THE UNIVERSITY
OF BRITISH COLUMBIA

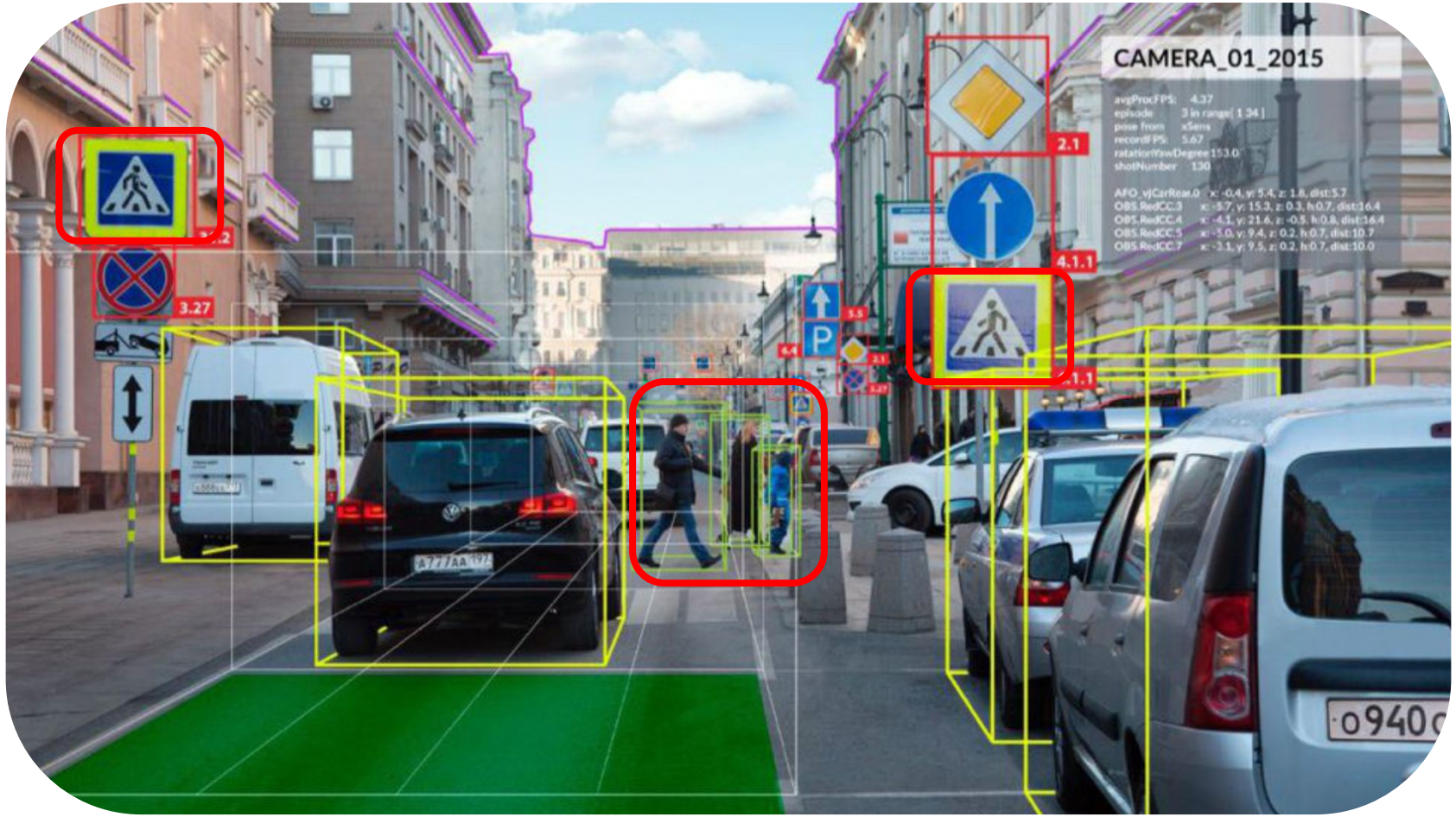
Group Equivariant Convolutional Networks

