Meta learning symmetries by reparametrization

Authors: Allan Zhou, Tom Knowles, Chelsea Finn

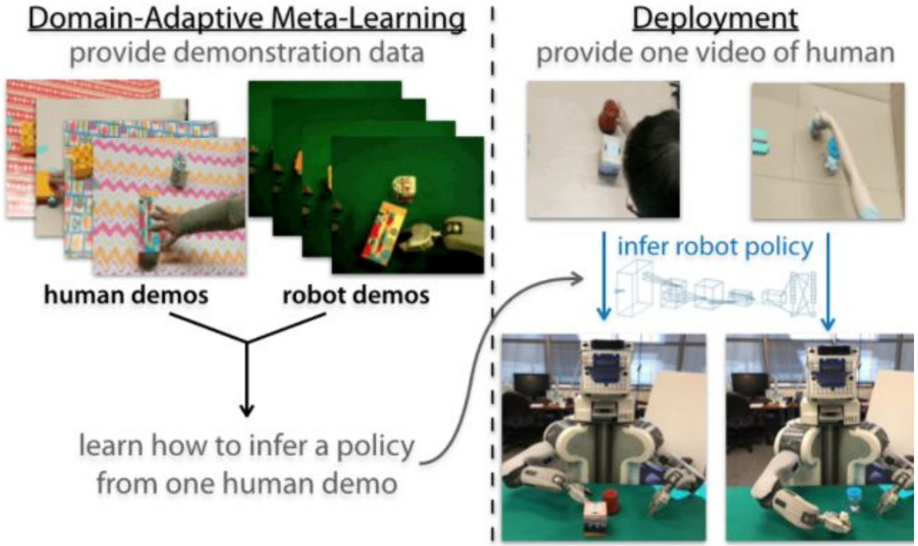
Presented by: Sakshi B.

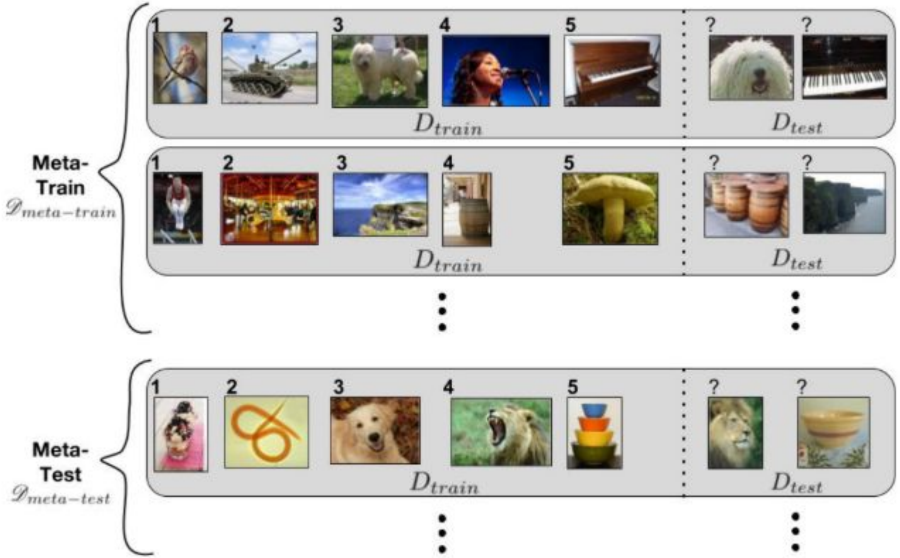
Motivation

- To build symmetry into NN, we need to know symmetry in advance.
- Example, CNNs are designed to be shift invariant.
- How to design NNs when symmetry is not as explicit?

This paper: Learn the symmetry structure from data, rather than hard-code it

Meta learning





$$arg \max_{\phi} logp(D|\phi) + logp(\phi)$$

$$arg \max_{\phi} \sum_{i} logp(y_i|x_i, \phi) + logp(\phi)$$

[Optimization as a model for few-shot learning]

- *Equivalence*
  - A function is equivariant to a transformation if *transforming the input to the function is equivalent to transforming the output*
  - Consider a network layer $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^m$
  - Assume we have two representations of a group G where $\pi_1(g)$ transforms the input vectors and $\pi_2(g)$ transforms the output vectors. The layer ϕ is G-equivariant w.r.t these transformations if:

$$\phi(\pi_1(g)\mathbf{v}) = \pi_2(g)\phi(\mathbf{v}) \qquad \text{for all} \quad g \in G, v \in \mathbb{R}^n$$

- *Invariance is a special case of equivariance*
  - If $\pi_2 = \mathrm{id} = I$ then we have:

$$\underbrace{\phi(\pi_1(g)\mathbf{v})}_{\text{Layer output on the transformed input}} = \pi_2(g)\phi(\mathbf{v}) = \underbrace{\phi(\mathbf{v})}_{\text{Layer output on the original input}}$$

