Regularization

EE807: Recent Advances in Deep Learning

Lecture 3

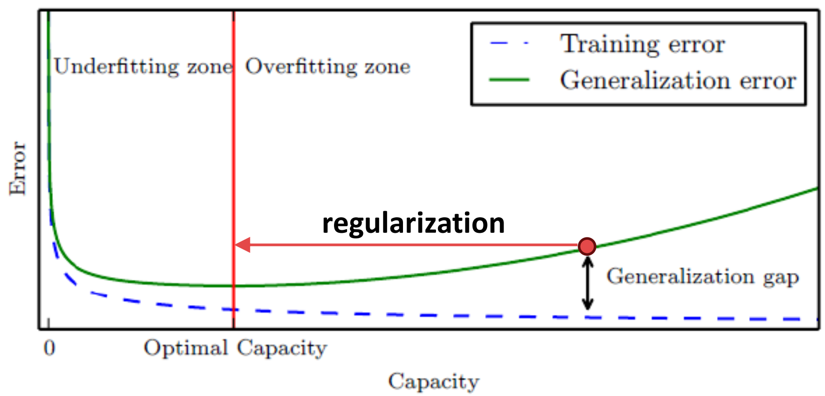
Slide made by

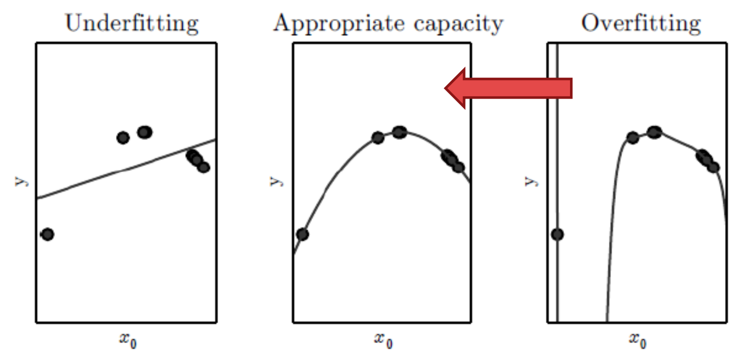
Jongheon Jeong and Insu Han

KAIST EE

"Any modification we make to a learning algorithm that is intended to reduce its generalization error, but not its training error" [Goodfellow et al., 2016]

- Regularization is a central problem in machine learning
  - Making an algorithm that perform well on new inputs, not just on the training data, i.e., there is no universal model working for all tasks
  - The main challenge is to find a right model complexity for a given task



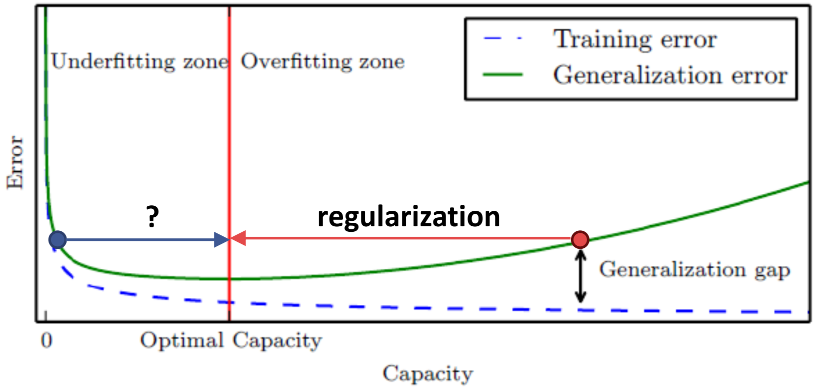


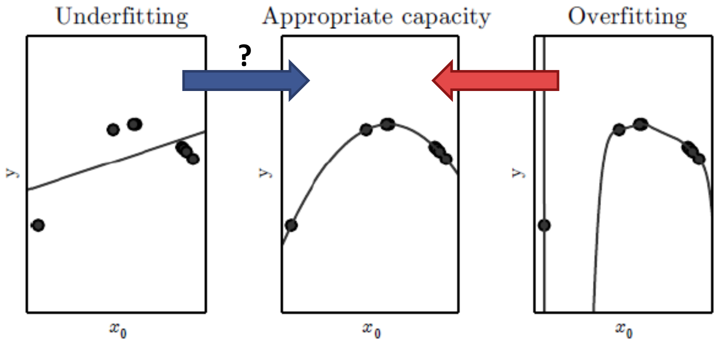
- Trading **increased bias** for **reduced variance** (bias-variance tradeoff)
- **In practice**: Introducing a *preference* between models with same training loss

*source : https://www.deeplearningbook.org/contents/ml.html

Why do we need to "regularize" our model?

- Two essential ways to find the right model complexity
  1. Increase the capacity from a small model
  2. Reduce the capacity from a large model → regularization





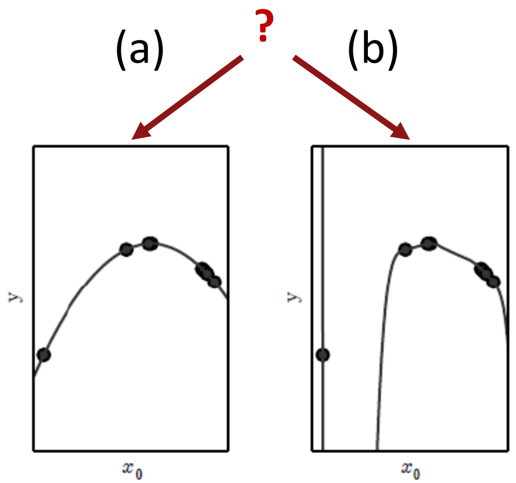
- In the context of deep learning, the latter way is more favorable
  1. **Increasing model complexity is not that simple**
     - We don't know how much the true distribution would be complex
     - "Trying to fit a square peg into a round hole" [Goodfellow et al., 2016]
  2. **Over-parametrized networks are typically easier to optimize**
     - [Dauphin et al., 2014; Goodfellow et al., 2014; Arora et al., 2018]

*source : https://www.deeplearningbook.org/contents/ml.html

How can we regularize our model?

- Specifying a **preference** between models for the optimizer

  Quiz: Which one do you prefer, (a) or (b) ?



- No free lunch theorem [Wolpert et al., 1997]

  ⇒ No best form of regularization
  - There can be another universe where (b) is correct!
  - We have to rely on **prior knowledge** of a given task

- Regularization in deep learning ≈ **Encoding prior knowledge** into the model

- Priors widely applicable for "intellectual" tasks (e.g. visual recognition, …)
  - Occam's razor → Loss penalty
  - Equivariance and invariance → Parameter sharing
  - Robustness on noise
  - …

*source : https://www.deeplearningbook.org/contents/ml.html

Table of Contents

1. **Loss Penalty**
   - Parameter norm penalty
   - Directly approximately regularizing complexity
   - Penalizing confident output distributions

2. **Parameter Sharing**
   - Convoluational neural networks
   - Equivariance through parameter-sharing
   - Appication: Movie recommendation

3. **Noise Robustness**
   - Noises on inputs or hidden units
   - Noises on model parameters
   - Noises on gradients

4. **Dataset Augmentation**
   - Making new data by local masking
   - Mixing two samples in dataset

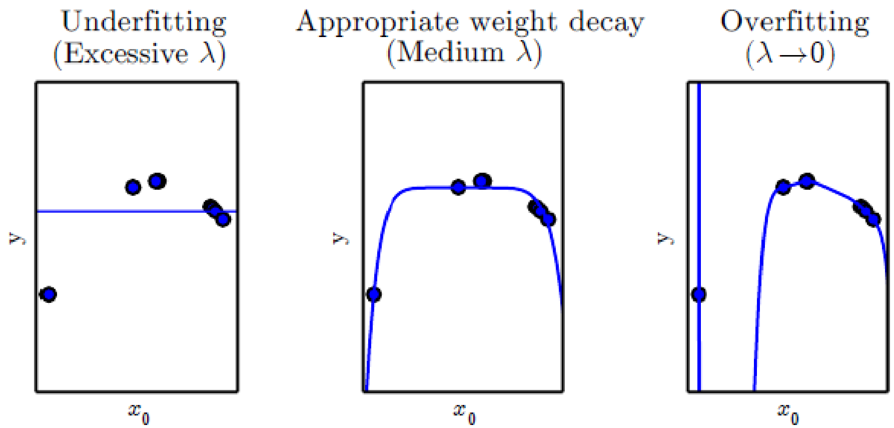
Table of Contents

- **Prior:** "Occam's razor"
  - Among hypotheses that are equally good, choose the simplest one

- Adding a parameter penalty $\Omega(\boldsymbol{\theta})$ to the objective L

$$\tilde{L}(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) = L(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) + \lambda\Omega(\boldsymbol{\theta})$$

  - $\lambda \in [0, \infty)$: a hyperparameter that controls the relative power of $\Omega(\boldsymbol{\theta})$
  - Different penalty Ω results in a different solution being preferred



Next, L^2 and L^1 regularization

*source : https://www.deeplearningbook.org/contents/ml.html

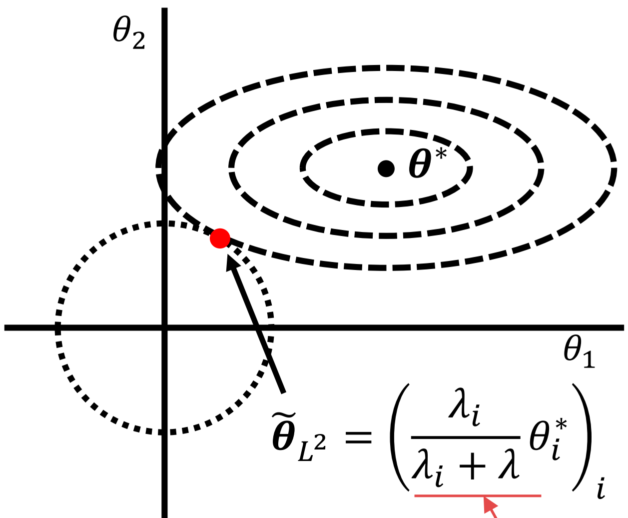
- **Parameter norm penalty**: Penalizing on the size of parameters $\boldsymbol{\theta}$
- The two most commonly used forms: $\boldsymbol{L^2}$ **and** $\boldsymbol{L^1}$ **penalty**

	$\boldsymbol{L^2}$ ("weight decay")	$\boldsymbol{L^1}$
$\Omega(\boldsymbol{\theta})$	$\frac{1}{2}\|\|\boldsymbol{\theta}\|\|_2^2 := \frac{1}{2}\sum_i \theta_i^2$	$\|\|\boldsymbol{\theta}\|\|_1 := \sum_i \|\theta_i\|$
Aliases	Ridge regression Tikhonov regularization	LASSO
MAP Prior	$\mathcal{N}(\theta_i; 0, \frac{1}{\lambda})$	$\mathrm{Laplace}(\theta_i; 0, \frac{1}{\lambda})$

- In fact, they lead the solution to the *maximum a posteriori* (MAP) estimation that a certain prior on weights is assumed

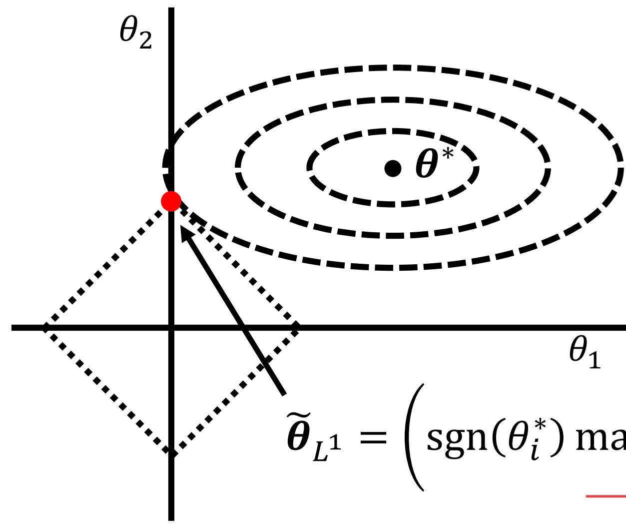
- If L is quadratic with diagonal Hessian $H = (\lambda_i)_{ii}$, we get the analytic solutions from each regularization [Goodfellow et al., 2016]:

$$\tilde{L}(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) = L(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) + \lambda\Omega(\boldsymbol{\theta})$$



$$\tilde{\boldsymbol{\theta}}_{L^2} = \left(\underline{\frac{\lambda_i}{\lambda_i + \lambda}}\theta_i^*\right)_i$$

"rescaling" eigenvalues



$$\tilde{\boldsymbol{\theta}}_{L^1} = \left(\mathrm{sgn}(\theta_i^*) \underline{\max\left\{|\theta_i^*| - \frac{\lambda}{\lambda_i}, 0\right\}}\right)_i$$

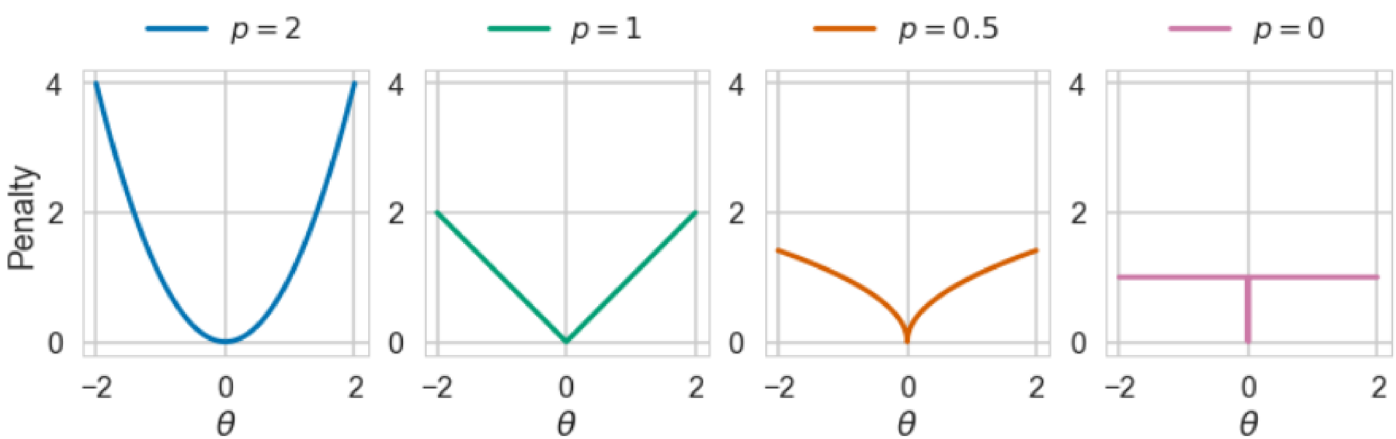
"sparse" solution

$\boxed{\text{Next, } L^0\text{-regularization}}$

Parameter norm penalty: L^0-regularization [Louizos et al., 2018]

- We typically use the popular L^1-regularization to induce sparsity
  - Sparse models are advantageous on computational efficiency
  - Of course, it is a nice policy for regularization as well

- Why don't we use $\boldsymbol{L^0}$**-penalty**?
  - $\Omega(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_0 \coloneqq |\{\theta_i : \theta_i \neq 0\}|$
  - A more direct measure of sparsity
  - It does not shrink the non-sparse weights



*source : Louizos et al. "Learning Sparse Neural Networks through L_0 Regularization", ICLR 2017

- We typically use the popular L^1-regularization to induce sparsity
  - Sparse models are advantageous on computational efficiency
  - Of course, it is a nice policy for regularization as well

- Why don't we use $\boldsymbol{L^0}$**-penalty**?
  - $\Omega(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_0 \coloneqq |\{\theta_i : \theta_i \neq 0\}|$
  - A more direct measure of sparsity
  - It does not shrink the non-sparse weights

- **Problem:** Optimization with L^0-penalty is intractable in general
  - Discrete optimization with $2^{|\boldsymbol{\theta}|}$ possible states
  - Standard gradient-based methods are not applicable

- Can we relax this problem so that to an efficient continuous optimization?