Armin Saghafian, Karen Shen, Masih Beigi Rizi

Feb 9 2026

Benefits of Invariance: Motivating Example 1

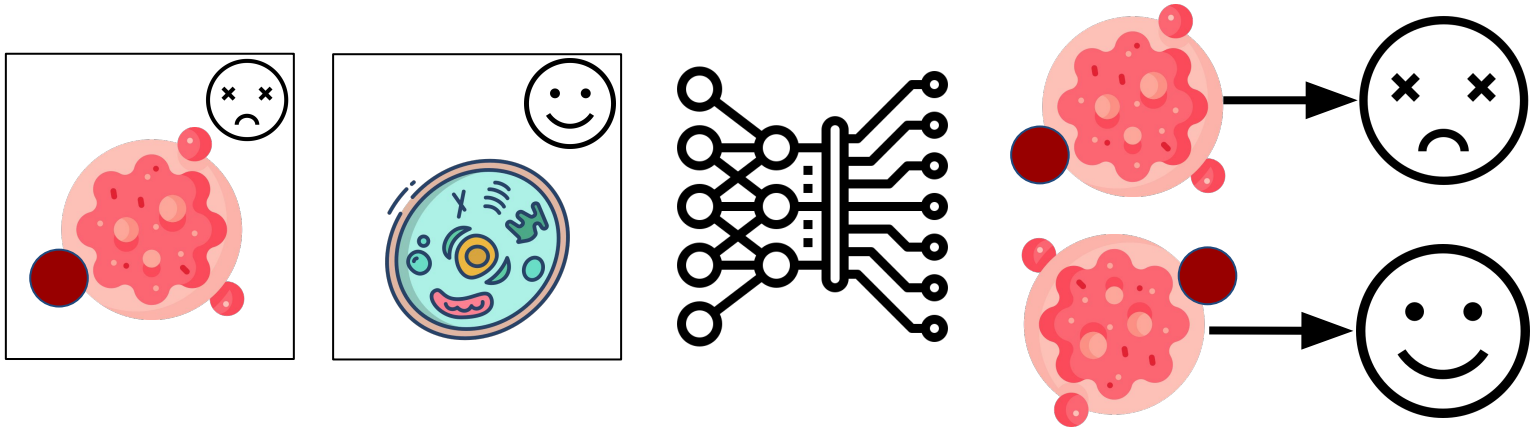
- There are variations that we don't care about:



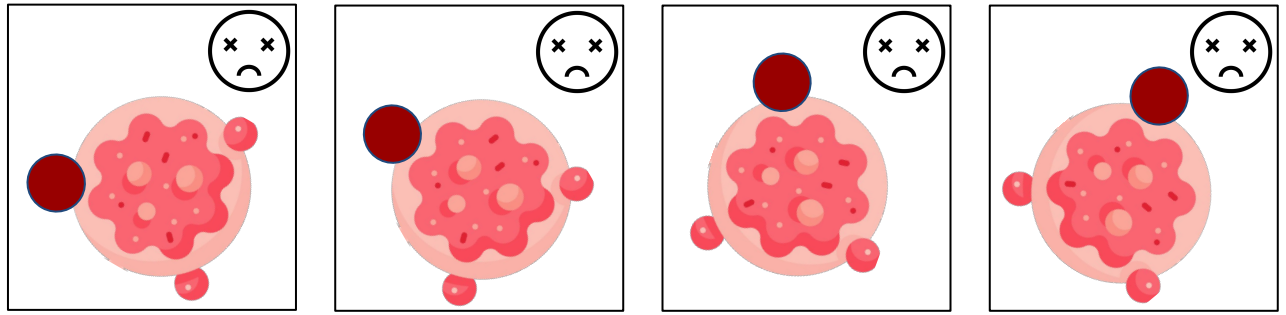
[*] https://www.augmentedstartups.com/blog/convolutional-neural-networks-cnn-in-self-driving-cars

Benefits of Invariance: Motivating Example 2

- There are variations that we don't care about:



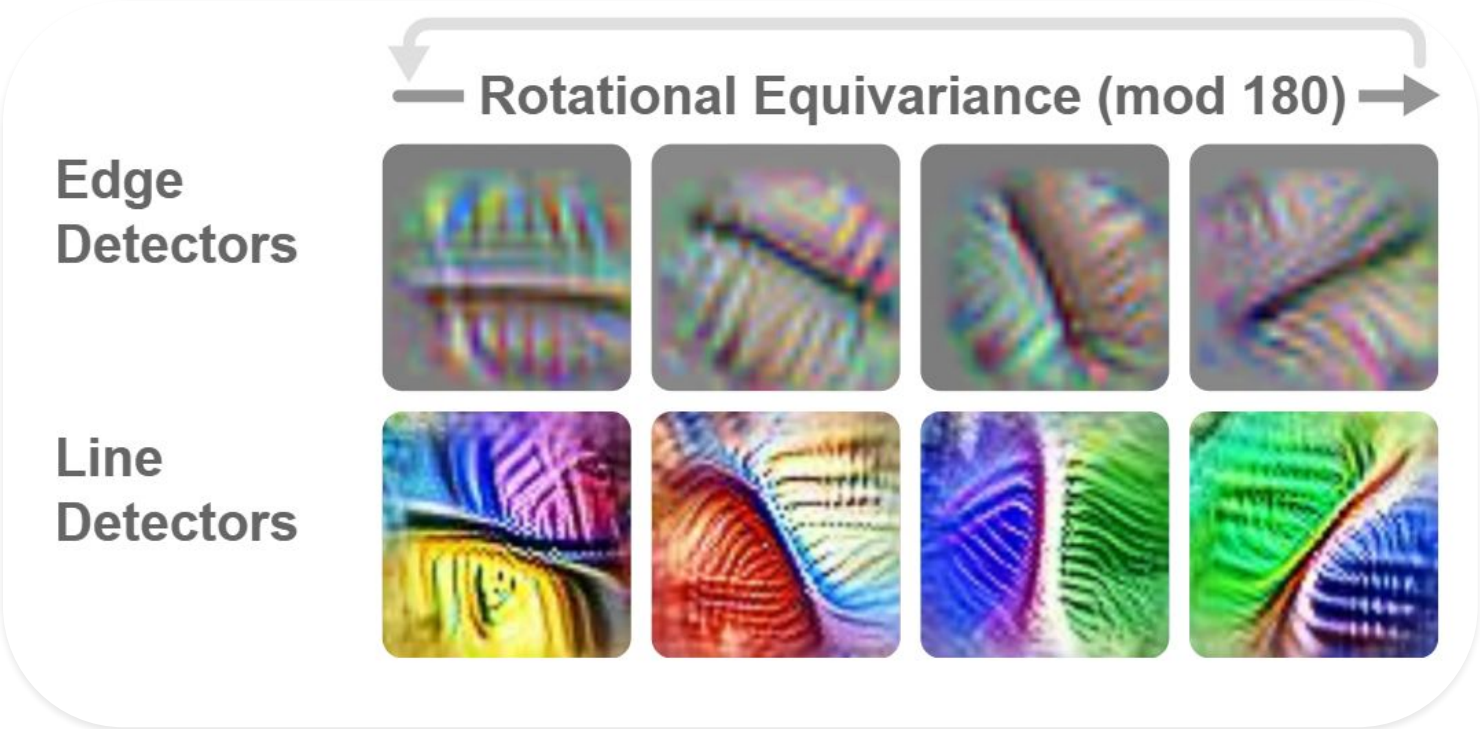
- One possible solution:

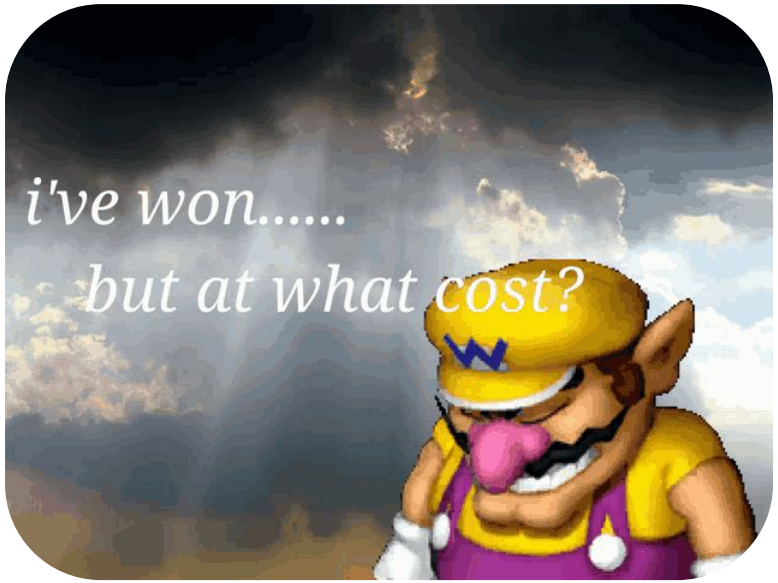


[*] https://uvagedl.github.io/

CNNs Can Learn Invariance But…

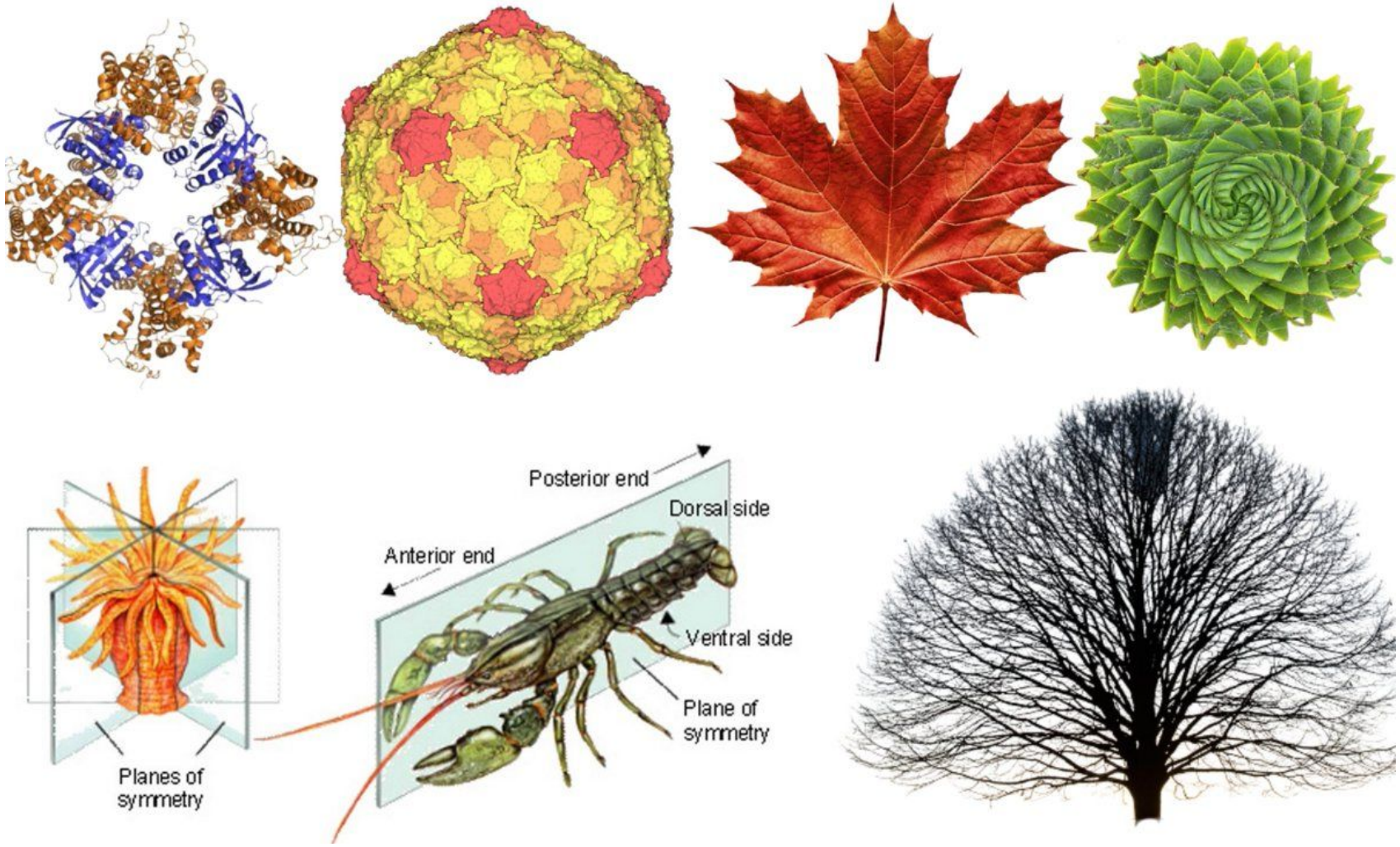
- Augmenting the dataset is helpful, however major issues remain:
  - Learning capacity of the model is used up
  - Feature representations become repetitive
  - There's no guarantee of invariance