Layer output on the transformed input **=** Layer output on the original input

This paper:

- Meta-learning can learn equivariant features from data
- Looses this equivariance information during gradient update

- Reparameterization trick: Factorize weight matrix W as product of symmetry matrix U and parameters vector v

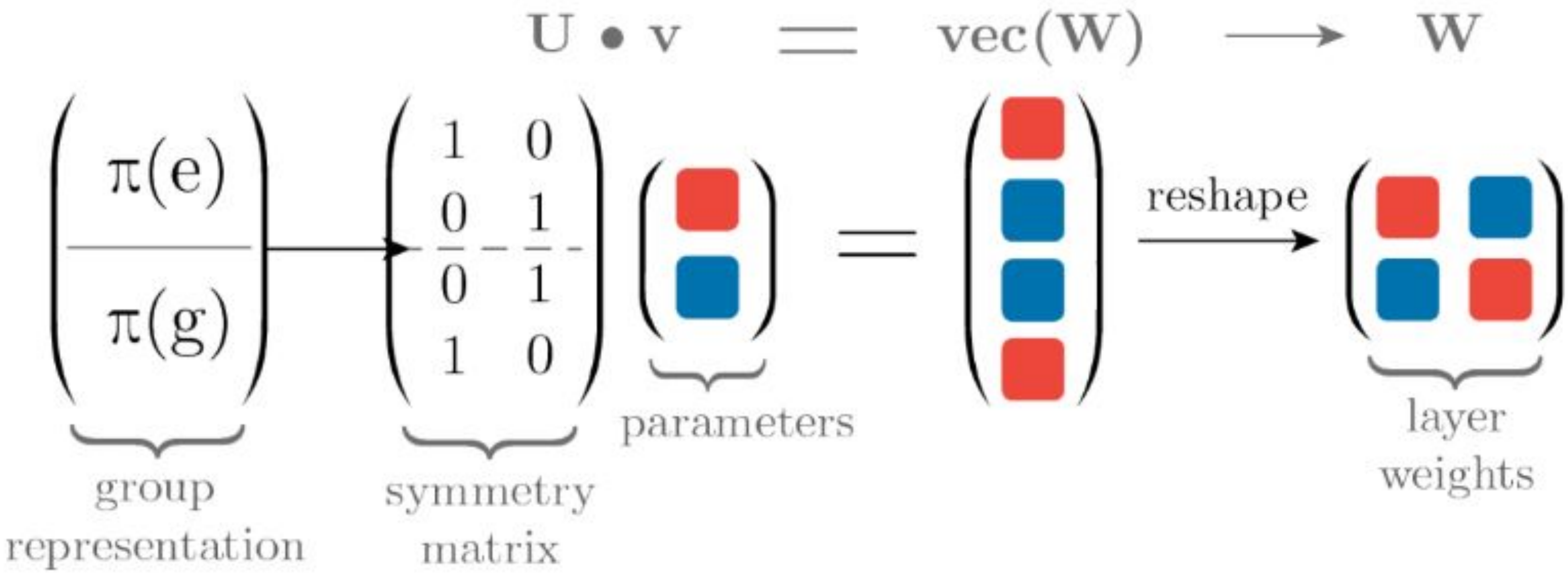
$$\mathrm{vec}(W) = Uv, \quad v \in \mathbb{R}^k, U \in \mathbb{R}^{mn \times k}$$