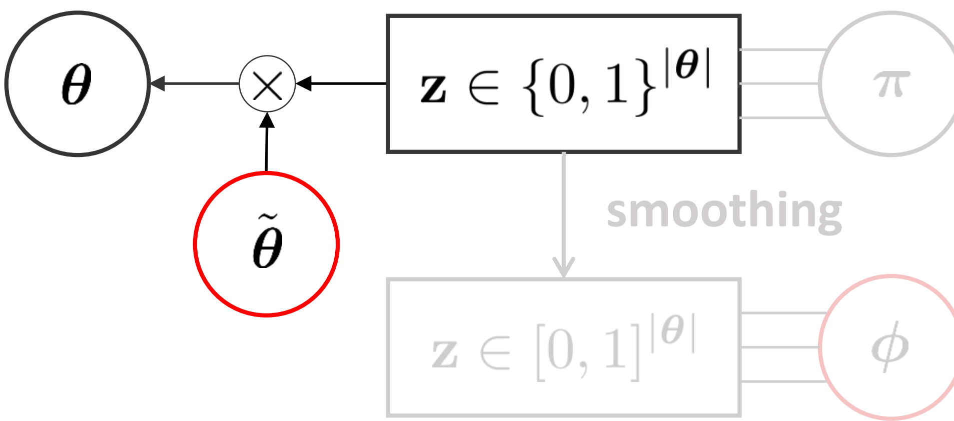
*source : Louizos et al. "Learning Sparse Neural Networks through L_0 Regularization", ICLR 2017

Parameter norm penalty: L^0-regularization [Louizos et al., 2018]

- **Idea:** Regard $\boldsymbol{\theta}$ as a random variable, where $\mathbb{E}[||\boldsymbol{\theta}||_0]$ is differentiable
  1. Consider a simple **re-parametrization** of $\boldsymbol{\theta}$:

$$\theta_j = \tilde{\theta}_j z_j, \quad z_j \in \{0,1\}, \quad \tilde{\theta}_j \neq 0$$

  - Then, the L^0-penalty becomes $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_0 = \sum_{j=0}^{|\theta|} z_j$

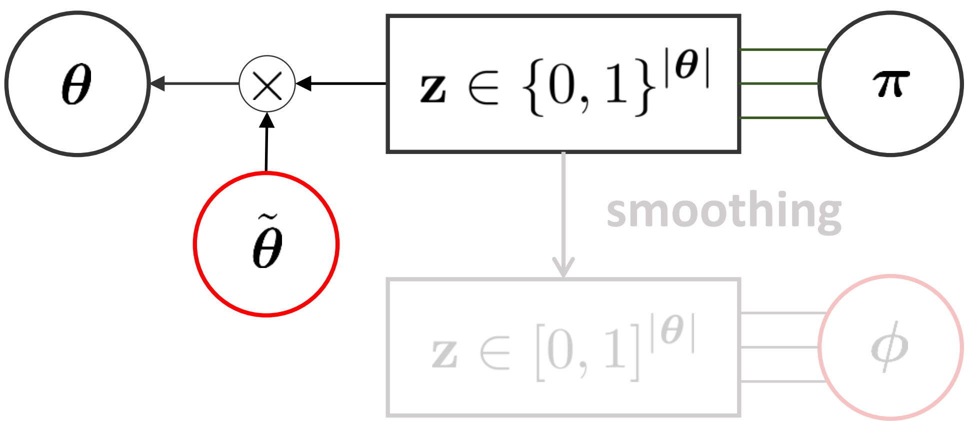


*source : Louizos et al. “Learning Sparse Neural Networks through L_0 Regularization”, ICLR 2017

- **Idea:** Regard $\boldsymbol{\theta}$ as a random variable, where $\mathbb{E}[\|\boldsymbol{\theta}\|_0]$ is differentiable
  2. Letting $q(z_i|\pi_i) = \text{Bernoulli}(\pi_i)$, we define the **expected loss** $\mathcal{R}$:

$$\mathcal{R}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\pi}) := \mathbb{E}_{q(\mathbf{z}|\boldsymbol{\pi})}\left[L(\tilde{\boldsymbol{\theta}} \odot \mathbf{z})\right] + \lambda \sum_{j=1}^{|\theta|} \pi_j$$

  - However, optimizing $\mathcal{R}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\pi})$ is still hard
    - Estimating $\nabla \mathbb{E}_{q(\mathbf{z}|\boldsymbol{\pi})}\left[L(\tilde{\boldsymbol{\theta}} \odot \mathbf{z})\right]$ is not easy due to the discrete nature of $\mathbf{z}$



*source : Louizos et al. "Learning Sparse Neural Networks through L_0 Regularization", ICLR 2017

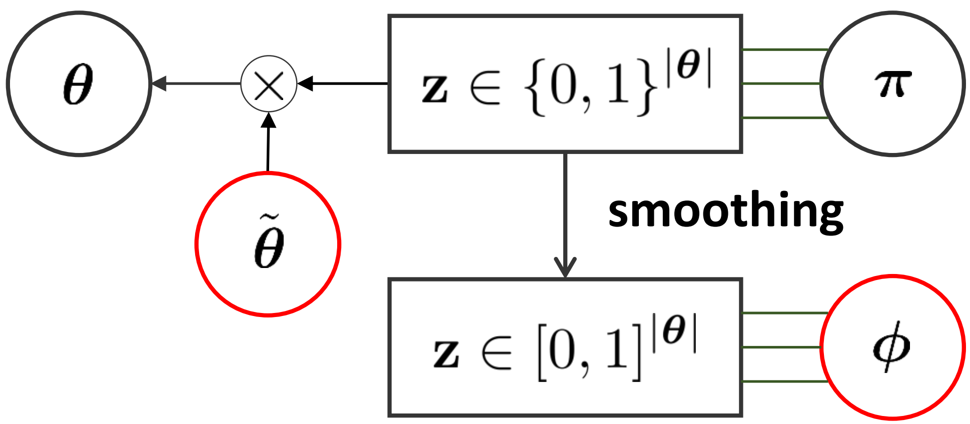
Parameter norm penalty: L^0-regularization [Louizos et al., 2018]

- **Idea:** Regard $\boldsymbol{\theta}$ as a random variable, where $\mathbb{E}[\|\boldsymbol{\theta}\|_0]$ is differentiable
  3. Smoothing the discrete r.v. $\mathbf{z}$ via a continuous r.v. $\mathbf{s}$:

$$\mathbf{z} = \min(1, \max(0, \mathbf{s})), \quad \mathbf{s} \sim q(\mathbf{s}|\boldsymbol{\phi})$$

  - Since $q(\mathbf{z} \neq 0|\boldsymbol{\phi}) = 1 - \mathbb{P}(\mathbf{s} \leq 0|\boldsymbol{\phi})$, we get:

$$\mathcal{R}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi}) = \mathbb{E}_{q(\mathbf{s}|\boldsymbol{\phi})}\left[L(\tilde{\boldsymbol{\theta}} \odot \min(1, \max(0, \mathbf{s})))\right] + \lambda \sum_{j=1}^{|\theta|} (1 - \mathbb{P}(s_j \leq 0|\phi_j))$$

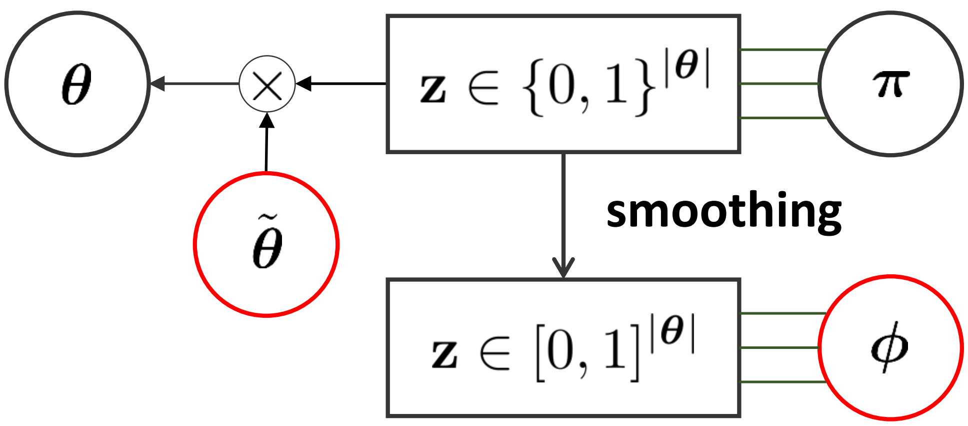


*source : Louizos et al. "Learning Sparse Neural Networks through L_0 Regularization", ICLR 2017

- **Idea:** Regard $\boldsymbol{\theta}$ as a random variable, where $\mathbb{E}[\|\boldsymbol{\theta}\|_0]$ is differentiable
  - Finally, the original loss $\tilde{L}$ is transformed by:

$$\mathcal{R}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi}) = \mathbb{E}_{q(\mathbf{s}|\boldsymbol{\phi})}\left[L(\tilde{\boldsymbol{\theta}} \odot \min(1, \max(0, \mathbf{s})))\right] + \lambda \sum_{j=1}^{|\theta|} \left(1 - \mathbb{P}\left(s_j \leq 0 | \phi_j\right)\right)$$

  - We can optimize this via minibatch-based gradient estimation methods
    - For details, see [Kingma et al., 2013]



*source : Louizos et al. "Learning Sparse Neural Networks through L_0 Regularization", ICLR 2017

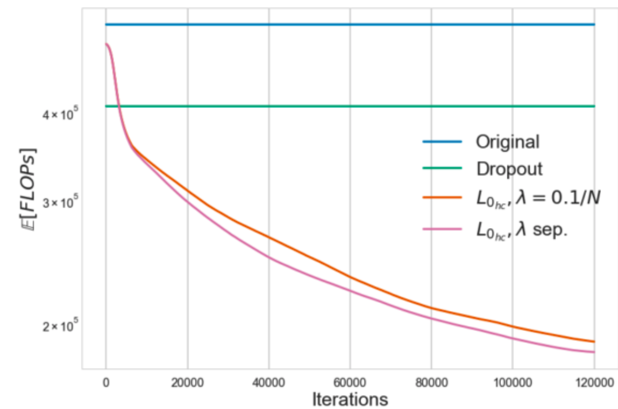
Parameter norm penalty: L^0-regularization [Louizos et al., 2018]

- L^0-regularization leads the networks to a sparse solution, with a good regularization as well on MNIST and CIFAR-10/100

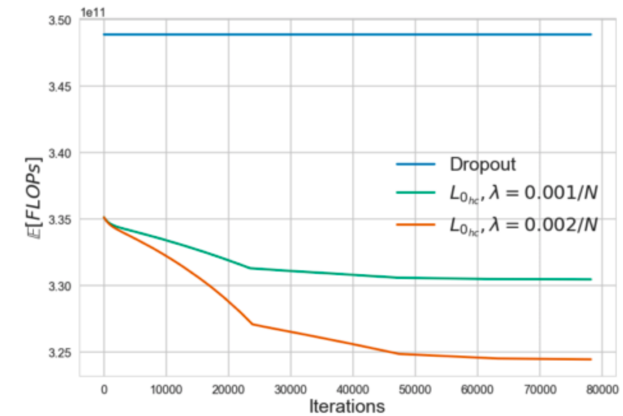
Network & size	Method	Pruned architecture	Error (%)
MLP 784-300-100	Sparse VD (Molchanov et al., 2017)	512-114-72	1.8
	BC-GNJ (Louizos et al., 2017)	278-98-13	1.8
	BC-GHS (Louizos et al., 2017)	311-86-14	1.8
	$L_{0_{hc}}, \lambda = 0.1/N$	219-214-100	1.4
	$L_{0_{hc}}, \lambda$ sep.	266-88-33	1.8
LeNet-5-Caffe 20-50-800-500	Sparse VD (Molchanov et al., 2017)	14-19-242-131	1.0
	GL (Wen et al., 2016)	3-12-192-500	1.0
	GD (Srinivas & Babu, 2016)	7-13-208-16	1.1
	SBP (Neklyudov et al., 2017)	3-18-284-283	0.9
	BC-GNJ (Louizos et al., 2017)	8-13-88-13	1.0
	BC-GHS (Louizos et al., 2017)	5-10-76-16	1.0
	$L_{0_{hc}}, \lambda = 0.1/N$	20-25-45-462	0.9
	$L_{0_{hc}}, \lambda$ sep.	9-18-65-25	1.0

Network	CIFAR-10	CIFAR-100
original-ResNet-110 (He et al., 2016a)	6.43	25.16
pre-act-ResNet-110 (He et al., 2016b)	6.37	-
WRN-28-10 (Zagoruyko & Komodakis, 2016)	4.00	21.18
WRN-28-10-dropout (Zagoruyko & Komodakis, 2016)	3.89	18.85
WRN-28-10-$L_{0_{hc}}, \lambda = 0.001/N$	**3.83**	**18.75**
WRN-28-10-$L_{0_{hc}}, \lambda = 0.002/N$	3.93	19.04

MLP



WRN-28-10



Next, complexity regularization

*source : Louizos et al. “Learning Sparse Neural Networks through L_0 Regularization”, ICLR 2017

- Reducing complexity of a model might be a direct way of regularization
  - But, how do we know whether a model is complex or not?
  - **Computational learning theory** provides a way for it

- Suppose we have a **model** F, i.e. a set of hypothesis functions
- **DARC** attempts to reduce the **Rademacher complexity** of F:

$$\mathrm{Rad}_m(F) := \mathbb{E}_{\mathbf{x}\sim\mathcal{D}^m}\left[\frac{1}{m}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{f\in F}\sum_{i=1}^{m}\sigma_i f(x_i)\right]\right]$$

sample size

  - $\sigma_1, \ldots, \sigma_m$: *i.i.d.* random variables, $\mathbb{P}(\sigma_i = 1) = \mathbb{P}(\sigma_i = -1) = \frac{1}{2}$
  - High $\mathrm{Rad}_m(F) \Rightarrow F$ is more expressive on $\mathcal{D}^m$
  - It can be used to give a bound of the generalization error in ERM
    - For details, see [Shalev-Shwartz et al., 2014]

*source : Kawaguchi et al. "Generalization in Deep Learning", Arxiv 2017

Directly approximately regularizing complexity (DARC) [Kawaguchi et al., 2017]

- **DARC** attempts to reduce the **Rademacher complexity** of F :

$$\mathrm{Rad}_m(F) := \mathbb{E}_{\mathbf{x}\sim\mathcal{D}^m}\left[\frac{1}{m}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{f\in F}\sum_{i=1}^{m}\sigma_i f(x_i)\right]\right]$$

sample size

- Of course, computing $\mathrm{Rad}_m(F)$ is intractable when F is a family of NNs
- Instead, DARC uses a rough approximation of it:

$$\tilde{L}(\boldsymbol{\theta};\mathbf{X},\mathbf{y}) = L(\boldsymbol{\theta};\mathbf{X},\mathbf{y}) + \lambda \boxed{\left(\frac{1}{m}\max_k \sum_{i=1}^{m}|f_k(x_i;\boldsymbol{\theta})|\right)}$$

mini-batch size

  - $f = (f_1, \cdots, f_d) \in \mathbb{R}^d$: the model to optimize (e.g. neural network)
  - In other words, here F is approximated by $\{f_k : k = 1, \cdots, d\}$

*source : Kawaguchi et al. "Generalization in Deep Learning", Arxiv 2017

Directly approximately regularizing complexity (DARC) [Kawaguchi et al., 2017]

- Despite its simplicity, DARC improves state-of-the-art level models
  - Results on MNIST and CIFAR-10 are presented

Table 1: Test error (%). A standard variant of LeNet (LeCun et al., 1998) and ResNeXt-29(16×64d) (Xie et al., 2016) are used for MNIST and CIFAR-10, and compared with the addition of the studied regularizer.