[*] https://distill.pub/2020/circuits/equivariance/

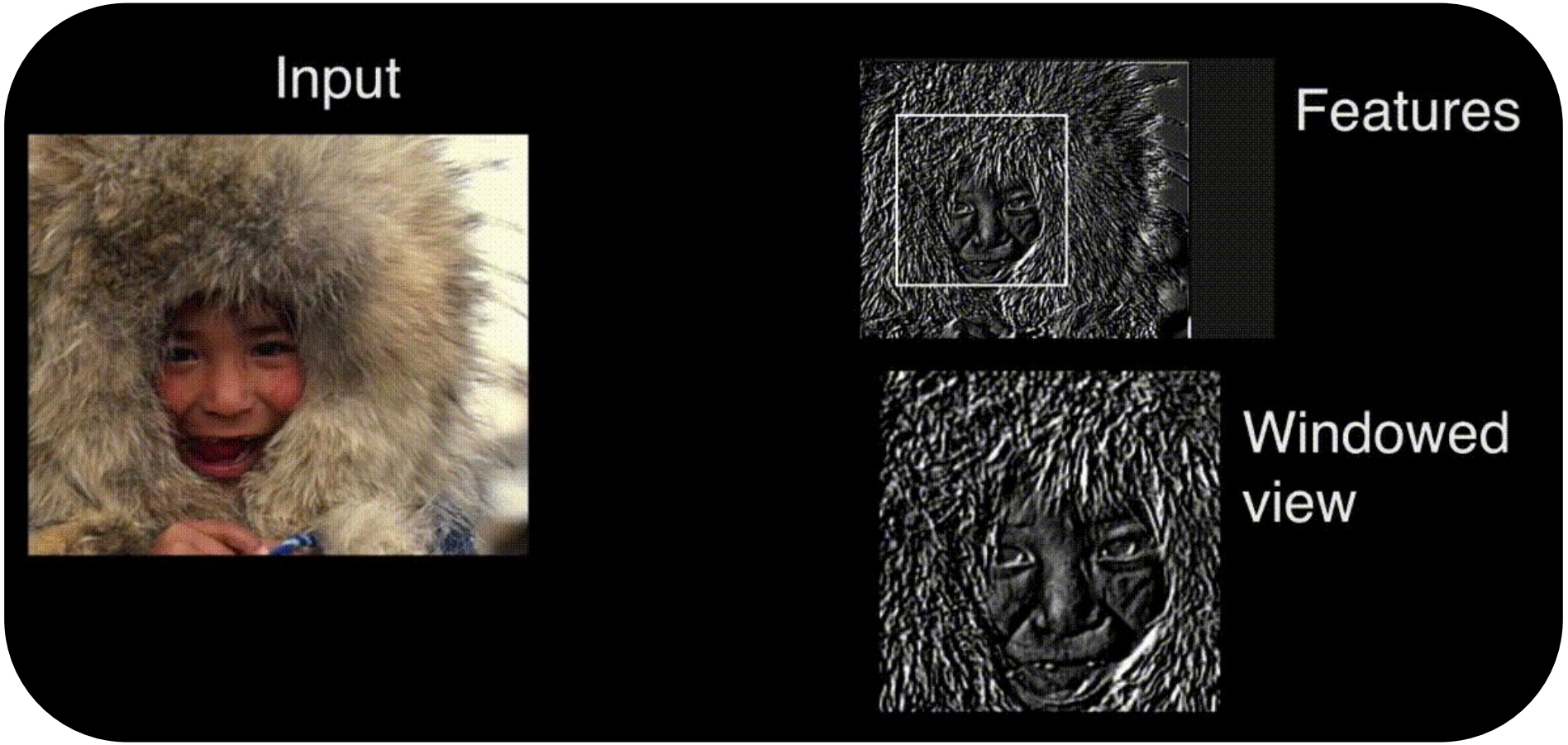
Intuition: Let's Exploit the Symmetry in the Data

