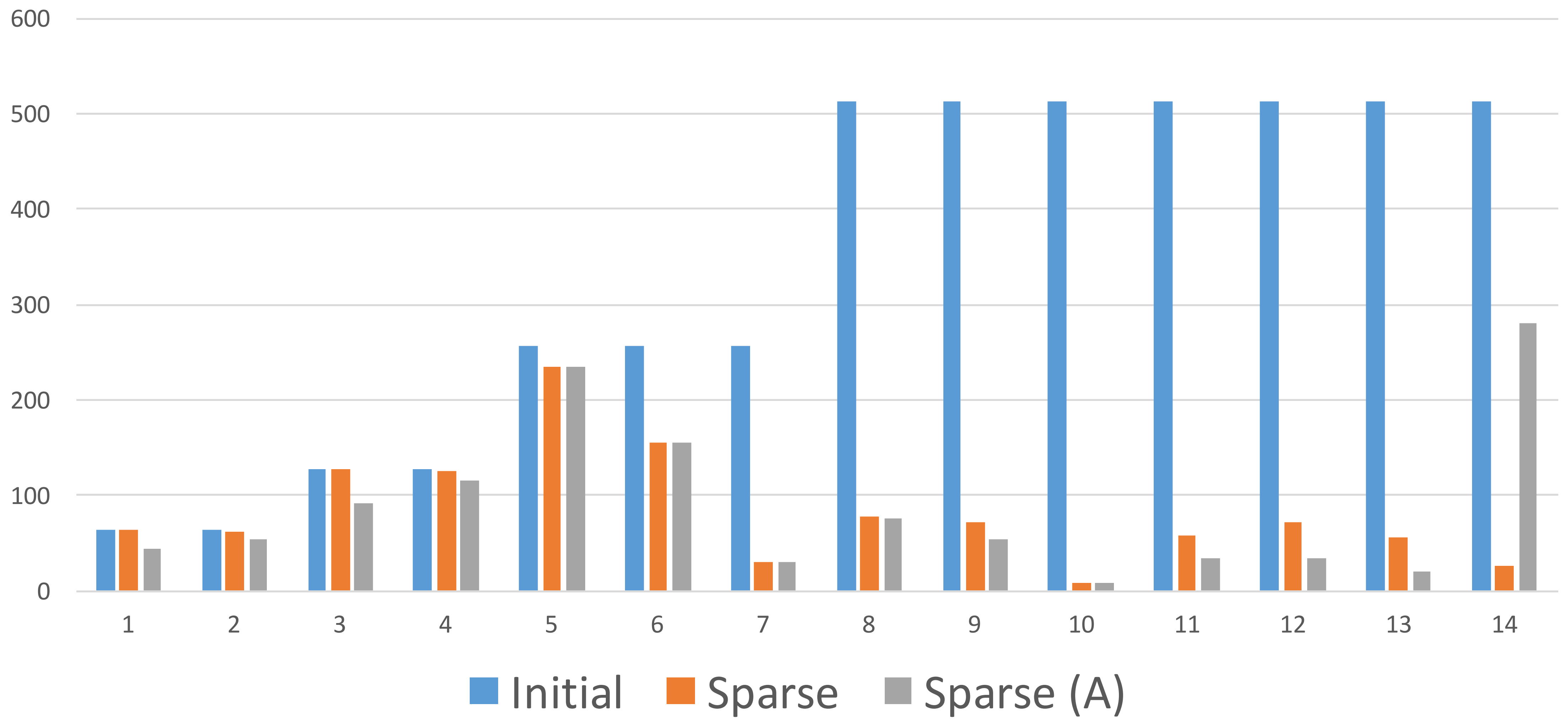
Results MNIST

Network	Method	Error %	Neurons per Layer %	CPU	GPU	FLOPs
	Original	1.54	784 – 500 – 300 – 10	1.00×	1.00×	1.00×
	SparseVD	1.57	537 – 217 – 130 – 10	1.19×	1.03×	3.73×
LeNet-500-300	SSL	1.49	434 – 174 – 78 – 10	2.21×	1.04×	6.06×
(ours)	StructuredBP	1.55	245 – 160 – 55 – 10	2.33×	1.08×	11.23×
	Original	0.80	20 – 50 – 800 – 500	1.00×	1.00×	1.00×
	SparseVD	0.75	17 – 32 – 329 – 75	1.48×	1.41×	2.19×
LeNet5-Caffe	SSL	1.00	3 – 12 – 800 – 500	5.17×	1.80×	3.90×
(ours)	StructuredBP	0.86	3 – 18 – 284 – 283	5.41×	1.91×	10.49×