Method	MNIST	CIFAR-10
Baseline	0.26	3.52
DARC1	**0.20**	**3.43**

- Comparisons in the values of DARC penalty
  - Data augmentation by itself implicitly regularize the DARC penalty

Table 3: Values of $\frac{1}{m}\left(\max_k \sum_{i=1}^{m} |h_k^{(H+1)}(x_i)|\right)$

Method	MNIST (ND) mean	MNIST (ND) stdv	MNIST mean	MNIST stdv	CIFAR-10 mean	CIFAR-10 stdv
Base	17.2	2.40	8.85	0.60	12.2	0.32
DARC1	1.30	0.07	1.35	0.02	0.96	0.01

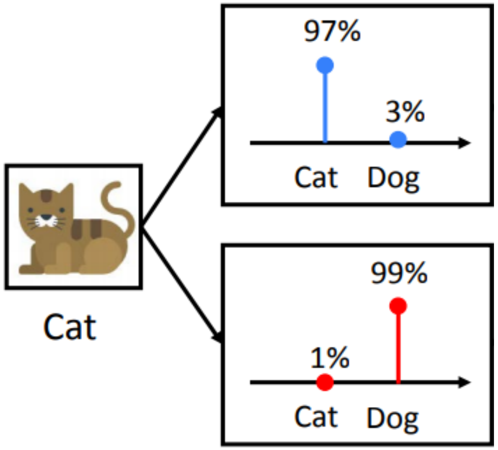
(ND) = no data augmentation

Next, Confidence penalty

*source : Kawaguchi et al. “Generalization in Deep Learning”, Arxiv 2017

- Regularization by preventing a network not to be **over-confident**

- **Confident predictions**: Output distributions that have low entropy
  - Placing all probability on a single class
  - **Overly-confident** predictions are often a sign of overfitting [Szegedy et al., 2015]



- Adding the negative entropy to loss prevents the over-confidence

$$\tilde{L}(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) = L(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) \underbrace{- \lambda H(p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{X}))}_{\text{negative entropy}}$$

*source :
Pereyra et al. "Regularizing Neural Networks by Penalizing Confident Output Distributions", ICLR 2017 Workshop

Penalizing confident output distributions [Pereyra et al., 2017]

- Confidence penalty improves generalization for various datasets

Model	Layers	Size	Test
Wan et al. (2013) - Unregularized	2	800	1.40%
Srivastava et al. (2014) - Dropout	3	1024	1.25%
Wan et al. (2013) - DropConnect	2	800	1.20%
Srivastava et al. (2014) - MaxNorm + Dropout	2	8192	**0.95%**
Dropout	2	1024	$1.28 \pm 0.06\%$
Label Smoothing	2	1024	$1.23 \pm 0.06\%$
Confidence Penalty	2	1024	$\mathbf{1.17 \pm 0.06}\%$

Table 3: Test error (%) for permutation-invariant MNIST.

Model	Layers	Parameters	Test
He et al. (2015) - Residual CNN	110	1.7M	13.63%
Huang et al. (2016b) - Stochastic Depth Residual CNN	110	1.7M	11.66%
Larsson et al. (2016) - Fractal CNN	21	38.6M	10.18%
Larsson et al. (2016) - Fractal CNN (Dropout)	21	38.6M	7.33%
Huang et al. (2016a) - Densely Connected CNN	40	1.0M	7.00%
Huang et al. (2016a) - Densely Connected CNN	100	7.0M	**5.77%**
Densely Connected CNN (Dropout)	40	1.0M	7.04%
Densely Connected CNN (Dropout + Label Smoothing)	40	1.0M	6.89%
Densely Connected CNN (Dropout + Confidence Penalty)	40	1.0M	**6.77**%

Table 4: Test error (%) on Cifar-10 without data augmentation.

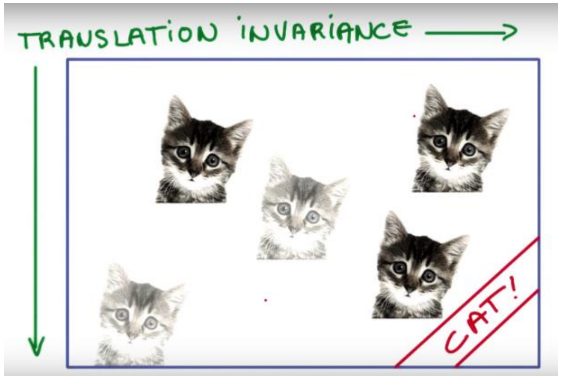
*source :
Pereyra et al. "Regularizing Neural Networks by Penalizing Confident Output Distributions", ICLR 2017 Workshop

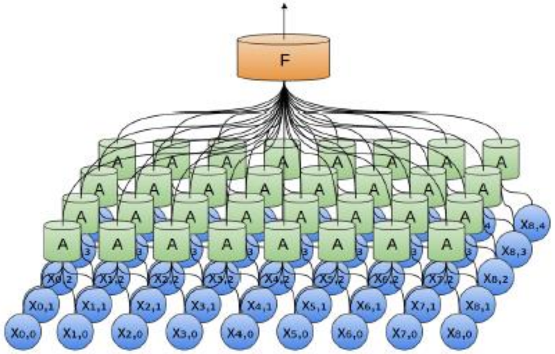
Table of Contents

Parameter sharing: Convolutional neural networks

- **Prior:** Good representations may contain **equivariance** or **invariance**
  - **Parameter sharing** is a good way to encode equivariance or invariance of features

- **Example**: Convolutional neural networks
  - Sharing parameters across multiple image locations
  - Natural images have many statistical properties that are invariant to translation
  - Due to the sharing, CNNs have dramatically less parameters compared to DNNs with strong generalization ability





Next, equivariance through parameter-sharing

*sources :
- https://www.udacity.com/course/deep-learning--ud730
- http://colah.github.io/posts/2014-07-Conv-Nets-Modular/

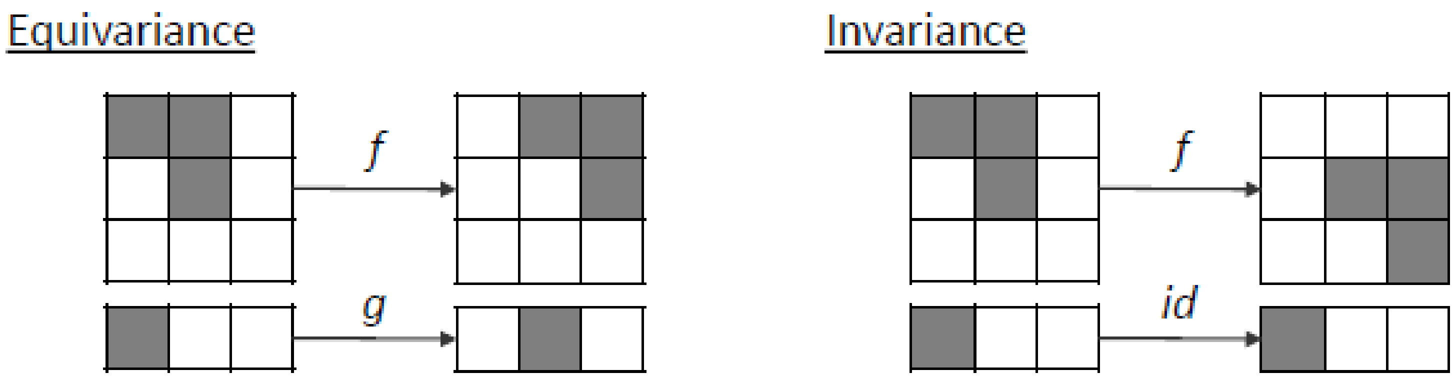
- **Prior:** Good representations may contain **equivariance** or **invariance**
  - **Parameter sharing** is a good way to encode equivariance or invariance of features

Definition $\phi : \mathbb{R}^N \to \mathbb{R}^M$ is $(\boldsymbol{f}, \boldsymbol{g})$**-equivariant** *iff* $\phi(f(\mathbf{x})) = g(\phi(\mathbf{x})), \forall \mathbf{x} \in \mathbb{X}^N$

Definition $\phi : \mathbb{R}^N \to \mathbb{R}^M$ is $\boldsymbol{f}$**-invariant** *iff* it is (f, id)-equivariant

  - $\mathrm{id}: \mathbb{R}^M \to \mathbb{R}^M$ is the identity function

- Equivariance is a more general concept of invariance



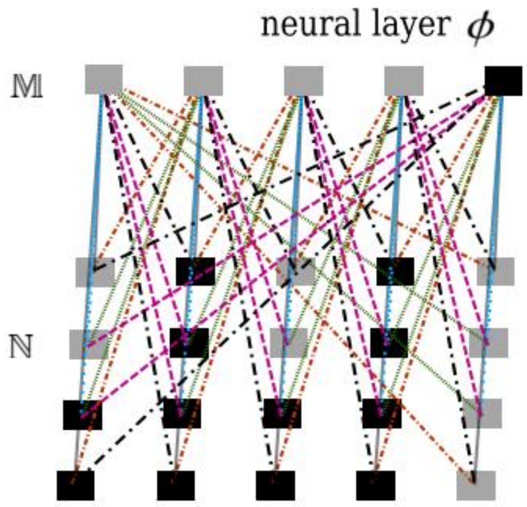
Equivariance through parameter-sharing [Ravanbakhsh et al., 2017]

- Parameter-sharing in ϕ is related to equivariance on permutations of indices
  - "*indices*": $\mathbb{N} = [1, \dots, N]$ (input) and $\mathbb{M} = [1, \dots, M]$ (output)
  - Formally, of the form $G_{\mathbb{N},\mathbb{M}} \leq \mathcal{S}_{\mathbb{N}} \times \mathcal{S}_{\mathbb{M}}$, where $\mathcal{S}_X$: The symmetric group on X

- Consider a coloring of weights between input and output

<u>Definition</u> A **colored multi-edged bipartite graph** $\Omega = (\mathbb{N}, \mathbb{M}, \alpha)$ is a triple, where

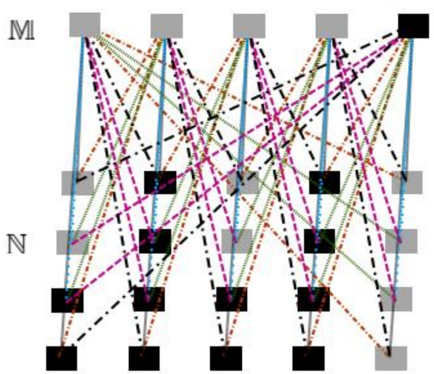
  - $\alpha: \mathbb{N} \times \mathbb{M} \to 2^{\{1,\dots,C\}}$; The edge functions that assigns colors
  - Non-existing edges receives no color



*source : Ravanbakhsh et al. "Equivariance Through Parameter-Sharing", ICML 2017

- Suppose that "edges of the same color = shared parameters"
  - $\mathbf{w} = [\mathrm{w}_1, \ldots, \mathrm{w}_C]$, C parameters in total
- Consider a neural network layer ϕ constructed from Ω
  - σ: a strictly monotonic non-linearity

$$\phi(\mathbf{x}; \mathbf{w}, \Omega) := \sigma\left(\sum_{n} \sum_{c \in \alpha(n,m)} \mathrm{w}_c x_n\right)$$



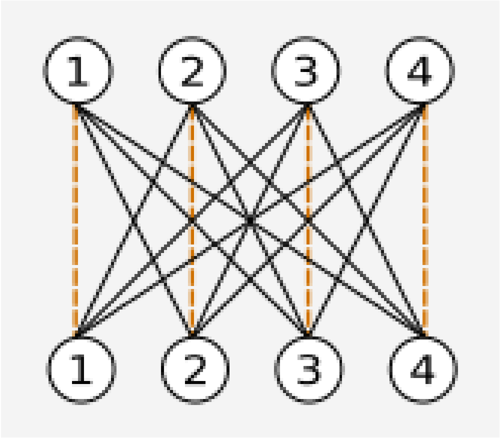
> **Theorem** (Ravanbakhsh et al.) $\phi(\mathbf{x}, \mathbf{w}, \Omega)$ is *equivariant* on any permutations among the same-colored edges, for any Ω.

*source : Ravanbakhsh et al. "Equivariance Through Parameter-Sharing", ICML 2017

Equivariance through parameter-sharing [Ravanbakhsh et al., 2017]

Theorem $\phi(\mathbf{x}, \mathbf{w}, \Omega)$ is *equivariant* on permutations among same-colored edges.

- **Example**: Permutation-equivariant layer
  - Ω constists $\mathbb{N} = \mathbb{M} = [1, 2, 3, 4]$, and α of 2 colors
  - $\phi(\mathbf{x}; \mathbf{w} = [w_1, w_2], \Omega) = \sigma((w_1 \mathbf{I} + w_2(\mathbf{1}\mathbf{1}^T - \mathbf{I}))\mathbf{x})$
  - Then, ϕ is equivariant on $\{(g, g) |\ g \in \mathcal{S}_{\mathbb{N}}\} \cong \mathcal{S}_{\mathbb{N}}$

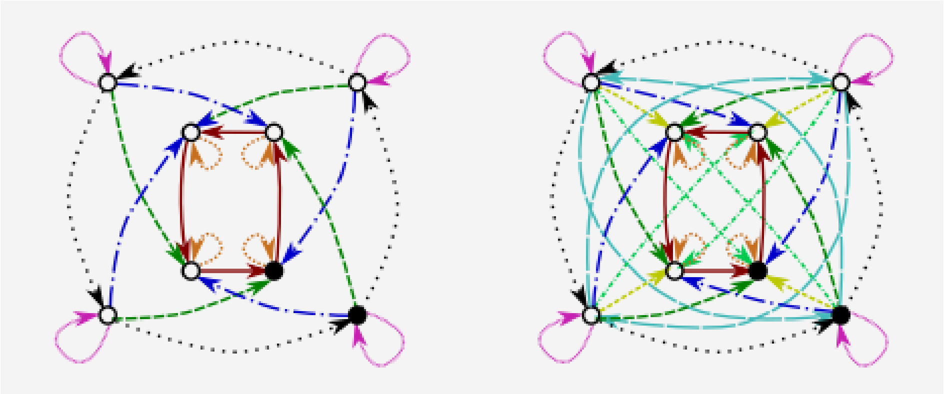


$N = M = 4$

*source : Ravanbakhsh et al. “Equivariance Through Parameter-Sharing”, ICML 2017

- Now, suppose we have a group G
- Can we design a colored bipartite graph Ω, that is G-equivariant?
  - **Yes**, if G is of the form $\{(k_{\mathbb{N}}, k_{\mathbb{M}}) | \ k \in K\}$ for a finite group K

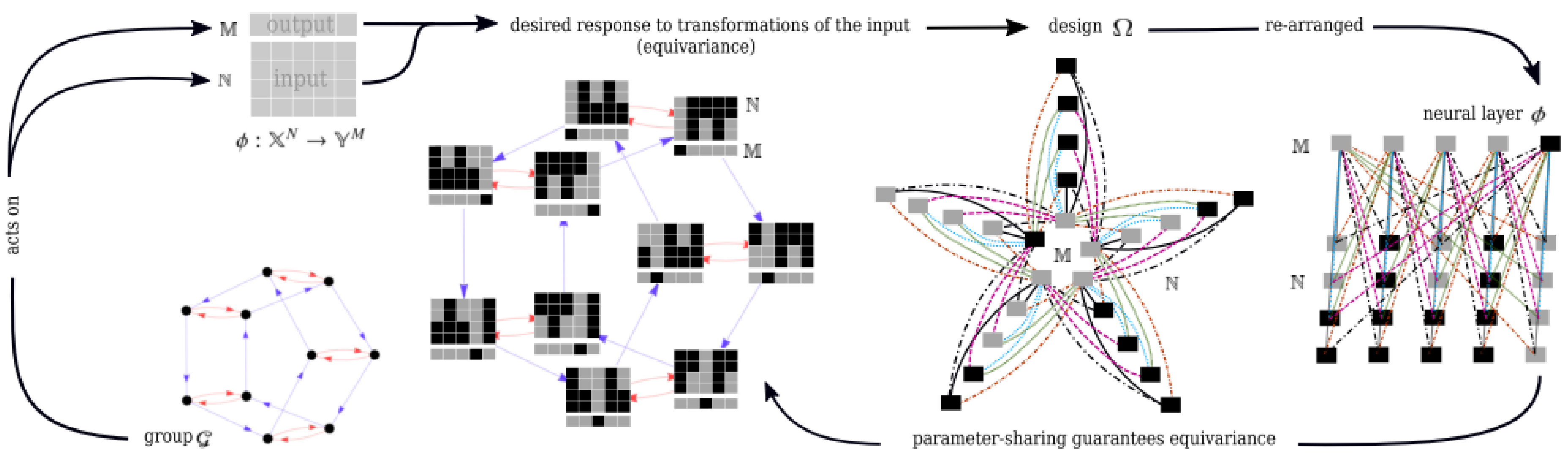
- **Example:** Equivariance to $90°$ rotations
  - $\pm 90°$ rotations is produced as the action of cyclic group $\mathbb{Z}_4 = \{e, g, g^2, g^3\}$
  - Letting $\mathbb{N} = \mathbb{M} = [1, \dots, 8]$, possible Ω's are presented below