-

Encodes the pattern by which the weights W will “share” the filter parameters v

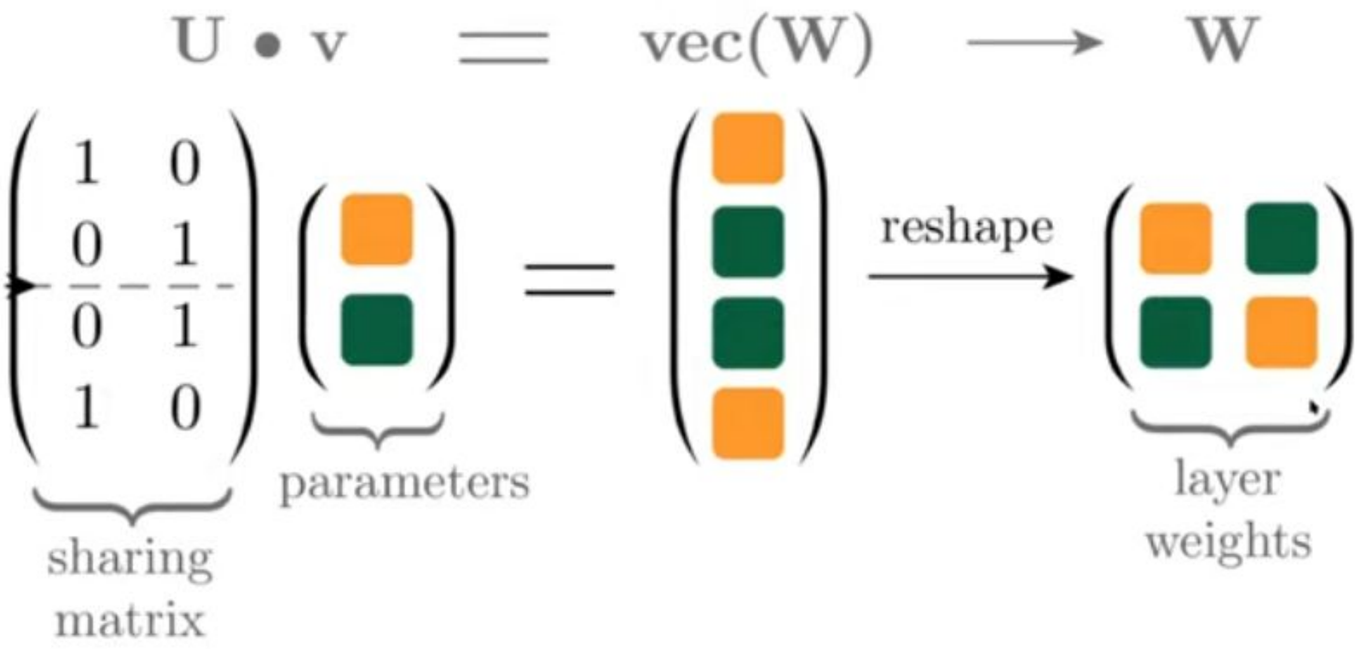
Shared filter parameters

Reparameterization:



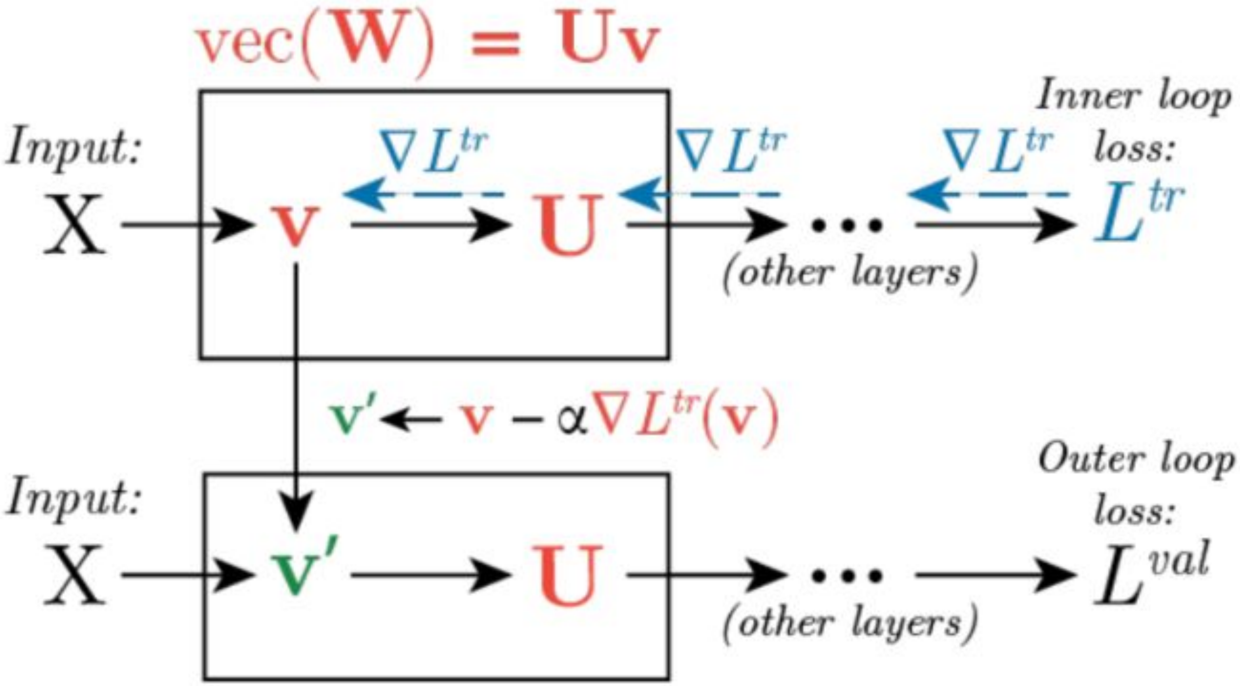
It is proved that this represents decoupled equivariant sharing patterns and filter parameters for all G-convolutions with finite group G