Lenet500-300 feature importance



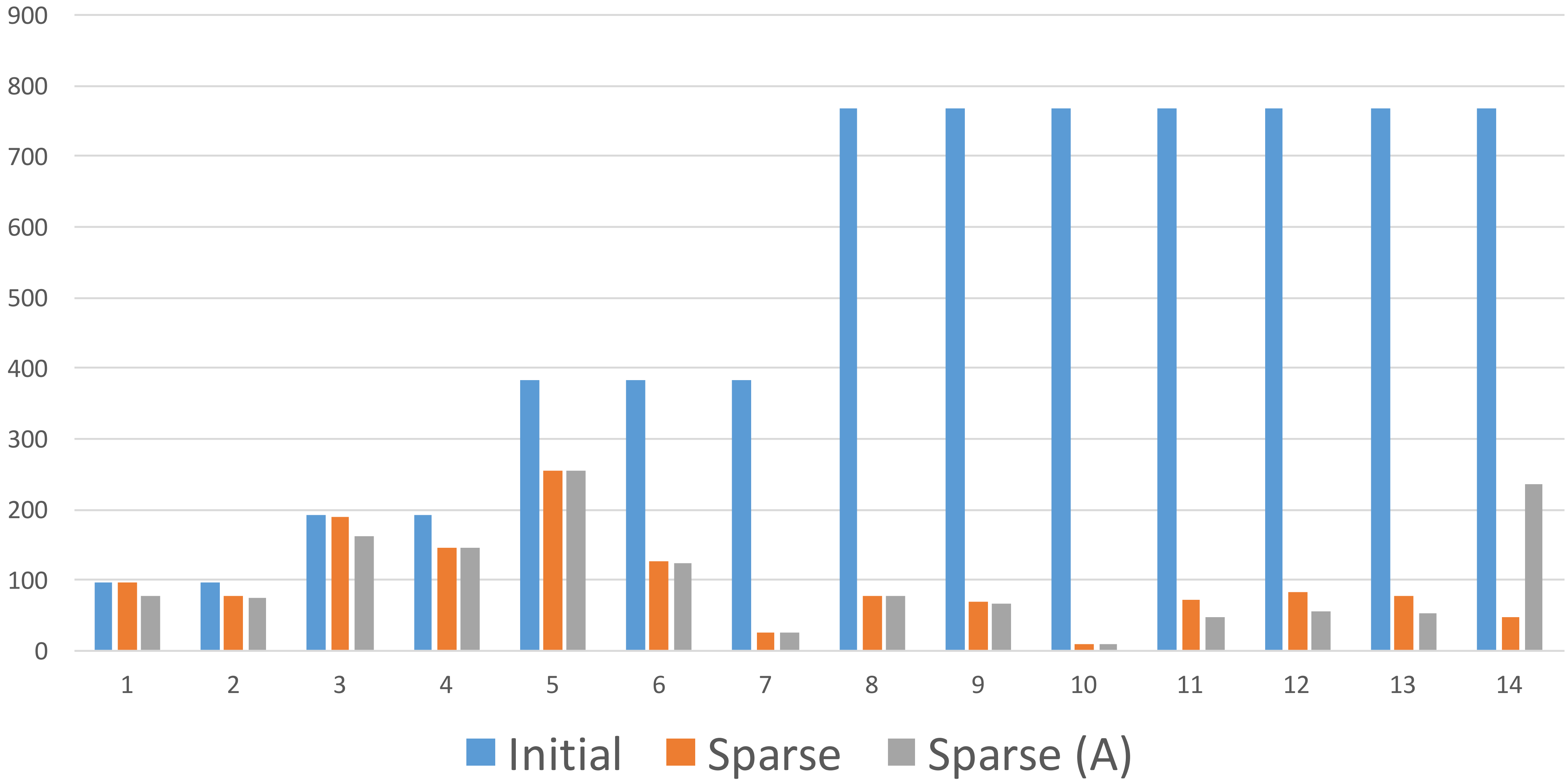
Results for VGG on CIFAR-10



Results for VGG on CIFAR-10

	Speed-up (GPU)	Error, %
Initial	x1.0	7,2
Sparse	x1.74	7,5
Sparse (A)	x2.06	9,0

Results for VGGx1.5 on CIFAR-10



Results for VGGx1.5 on CIFAR-10

	Speed-up (GPU)	Error, %
Initial	x1.0	6,8
Sparse	x2.17	7,2
Sparse (A)	x2.47	7,8

Results on Random Labeling

Dataset	Architecture	Train Accuracy	Test Accuracy	Sparsity
MNIST	Lenet5 BDO	1.0	0.1	–
MNIST	Lenet5 (ours)	0.1	0.1	100%
CIFAR10	VGG BDO	1.0	0.1	–
CIFAR10	VGG (ours)	0.1	0.1	100%

Fin

- Structured sparsity is essential for acceleration
- Proper probabilistic model
- Easy to apply
- Our paper https://arxiv.org/pdf/1705.07283.pdf