*source : Ravanbakhsh et al. "Equivariance Through Parameter-Sharing", ICML 2017

Equivariance through parameter-sharing [Ravanbakhsh et al., 2017]

- Now, suppose we have a group G
- Can we design a colored bipartite graph Ω, that is G-equivariant?
  - **Yes**, if G is of the form $\{(k_{\mathbb{N}}, k_{\mathbb{M}}) |\ k \in K\}$ for a finite group K

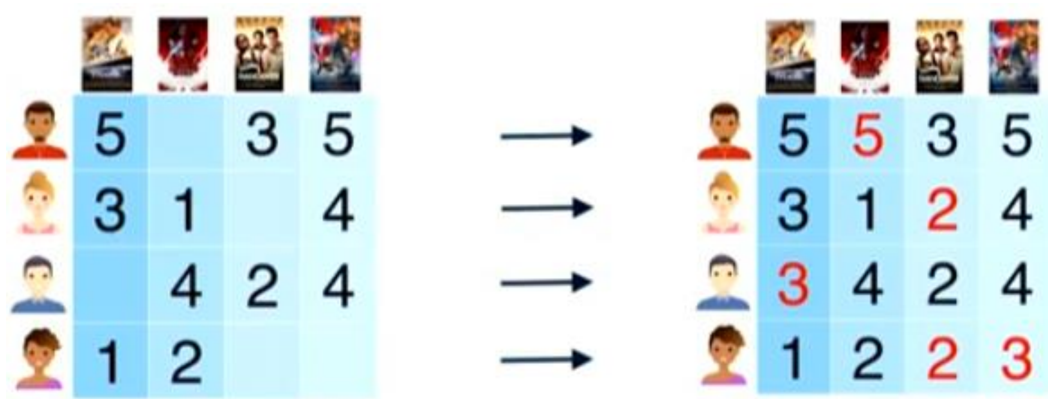


Next, application to movie recommendation

*source : Ravanbakhsh et al. "Equivariance Through Parameter-Sharing", ICML 2017

- **Example:** Movie recommendation
  - Predicting movie ratings from the previous incomplete matrix of ratings
  - Well-known as the **matrix completion**

- How can be build a deep model for this problem?
- **Idea:** The prediction must be equivariant on permuting rows & columns
  - “Exchangeable matrix”



*source : Hartford et al. “Deep Models of Interactions Across Sets”, ICML 2018

- **Example:** Movie recommendation

- **Exchangeable matrix layer**
  - We have an input matrix, which can be in a *flatten* form

5		3	5
3	1		4
	4	2	4
1	2		

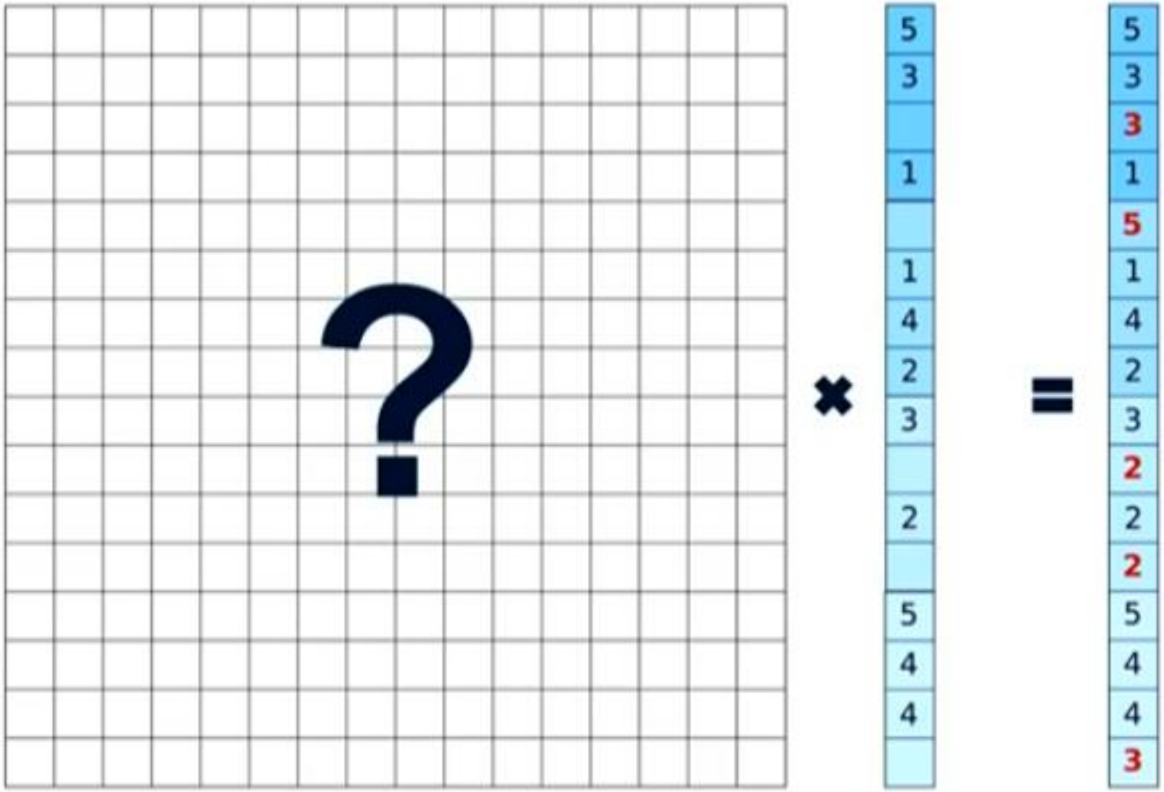
*source : Hartford et al. “Deep Models of Interactions Across Sets”, ICML 2018

Equivariance through parameter-sharing: Application [Hartford et al., 2018]

Original Input

- **Example:** Movie recommendation

- **Exchangeable matrix layer**
  - **Goal**: Design a matrix layer that is equivariant on permuting the original input

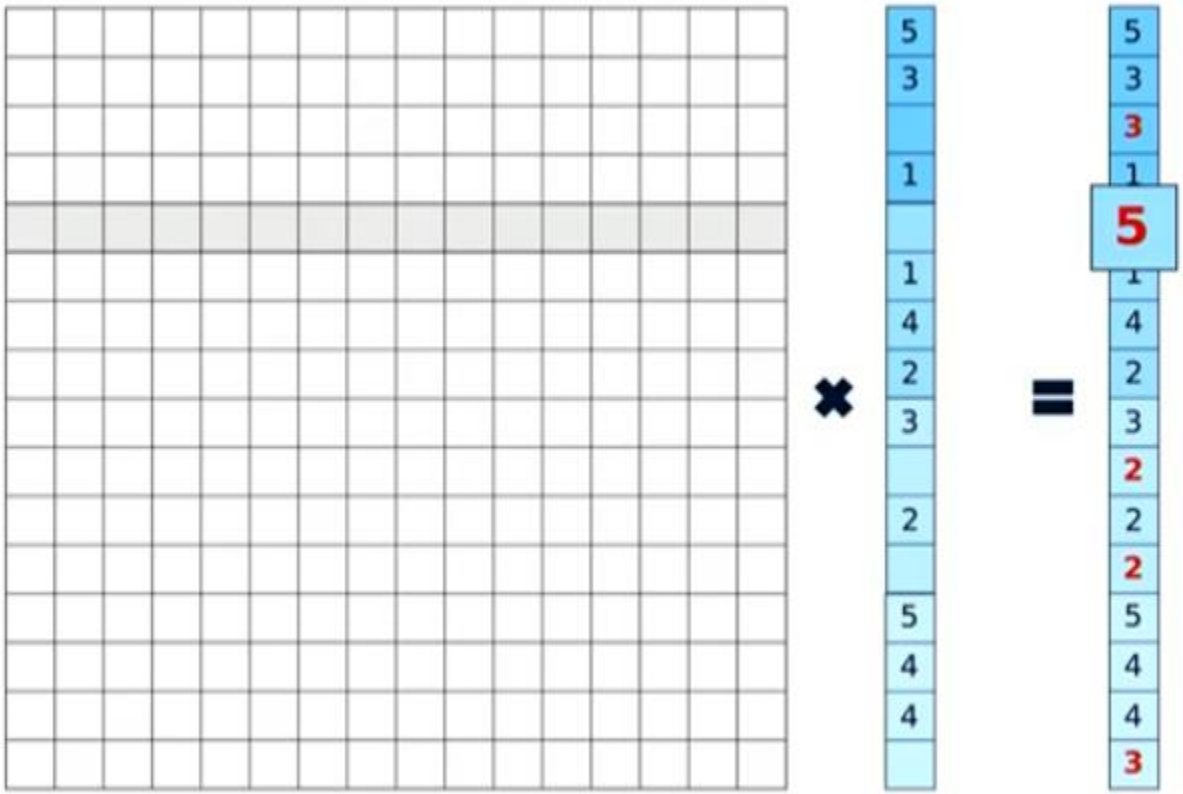


*source : Hartford et al. "Deep Models of Interactions Across Sets", ICML 2018

Equivariance through parameter-sharing: Application [Hartford et al., 2018]

- **Example:** Movie recommendation

- **Exchangeable matrix layer**
  - Suppose we calculate a single entry in the output (i.e. dot product)



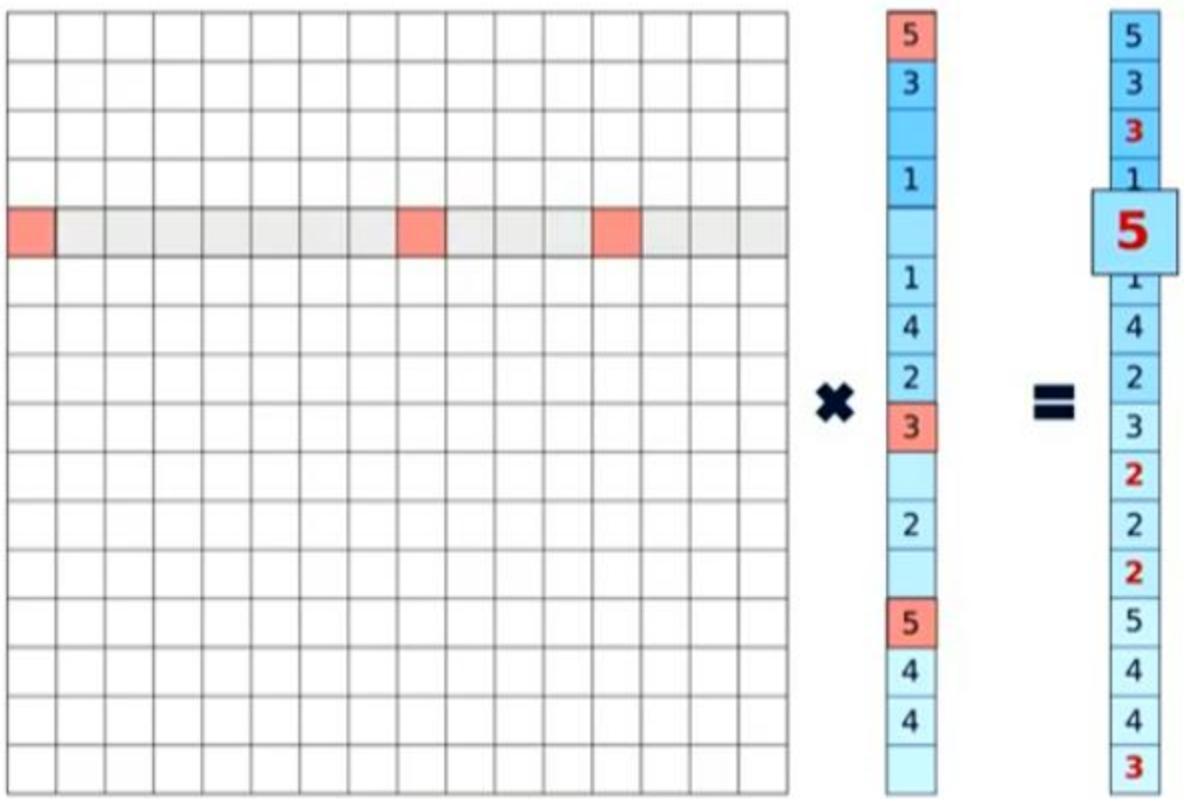
*source : Hartford et al. "Deep Models of Interactions Across Sets", ICML 2018

Equivariance through parameter-sharing: Application [Hartford et al., 2018]

Original Input

- **Example:** Movie recommendation

- **Exchangeable matrix layer**
  - The ratings for other movies affects to the prediction, regardless to its order

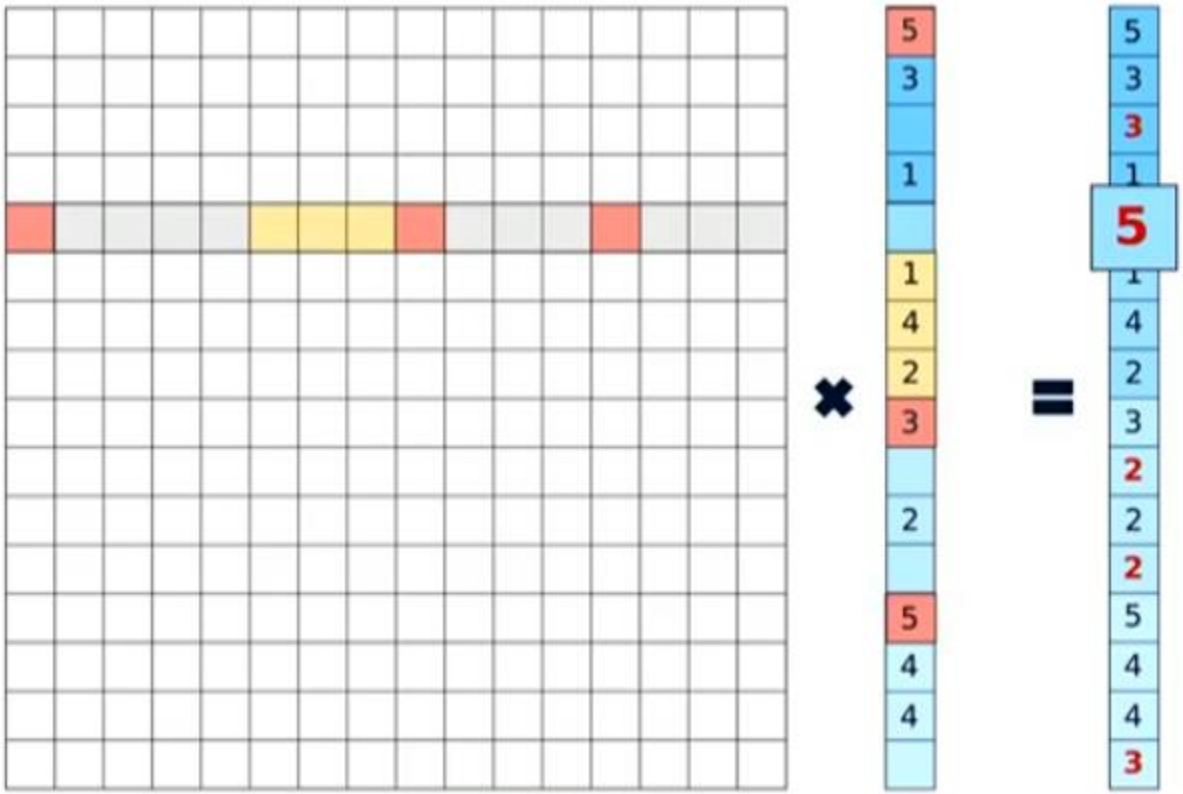


same color = parameter sharing

Equivariance through parameter-sharing: Application [Hartford et al., 2018]