Example: Permutation equivariance



Meta-training algorithm

input: $\{\mathcal{T}_j\}_{j=1}^{N} \sim p(\mathcal{T})$: Meta-training tasks
input: $\{\mathbf{U}, \mathbf{v}\}$: Randomly initialized symmetry matrices and filters.
input: α, η: Inner and outer loop step sizes.
while *not done* **do**
  sample minibatch $\{\mathcal{T}_i\}_{i=1}^{n} \sim \{\mathcal{T}_j\}_{j=1}^{N}$;
  forall $\mathcal{T}_i \in \{\mathcal{T}_i\}_{i=1}^{n}$ **do**
    $\{\mathcal{D}_i^{tr}, \mathcal{D}_i^{val}\} \leftarrow \mathcal{T}_i$; `// task data`
    $\boldsymbol{\delta}_i \leftarrow \nabla_{\mathbf{v}} \mathcal{L}(\mathbf{U}, \mathbf{v}, \mathcal{D}_i^{tr})$;
    $\mathbf{v}' \leftarrow \mathbf{v} - \alpha \boldsymbol{\delta}_i$; `// inner step`
    `/* outer gradient */`
    $\mathbf{G}_i \leftarrow \frac{\mathrm{d}}{\mathrm{d}\mathbf{U}} \mathcal{L}(\mathbf{U}, \mathbf{v}', \mathcal{D}_i^{val})$;
  `/* outer step */`
  $\mathbf{U} \leftarrow \mathbf{U} - \eta \sum_i \mathbf{G_i}$;



- Since we aim to learn equivariances that are shared across tasks, the symmetry matrix doesn't update with the task.
- Only filter parameters update with task training data.
- Symmetry matrix updates using task's validation data.

(Meta-training)

Meta-train:

Inner loop (train):

$$\mathbf{v}' \leftarrow \mathbf{v} - \alpha \nabla_{\mathbf{v}} \mathcal{L}(\mathbf{U}, \mathbf{v}, \mathcal{D}_i^{\text{train}})$$

Outer loop (val):

$$\mathbf{U} \leftarrow \mathbf{U} - \eta \nabla_{\mathbf{U}} \mathcal{L}(\mathbf{U}, \mathbf{v}', \mathcal{D}_i^{\text{val}})$$

Meta-test: U is frozen and filter parameters v are updated

Experiments: Translation equivariance

Synthetic Problems MSE (lower is better)						
	Small train dataset			Large train dataset		
Method	$k=1$	$k=2$	$k=5$	$k=1$	$k=2$	$k=5$
MAML-FC	$3.4 \pm .60$	$2.1 \pm .35$	$1.0 \pm .10$	$3.4 \pm .49$	$2.0 \pm .27$	$1.1 \pm .11$
MAML-LC	$2.9 \pm .53$	$1.8 \pm .24$	$.87 \pm .08$	$2.9 \pm .42$	$1.6 \pm .23$	$.89 \pm .08$
MAML-Conv	$\mathbf{.00 \pm .00}$	$.43 \pm .09$	$.41 \pm .04$	$\mathbf{.00 \pm .00}$	$.53 \pm .08$	$.49 \pm .04$
MTSR-FC (Ours)	$3.2 \pm .49$	$1.4 \pm .17$	$.86 \pm .06$	$.12 \pm .03$	$\mathbf{.07 \pm .02}$	$\mathbf{.07 \pm .01}$
MSR-Joint-FC (Ours)	$.25 \pm .16$	$\mathbf{.12 \pm .04}$	$.21 \pm .03$	$.01 \pm .00$	$.08 \pm .02$	$.12 \pm .02$
MSR-FC (Ours)	$.07 \pm .02$	$\mathbf{.07 \pm .02}$	$\mathbf{.16 \pm .02}$	$\mathbf{.00 \pm .00}$	$\mathbf{.05 \pm .01}$	$.09 \pm .01$

- Baseline: Gradient based meta-learning (MAML)
- k=1 denotes translation equivariance, k=2,5 is less symmetry
- This algorithms is closest to MAML-Conv among all Non-Conv methods on k=1
- Outperforms all methods on k=2,5

Experiments: Rotation and flip equivariance

Rotation/Flip Equivariance MSE		
Method	Rot	Rot+Flip
MAML-Conv	.504	.507
MSR-Conv (Ours)	**.004**	**.001**

- Task data generated by passing randomly generated inputs through single-layer E(2) steerable CNN, equivariant to combinations of translations, discrete rotations, and reflections.
- Outperforms MAML significantly.

Contributions:

- Few-shot generalization on task distributions with shared underlying symmetries.
- Reduced number of task-specific parameters.

Drawbacks:

- All experimentation on synthetic data.
- Generalization in real-world missing.
- Works for discovering symmetries shared across a distribution of tasks. How about symmetries particular to a single task?

Thank you !