- **Example:** Movie recommendation

- **Exchangeable matrix layer**
  - Ratings of other users for a movie also affects, regardless to its order as well

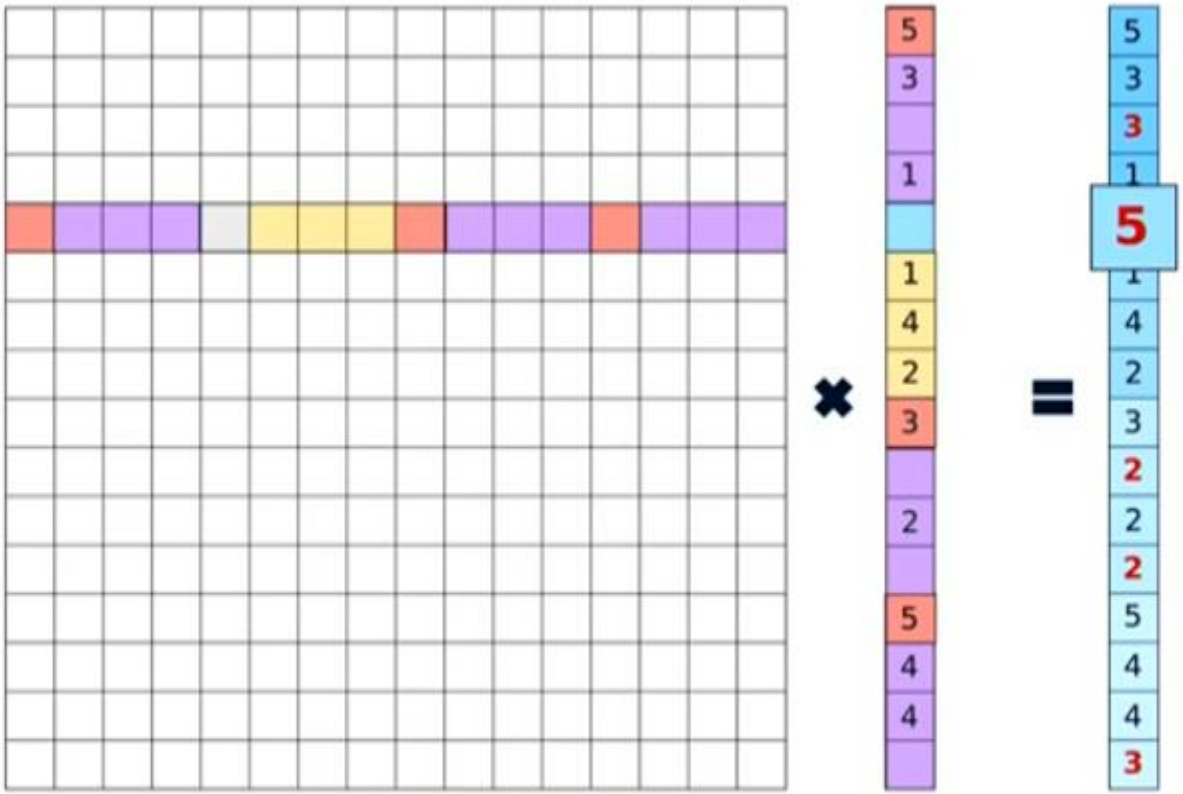


same color = parameter sharing

*source : Hartford et al. “Deep Models of Interactions Across Sets”, ICML 2018

- **Example:** Movie recommendation

- **Exchangeable matrix layer**
  - The same argument holds for the other parameters except one



same color = parameter sharing

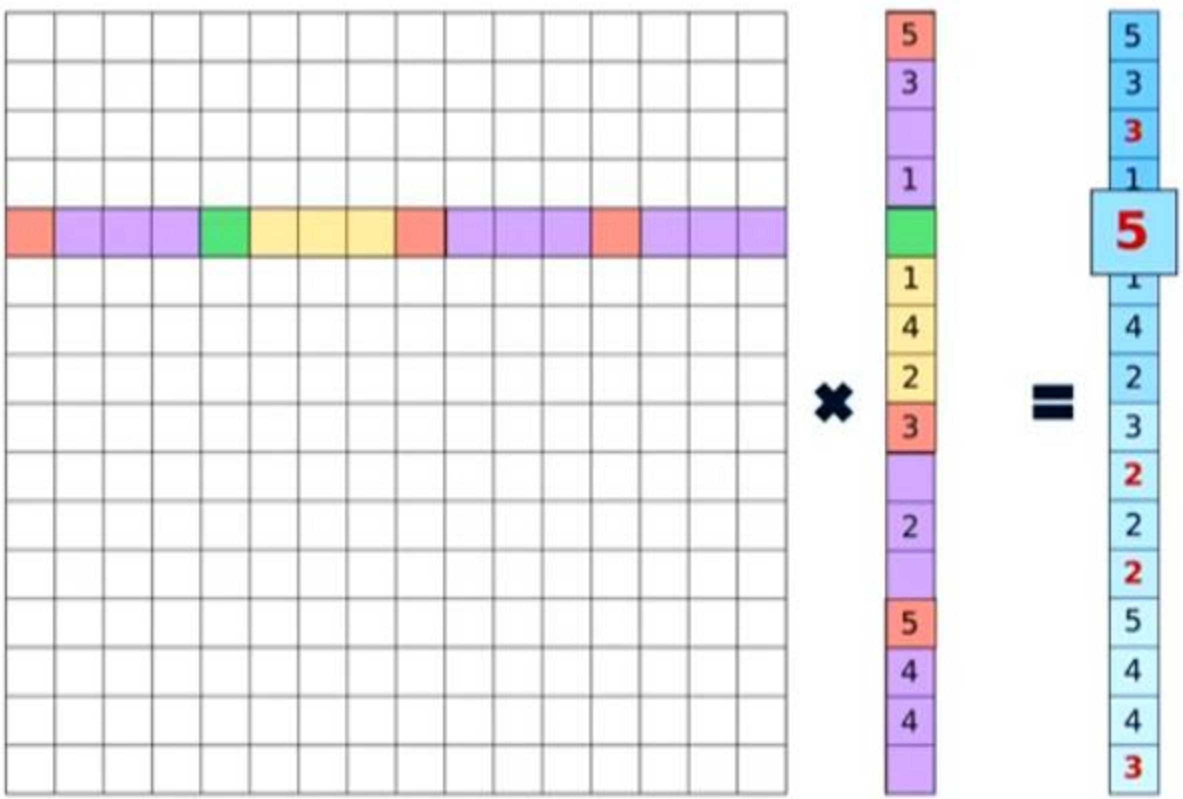
*source : Hartford et al. "Deep Models of Interactions Across Sets", ICML 2018

Equivariance through parameter-sharing: Application [Hartford et al., 2018]

- **Example:** Movie recommendation

Original Input

- **Exchangeable matrix layer**
  - The same argument holds for the other parameters except one

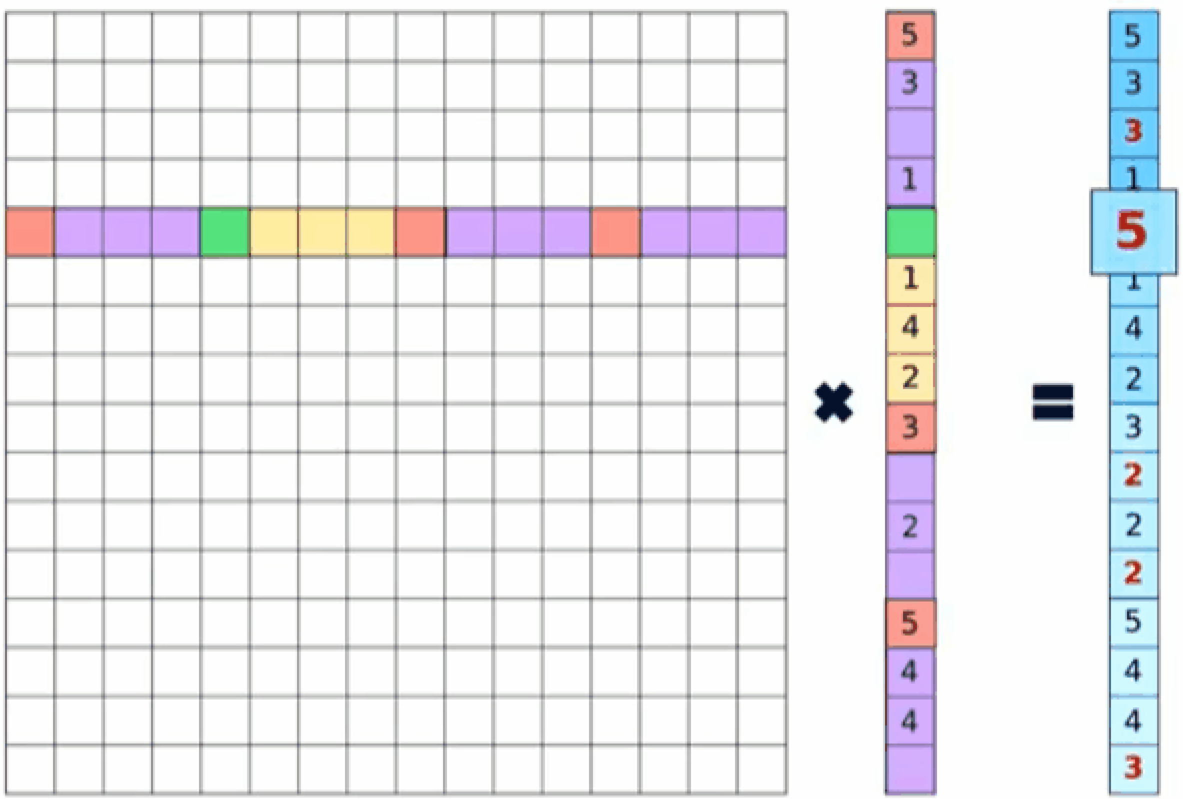


same color = parameter sharing

*source : Hartford et al. “Deep Models of Interactions Across Sets”, ICML 2018

- **Example:** Movie recommendation

- **Exchangeable matrix layer**
  - Exchangeability also holds for column-wise permutations



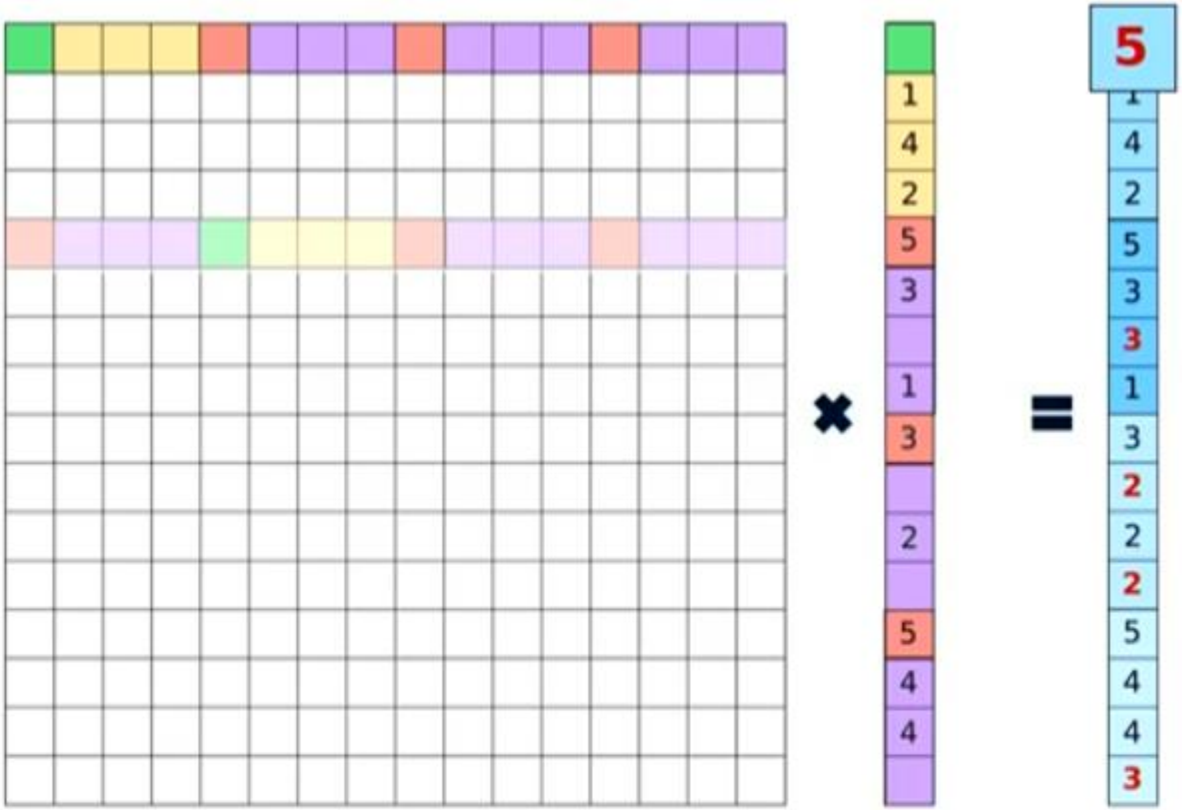
same color = parameter sharing

*source : Hartford et al. “Deep Models of Interactions Across Sets”, ICML 2018

- **Example:** Movie recommendation

Original Input

- **Exchangeable matrix layer**
  - Exchangeability also holds for column-wise permutations



same color = parameter sharing

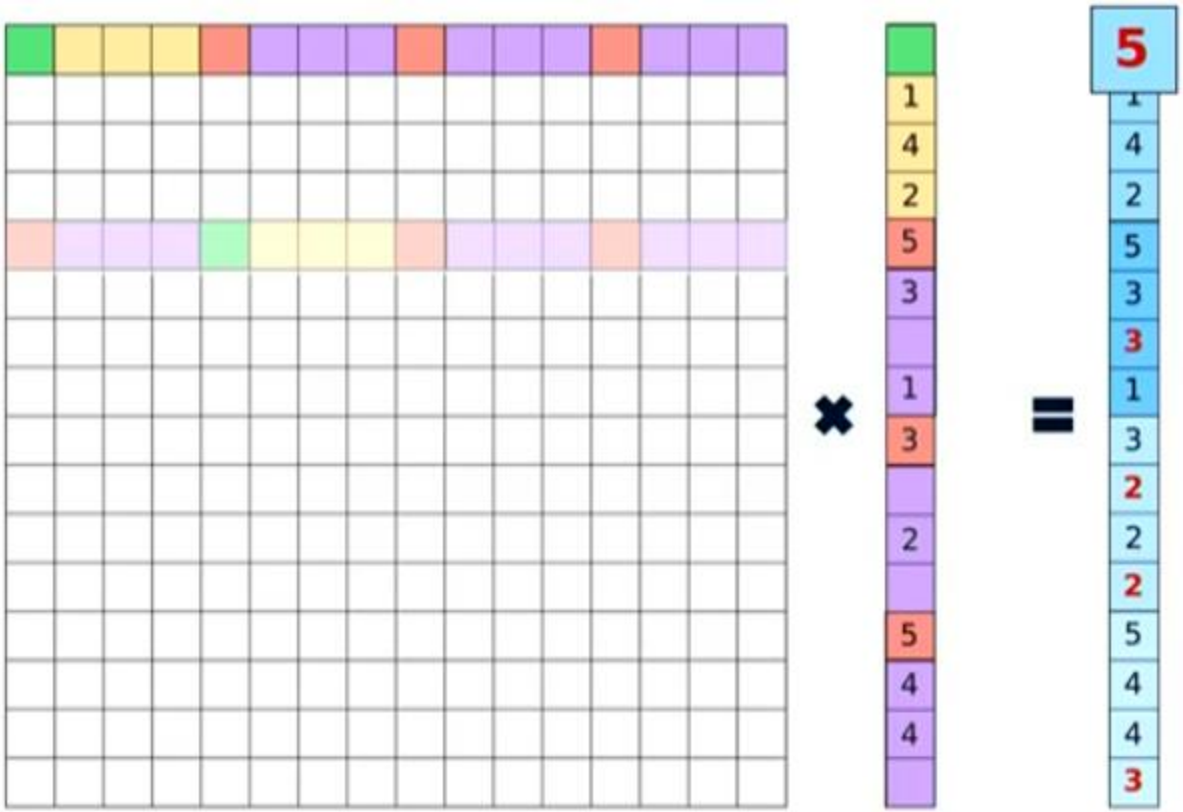
*source : Hartford et al. "Deep Models of Interactions Across Sets", ICML 2018

Equivariance through parameter-sharing: Application [Hartford et al., 2018]

Original Input

- **Example:** Movie recommendation
- **Exchangeable matrix layer**

  Quiz: How should we color the remaining part of the matrix?



same color = parameter sharing

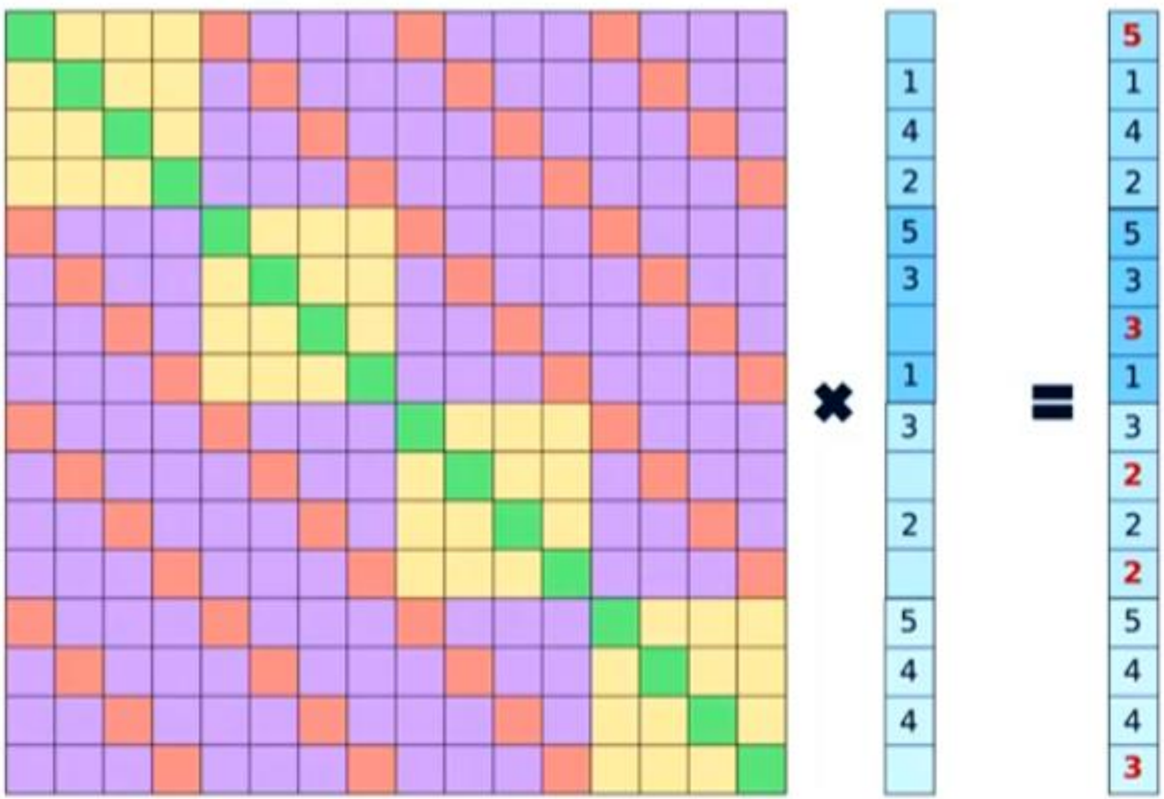
*source : Hartford et al. “Deep Models of Interactions Across Sets”, ICML 2018

Equivariance through parameter-sharing: Application [Hartford et al., 2018]

- **Example:** Movie recommendation

- **Exchangeable matrix layer**

  Quiz: How should we color the remaining part of the matrix?

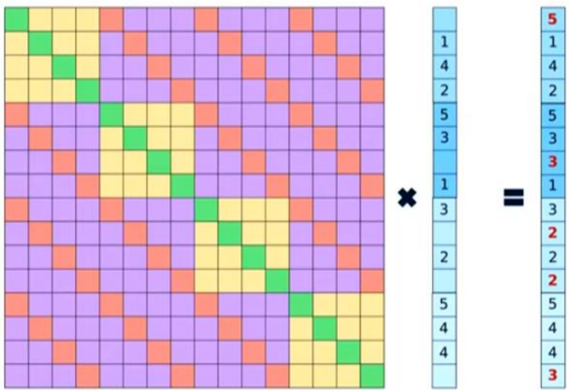


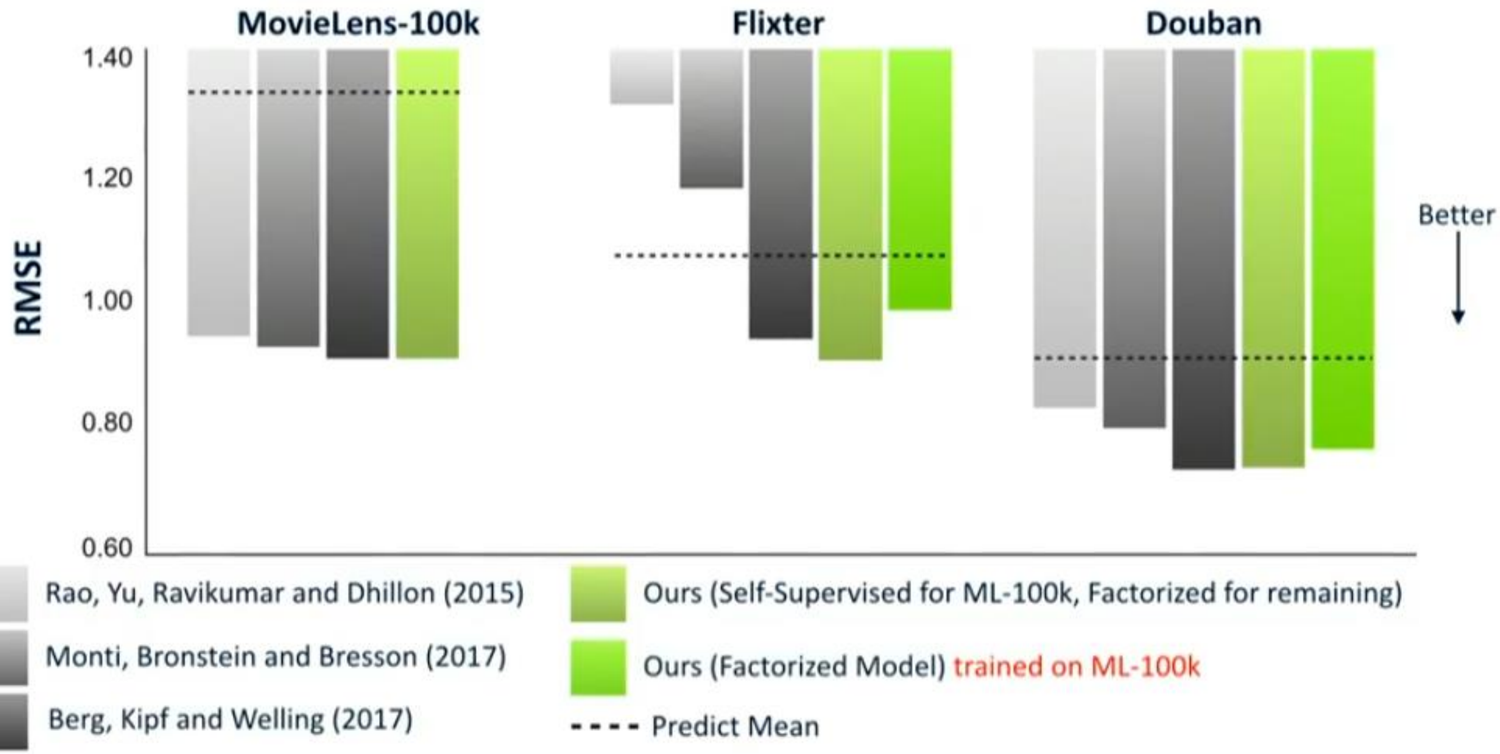
same color = parameter sharing

*source : Hartford et al. "Deep Models of Interactions Across Sets", ICML 2018

Equivariance through parameter-sharing: Application [Hartford et al., 2018]

- **Example:** Movie recommendation
  - Deep models constructed from this matrix outperforms many existing benchmarks
  - The model trained on MovieLens-100k surprisingly generalize well on other datasets



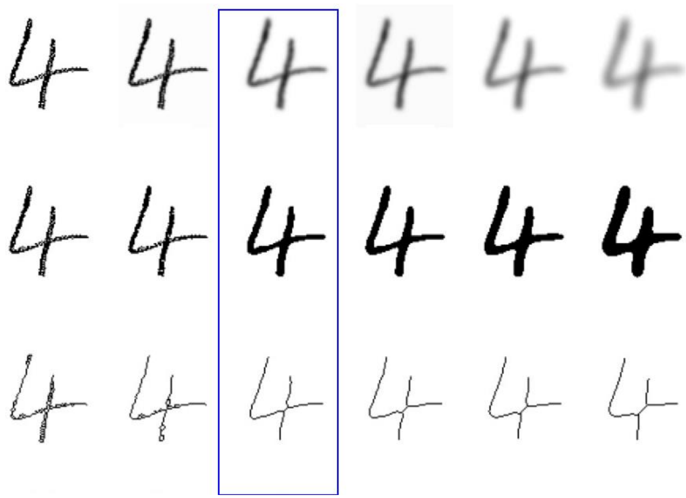


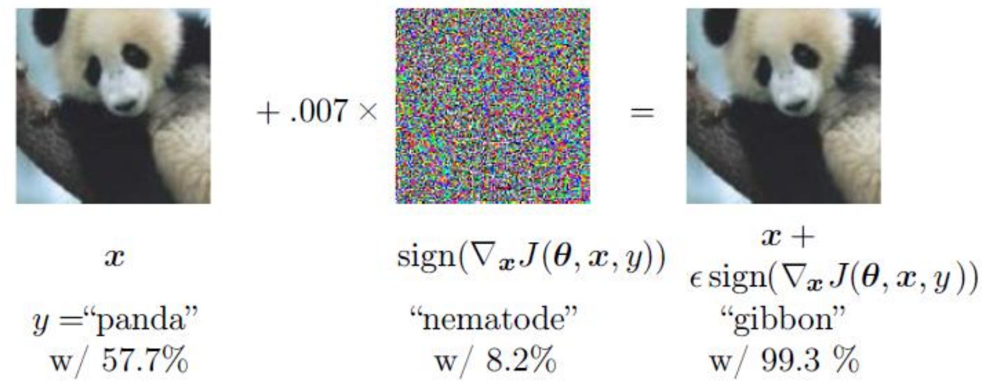
*source : Hartford et al. “Deep Models of Interactions Across Sets”, ICML 2018

Table of Contents

Noise robustness

- **Prior:** Most AI tasks have certain levels of resilience on noise
- One can incorporate such prior by injecting noises to the network





- Noise robustness is also related to **adversarial examples**
  - We will discuss this topic more in detail later

*sources :
- Chatbri, Houssem et al. "Using scale space filtering to make thinning algorithms robust against noise in sketch images." Pattern Recognition Letters 42 (2014): 1-10.
- https://www.deeplearningbook.org/contents/ml.html

Noise robustness

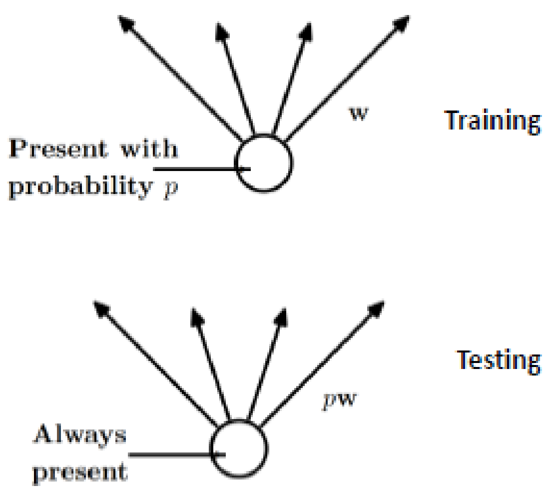
- **Prior:** Most AI tasks have certain levels of resilience on noise
- One can incorporate such prior by injecting noises to the network

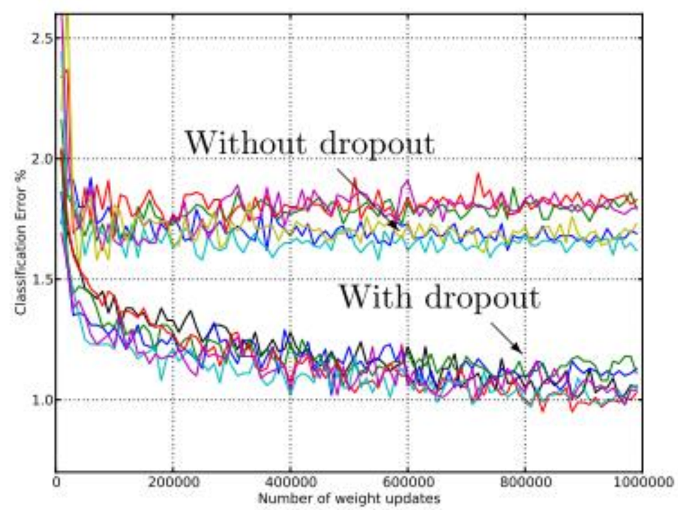
- There can be many ways to impose noises:
  1. On **inputs** or **hidden units** (e.g. *Dropout*)
     - Noise with infinitesimal variance at the input is equivalent to imposing a penalty on the norm of the weights for some models [Bishop, 1995a,b]
  2. On **model parameters** (e.g. *Variational dropout*)
     - A stochastic implementation of a Bayesian inference over the weights
  3. On **gradients** during optimization (e.g. *Shake-shake regularization*)
     - In practice, SGD can generalize better than full GD in training DNNs [Keskar et al., 2016]

Next, dropout

- **Dropout** [Srivastava et al., 2014] randomly drops a neuron with probability p during training
  - Same as **multiplying a noise** $\mu \sim \text{Bernulli}(p)$ to each neuron
- At testing, each weights are scaled by p

- Dropout is applied to **hidden units** typically
  - Destruction of high-level information e.g. edges, nose, …

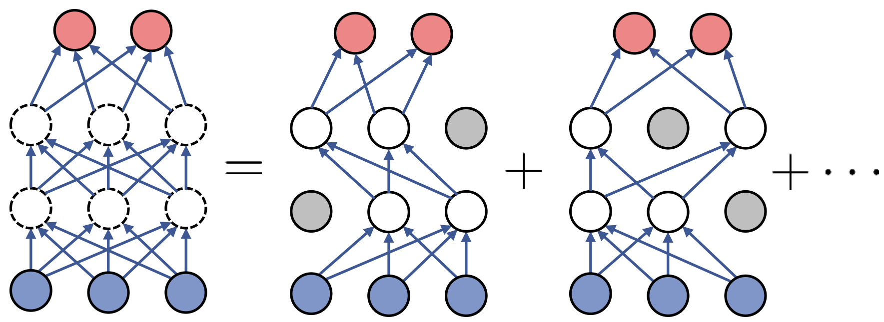




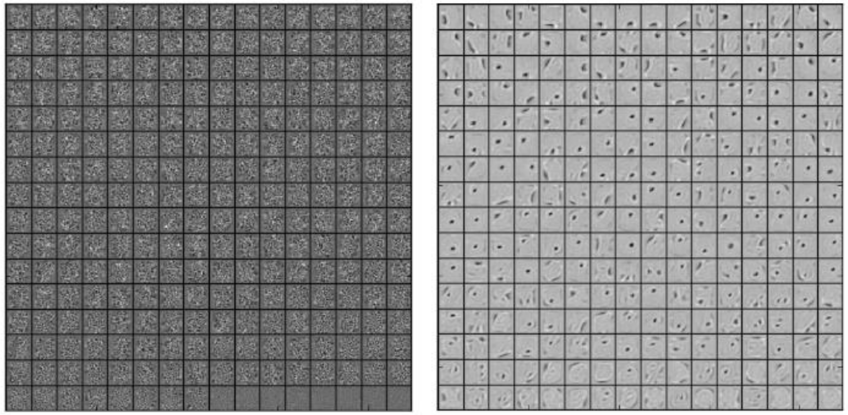
*source : Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". JMLR 2014

Why dropout generalizes well?

1. It can be thought of as ensemble of 2^n subnets with parameter sharing



2. Dropout prevents co-adaptation of neurons
   - Noisy neurons are **less reliable**
   - Each neuron must be prepared on which other neurons are dropped



(a) Without dropout (b) Dropout with $p = 0.5$.

*source : Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". JMLR 2014

The fully understanding on why dropout works is still an open question

- **Stochasticity might not be necessary**
  - **Fast dropout** [Wang et al., 2013]: A deterministic version of dropout with analytic marginalization
- **Dropout as an ensemble is not enough**
  - Dropout offers additional improvements to generalization error beyond those obtained by ensembles of independent models [Warde-Farley et al., 2013]
- **Dropping neurons are not necessary**
  - In principle, any kind of random modification is admissible
  - **Gaussian dropout**, i.e. $\mu \sim \mathcal{N}(1, \frac{1-p}{p})$, can work as well as the original dropout with probability p, or even work better

Next, variational dropout

*source : Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". JMLR 2014

Noises on model parameters: Variational dropout [Kingma et al., 2015]

- In dropout, one have to find the best p manually
  - What if we want different rates for each of neurons?

- **Variational dropout** (VD) allows to learn the dropout rates separately
- Unlike Dropout, VD imposes noises on **model parameters $\boldsymbol{\theta}$**:

$$w_i := \theta_i \cdot \xi_i, \qquad \text{where} \quad p_{\alpha_i}(\xi_i) = \mathcal{N}(1, \alpha_i)$$

  - A Bayesian generalization of **Gaussian dropout** [Srivastava et al., 2014]
  - The random vector $\mathbf{w} = (w_i)_i$ is adapted to data in Bayesian sense by updating $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$

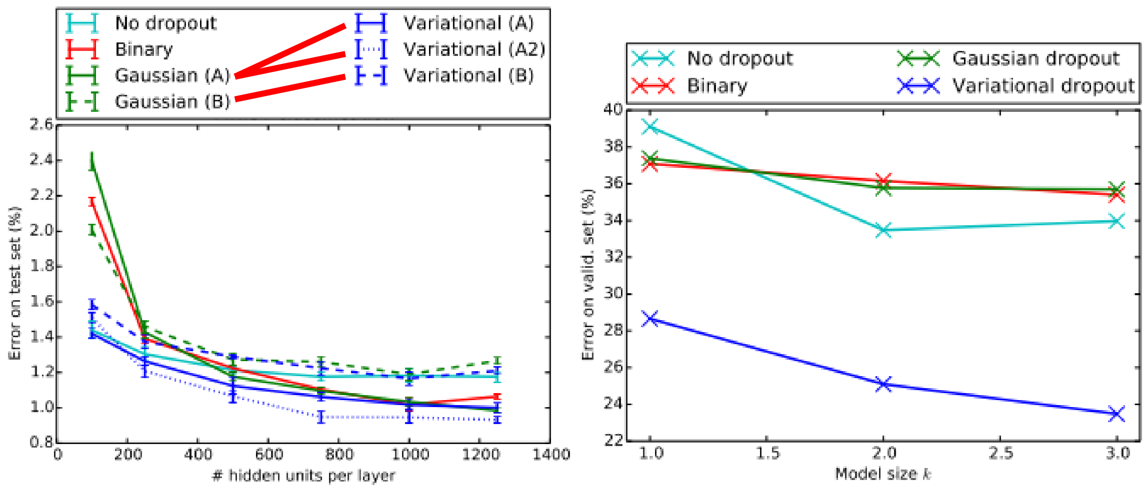
- **Re-parametrization trick** allows **w** to be learned via minibatch-based gradient estimation methods [Kingma et al., 2013]
  - $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ can be "optimized" separated from noises

$$w_i = \theta_i + (\theta_i \sqrt{\alpha_i}) \cdot \varepsilon_i, \qquad \text{where} \quad \varepsilon_i \sim \mathcal{N}(0, 1)$$

*source : Kingma et al. "Variational dropout and the local reparametrization trick". NIPS 2015

Noises on model parameters: Variational dropout [Kingma et al., 2015]

- VD lead to a better model than dropout
- VD could also improve CNN as well, while dropout could not(1b)



(a) Classification error on the MNIST dataset (b) Classification error on the CIFAR-10 dataset

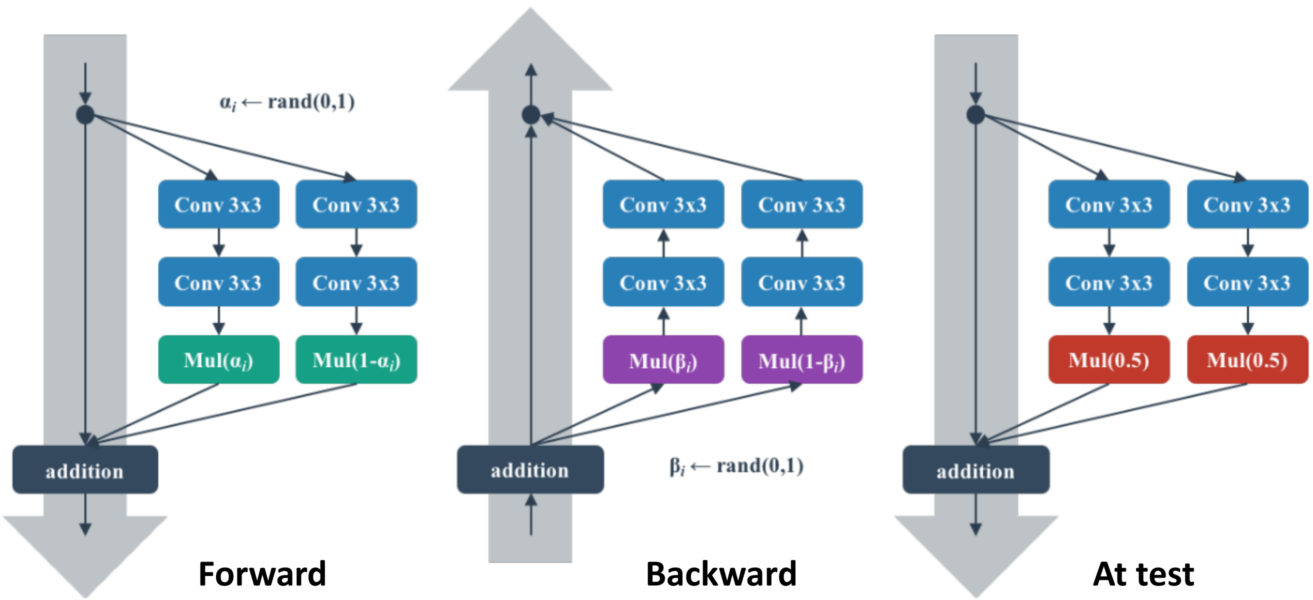
Figure 1: Best viewed in color. **(a)** Comparison of various dropout methods, when applied to fully-connected neural networks for classification on the MNIST dataset. Shown is the classification error of networks with 3 hidden layers, averaged over 5 runs. he variational versions of Gaussian dropout perform equal or better than their non-adaptive counterparts; the difference is especially large with smaller models, where regular dropout often results in severe underfitting. **(b)** Comparison of dropout methods when applied to convolutional net a trained on the CIFAR-10 dataset, for different settings of network size k. The network has two convolutional layers with each $32k$ and $64k$ feature maps, respectively, each with stride 2 and followed by a softplus nonlinearity. This is followed by two fully connected layers with each $128k$ hidden units.

Next, shake-shake regularization

*source : Kingma et al. “Variational dropout and the local reparametrization trick”. NIPS 2015

Noises on gradients: Shake-shake regularization [Gastaldi, 2017]

- Noises can be injected even in **gradients** during back-propagation
- **Shake-shake regularization** considers a 3-branch ResNeXt [Xie et al., 2017]



- Here, notice that α_i and β_i are independent random variables
  - α_i stochastically blends the outputs from two branches
  - β_i randomly re-distributes the returning gradient between two branches
- Those re-scaling are done in channel-wise

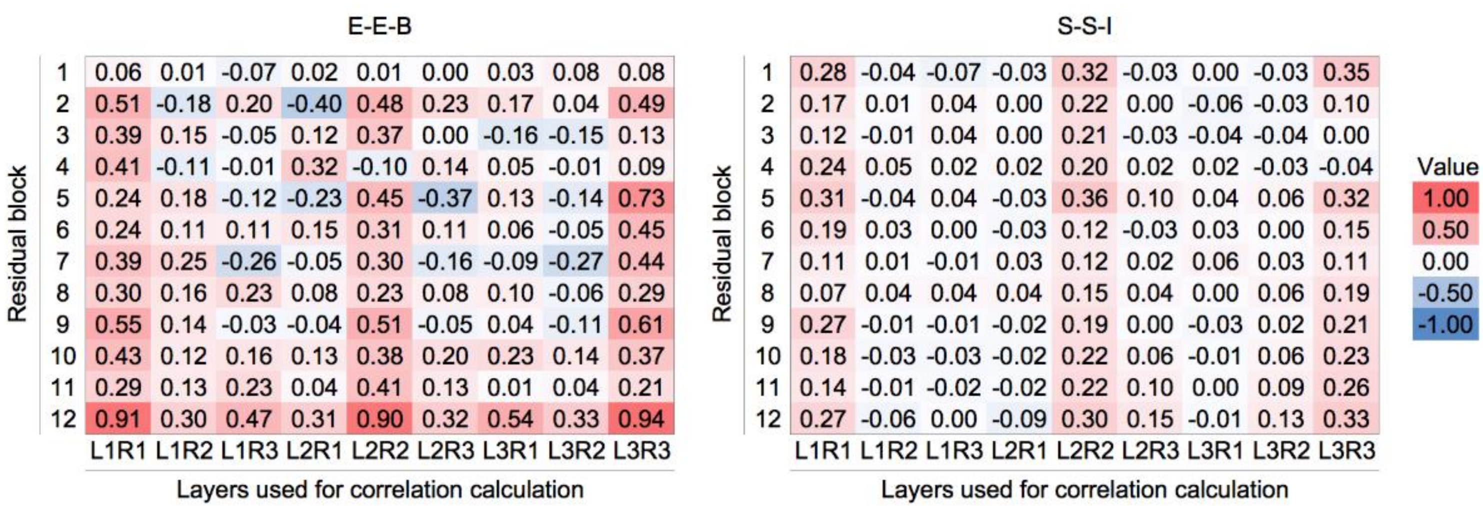
*source : Gastaldi, X. "Shake-Shake regularization". Arxiv 2017

Noises on gradients: Shake-shake regularization [Gastaldi, 2017]

- Shake-shake shows one of the current state-of-the-art result on CIFAR-10/100

Method	Depth	Params	C10	C100
Wide ResNet	28	36.5M	3.8	18.3
ResNeXt-29, 16x64d	29	68.1M	3.58	17.31
DenseNet-BC (k=40)	190	25.6M	3.46	17.18
C10 Model S-S-I	26	26.2M	**2.86**	-
C100 Model S-E-I	29	34.4M	-	**15.85**

- Shake-shake reduces layer-wise correlations between two branches



*source : Gastaldi, X. “Shake-Shake regularization”. Arxiv 2017

Table of Contents

- **Prior:** The best way to generalize better is to gain more data
- Create fake data and add it to the training set
  - Requires some knowledge on making good “fakes”
- Particularly effective for classification tasks
  - Some tasks may not be readily applicable, e.g. density estimation

- Example: Rigid transformation symmetries
  - Translation, dilation, rotation, mirror symmetry, …
  - Forms an affine group on pixels: $\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \mapsto \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$

Translation **Dilation** **Rotation** **Mirror symmetry**

Next, CutOut

*source : https://github.com/joanbruna/MathsDL-spring18/blob/master/lectures/lecture2.pdf

Making new data by local masking: CutOut [Devries et al., 2017]

- Dropout appears to be less powerful when used with convolutional layers
  - Dropping pixels randomly **may disturb gradients** due to parameter sharing
  - **Neighboring pixels in CNNs** would contains much of the dropped information

- Channel-wise dropout [Tompson et al., 2015] may alleviate these issues
  - However, the network capacity may be considerably reduced

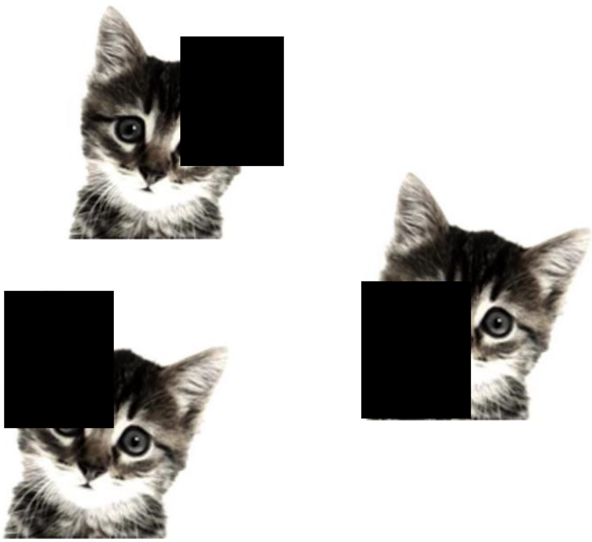
- What do we expect by performing dropout on images?
  - Preventing co-adaptation on high-level objects (nose, eyes, …)
  - For images, this can be also done by just using **local masking**

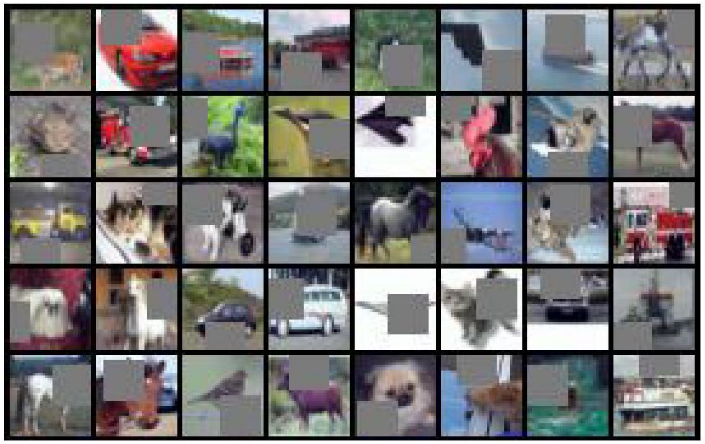
*source : Devries & Taylor. "Improved Regularization of Convolutional Neural Networks with Cutout", Arxiv 2017

Making new data by local masking: CutOut [Devries et al., 2017]

- What do we expect by performing dropout on images?
  - Preventing co-adaptation on high-level objects (nose, eyes, …)
  - For images, this can be also done by just using **local masking**

- **CutOut** directly brings this into data augmentation
  - Data augmentation via square-masking randomly on images





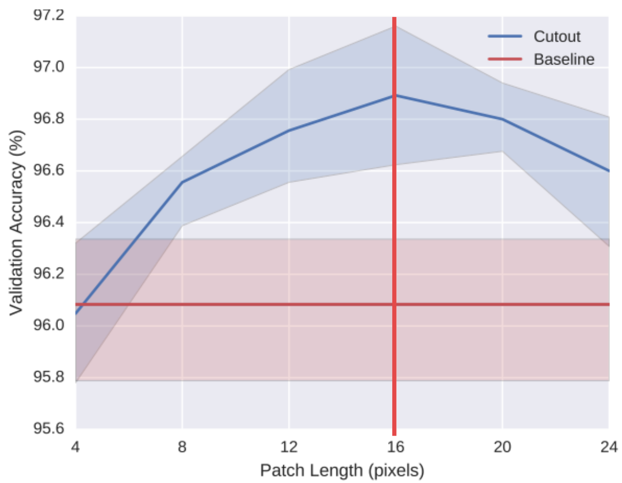
*source : Devries & Taylor. "Improved Regularization of Convolutional Neural Networks with Cutout", Arxiv 2017

Making new data by local masking: CutOut [Devries et al., 2017]

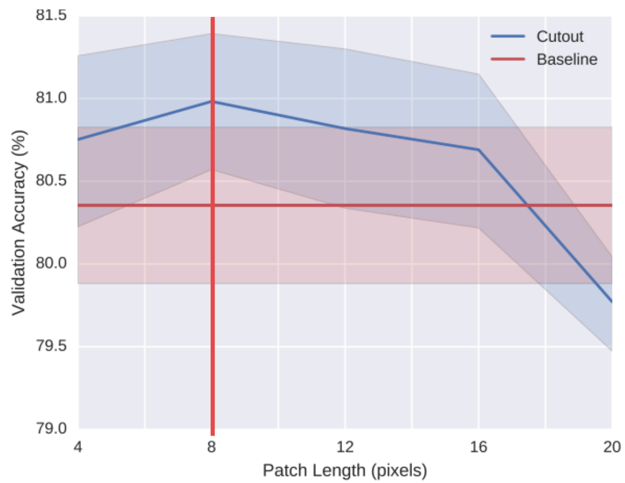
- CutOut further improved Shake-shake regularization [Gastaldi, 2017] achieving the state-of-the-art result on CIFAR-10/100

Method	C10	C10+	C100	C100+	SVHN
ResNet18 [5]	10.63 ± 0.26	4.72 ± 0.21	36.68 ± 0.57	22.46 ± 0.31	-
ResNet18 + cutout	9.31 ± 0.18	3.99 ± 0.13	34.98 ± 0.29	21.96 ± 0.24	-
WideResNet [22]	6.97 ± 0.22	3.87 ± 0.08	26.06 ± 0.22	18.8 ± 0.08	1.60 ± 0.05
WideResNet + cutout	$\mathbf{5.54 \pm 0.08}$	3.08 ± 0.16	$\mathbf{23.94 \pm 0.15}$	18.41 ± 0.27	$\mathbf{1.30 \pm 0.03}$
Shake-shake regularization [4]	-	2.86	-	15.85	-
Shake-shake regularization + cutout	-	$\mathbf{2.56 \pm 0.07}$	-	$\mathbf{15.20 \pm 0.21}$	-

- The size of the square should be set as a hyperparameter



(a) CIFAR-10



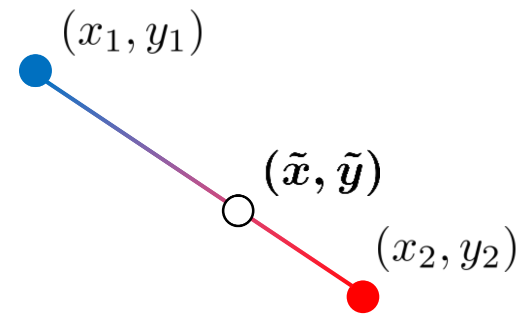
(b) CIFAR-100

Next, *mixup*

*source : Devries & Taylor. “Improved Regularization of Convolutional Neural Networks with Cutout”, Arxiv 2017

Mixing two samples in dataset: *mixup* [Zhang et al., 2018]

- In ***mixup***, a new training example is constructed by:



$$\tilde{x} = \lambda x_1 + (1 - \lambda) x_2$$
$$\tilde{y} = \lambda y_1 + (1 - \lambda) y_2$$

  - $\lambda \sim \mathrm{Beta}(\alpha, \alpha) \in [0, 1]$, where α: hyperparameter
  - (x_i, y_i)'s are uniformly sampled from the training data

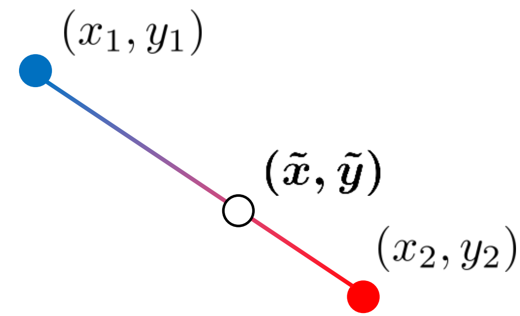
- Surprisingly, this simple scheme outperforms empirical risk minimization (ERM)
  - A new state-of-art performance on CIFAR-10/100 and ImageNet
  - Robustness when learning from corrupt labels
  - Handling adversarial examples
  - Stabilizing GANs
  - …

*source : Zhang, H. et al. "mixup: Beyond Empirical Risk Minimization", ICLR 2018

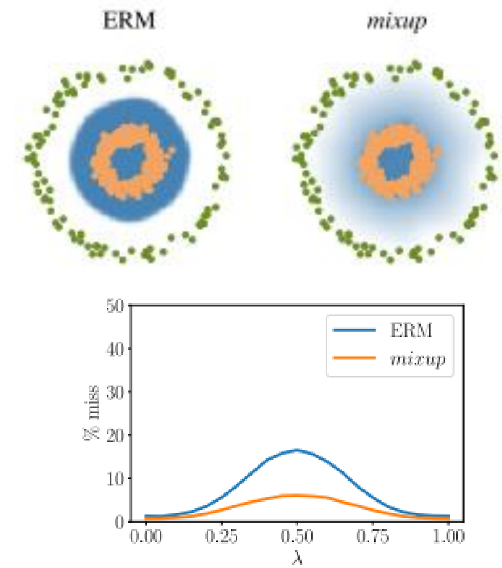
- In ***mixup***, a new training example is constructed by:

$$\tilde{x} = \lambda x_1 + (1 - \lambda) x_2$$
$$\tilde{y} = \lambda y_1 + (1 - \lambda) y_2$$

  - $\lambda \sim \mathrm{Beta}(\alpha, \alpha) \in [0, 1]$, where α: hyperparameter
  - (x_i, y_i)'s are uniformly sampled from the training data



- **What is *mixup* doing?**
  - Incorporating prior knowledge: the model should behave linearly in-between training examples
  - It reduces the amount of undesirable oscillations when predicting outside the training examples



(a) Prediction errors in-between training data. Evaluated at $x = \lambda x_i + (1 - \lambda) x_j$, a prediction is counted as a "miss" if it does not belong to $\{y_i, y_j\}$. The model trained with *mixup* has fewer misses.

*source : Zhang, H. et al. "mixup: Beyond Empirical Risk Minimization", ICLR 2018

Mixing two samples in dataset: *mixup* [Zhang et al., 2018]

- *mixup* significantly improves generalization in CIFIAR-10/100 and ImageNet

Dataset	Model	ERM	*mixup*
CIFAR-10	PreAct ResNet-18	5.6	**3.9**
	WideResNet-28-10	3.8	**2.7**
	DenseNet-BC-190	3.7	**2.7**
CIFAR-100	PreAct ResNet-18	25.6	**21.1**
	WideResNet-28-10	19.4	**17.5**
	DenseNet-BC-190	19.0	**16.8**

(a) Test errors for the CIFAR experiments.

Model	Method	Epochs	Top-1 Error	Top-5 Error
ResNet-50	ERM	200	23.6	7.0
	mixup $\alpha = 0.2$	200	**22.1**	**6.1**
ResNet-101	ERM	200	22.0	6.1
	mixup $\alpha = 0.2$	200	**20.8**	**5.4**
ResNeXt-101 32*4d	ERM	200	21.3	5.9
	mixup $\alpha = 0.4$	200	**20.1**	**5.0**

Table 1: Validation errors for ERM and *mixup* on the development set of ImageNet-2012.

- *mixup* also shows robustness on corrupted labels while improving memorization [Zhang et al., 2016]

Label corruption	Method	Test error		Training error	
		Best	Last	Real	Corrupted
20%	ERM	12.7	16.6	0.05	0.28
	ERM + dropout ($p = 0.7$)	8.8	10.4	5.26	83.55
	mixup ($\alpha = 8$)	**5.9**	**6.4**	2.27	86.32
50%	ERM	18.8	44.6	0.26	0.64
	ERM + dropout ($p = 0.8$)	14.1	15.5	12.71	86.98
	mixup ($\alpha = 32$)	**11.3**	**12.7**	5.84	85.71
80%	ERM	36.5	73.9	0.62	0.83
	ERM + dropout ($p = 0.8$)	30.9	35.1	29.84	86.37
	mixup ($\alpha = 32$)	**25.3**	**30.9**	18.92	85.44

*source : Zhang, H. et al. "mixup: Beyond Empirical Risk Minimization", ICLR 2018

Summary

- Reducing the test error, possibly at the expense of increased training error

- No free lunch theorem says that there is **no best form of regularization**
- We have to express our **prior knowledge** for each problem to guide the networks properly that generalizes well
- Developing effective regularizations is one of the major research in the field

- Nevertheless, as we are focusing on AI tasks, there could be some general strategies for a wide range of our problems
  - Loss penalty
  - Parameter sharing
  - Noise robustness
  - Dataset augmentation
  - **… there can be many other ways!**

References

- [Bishop, 1995a] Bishop, C. (1995). Regularization and Complexity Control in Feed-forward Networks. In *Proceedings International Conference on Artificial Neural Networks* (pp. 141–148).
  link : https://www.microsoft.com/en-us/research/publication/regularization-and-complexity-control-in-feed-forward-networks/
- [Bishop, 1995b] Bishop, C. (1995). Training with Noise is Equivalent to Tikhonov Regularization. *Neural Computation*, *7*, 108–116.
  link : https://ieeexplore.ieee.org/document/6796505/
- [Wolpert et al., 1997] Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82.
  link : https://ieeexplore.ieee.org/document/585893/
- [Hinton et al., 2012] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.
  link : https://arxiv.org/abs/1207.0580
- [Kingma et al., 2013] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.
  link : https://arxiv.org/abs/1312.6114
- [Wang et al., 2013] Wang, S., & Manning, C. (2013). Fast dropout training. In *Proceedings of the 30th International Conference on Machine Learning* (Vol. 28, pp. 118–126). Atlanta, Georgia, USA: PMLR.
  link : http://proceedings.mlr.press/v28/wang13a.html
- [Warde-Farley et al., 2013] Warde-Farley, D., Goodfellow, I. J., Courville, A., & Bengio, Y. (2013). An empirical analysis of dropout in piecewise linear networks. *ArXiv Preprint ArXiv:1312.6197*.
  link : https://arxiv.org/abs/1312.6197

References

- [Dauphin et al., 2014] Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., & Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in Neural Information Processing Systems 27* (pp. 2933–2941).
  link : https://papers.nips.cc/paper/5486-identifying-and-attacking-the-saddle-point-problem-in-high-dimensional-non-convex-optimization
- [Goodfellow et al., 2014] Goodfellow, I. J., & Vinyals, O. (2014). Qualitatively characterizing neural network optimization problems. *CoRR*, *abs/1412.6544*.
  link : https://arxiv.org/abs/1412.6544
- [Shalev-Shwartz et al., 2014] Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding Machine Learning: From Theory to Algorithms. Cambridge: Cambridge University Press. doi:10.1017/CBO9781107298019
  link : https://www.cambridge.org/core/books/understanding-machine-learning/3059695661405D25673058E43C8BE2A6
- [Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.
  link : http://jmlr.org/papers/v15/srivastava14a.html
- [Tompson et al., 2015] Tompson, J., Goroshin, R., Jain, A., LeCun, Y., & Bregler, C. (2015). Efficient object localization using convolutional networks. In *Computer Vision and Pattern Recognition* (pp. 648–656).
  link : https://arxiv.org/abs/1411.4280
- [Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press, pp.221-265.
  link : https://www.deeplearningbook.org/
- [Kingma et al., 2015] Kingma, D. P., Salimans, T., & Welling, M. (2015). Variational dropout and the local reparameterization trick. In Advances in Neural Information Processing Systems (pp. 2575-2583).
  link : https://papers.nips.cc/paper/5666-variational-dropout-and-the-local-reparameterization-trick

References

- [Maddison et al. 2016] Maddison, C. J., Mnih, A., & Teh, Y. W. (2016). The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*.
  link : https://openreview.net/forum?id=S1jE5L5gl
- [Keskar et al., 2016] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*.
  link : https://openreview.net/forum?id=H1oyRlYgg
- [Zhang et al., 2016] Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *CoRR*, *abs/1611.03530*.
  link : https://arxiv.org/abs/1611.03530
- [Ravanbakhsh et al., 2017] Ravanbakhsh, S., Schneider, J., & Póczos, B. (2017). Equivariance Through Parameter-Sharing. In *Proceedings of the 34th International Conference on Machine Learning* (Vol. 70, pp. 2892–2901). International Convention Centre, Sydney, Australia: PMLR.
  link : http://proceedings.mlr.press/v70/ravanbakhsh17a.html
- [Devries et al., 2017] Devries, T., & Taylor, G. W. (2017). Improved Regularization of Convolutional Neural Networks with Cutout. *CoRR*, *abs/1708.04552*. Retrieved from http://arxiv.org/abs/1708.04552
  link : https://arxiv.org/abs/1708.04552
- [Gastaldi, 2017] Gastaldi, X. (2017). Shake-Shake regularization. *CoRR*, *abs/1705.07485*.
  link : http://arxiv.org/abs/1705.07485
- [Kawaguchi et al., 2017] Kawaguchi, K., Kaelbling, L. P., & Bengio, Y. (2017). Generalization in deep learning. arXiv preprint arXiv:1710.05468.
  link : https://arxiv.org/abs/1710.05468

References

- [Pereyra et al., 2017] Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł., & Hinton, G. (2017). Regularizing neural networks by penalizing confident output distributions. arXiv preprint arXiv:1701.06548.
  link : https://arxiv.org/abs/1701.06548
- [Xie et al., 2017] Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition* (pp. 5987–5995).
  link : http://openaccess.thecvf.com/content_cvpr_2017/papers/Xie_Aggregated_Residual_Transformations_CVPR_2017_paper.pdf
- [Arora et al., 2018] Arora, S., Cohen, N., & Hazan, E. (2018). On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization. In *Proceedings of the 35th International Conference on Machine Learning* (Vol. 80, pp. 244–253). Stockholmsmässan, Stockholm Sweden: PMLR.
  link : http://proceedings.mlr.press/v80/arora18a.html
- [Hartford et al., 2018] Hartford, J., Graham, D., Leyton-Brown, K., & Ravanbakhsh, S. (2018). Deep Models of Interactions Across Sets. In *Proceedings of the 35th International Conference on Machine Learning* (Vol. 80, pp. 1909–1918). Stockholmsmässan, Stockholm Sweden: PMLR.
  Link : http://proceedings.mlr.press/v80/hartford18a.html
- [Louizos et al., 2018] Louizos, C., Welling, M., & Kingma, D. P. (2018). Learning Sparse Neural Networks through L_0 Regularization. In *International Conference on Learning Representations*.
  link : https://openreview.net/forum?id=H1Y8hhg0b
- [Zhang et al., 2018] Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2018). *mixup*: Beyond Empirical Risk Minimization. In *International Conference on Learning Representations*.
  link : https://openreview.net/forum?id=r1Ddp1-Rb