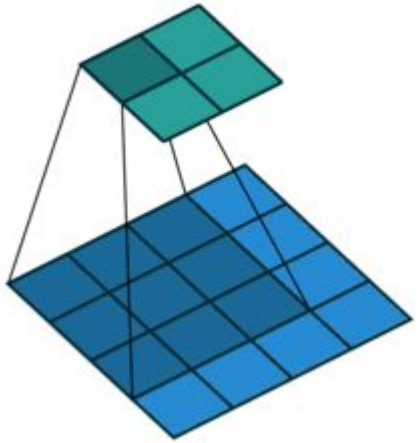


[*] https://twitter.com/evoluchico/status/1503307804043583488

CNNs Already Exploit Translation

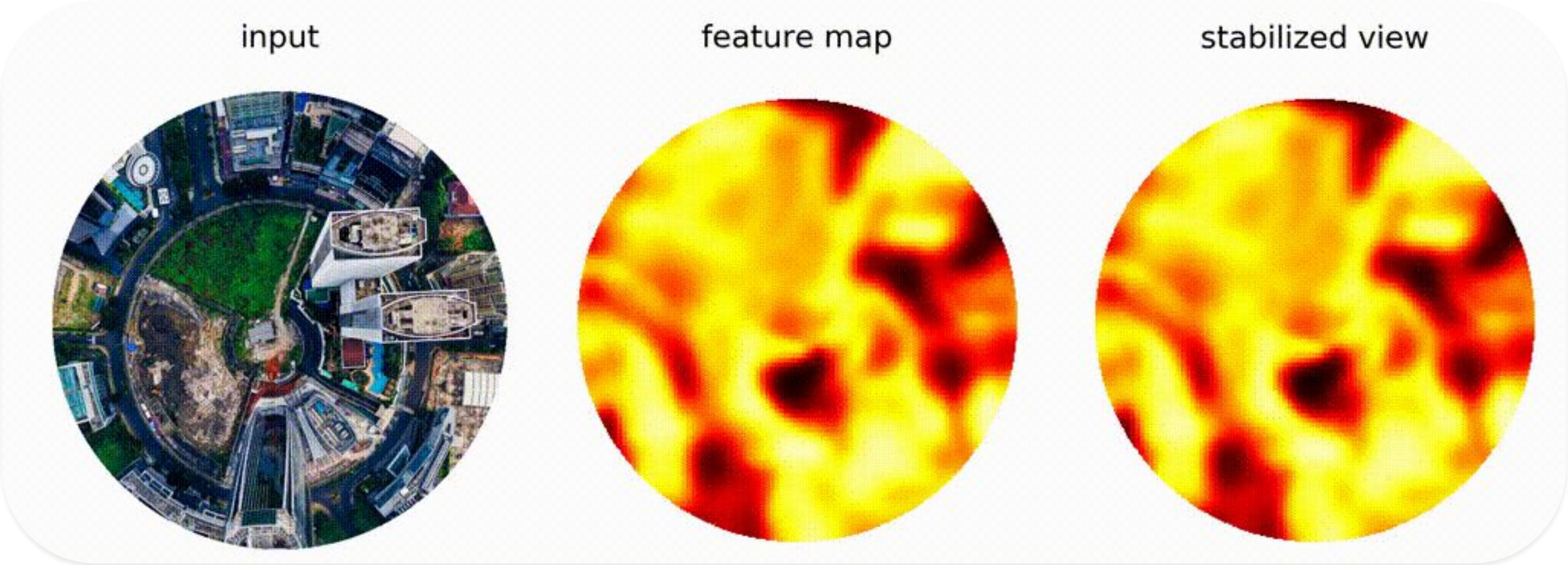
- Convolutional weight sharing → translation symmetry





[*] https://fabianfuchsml.github.io/equivariance1of2/
[**] https://github.com/vdumoulin/conv_arithmetic

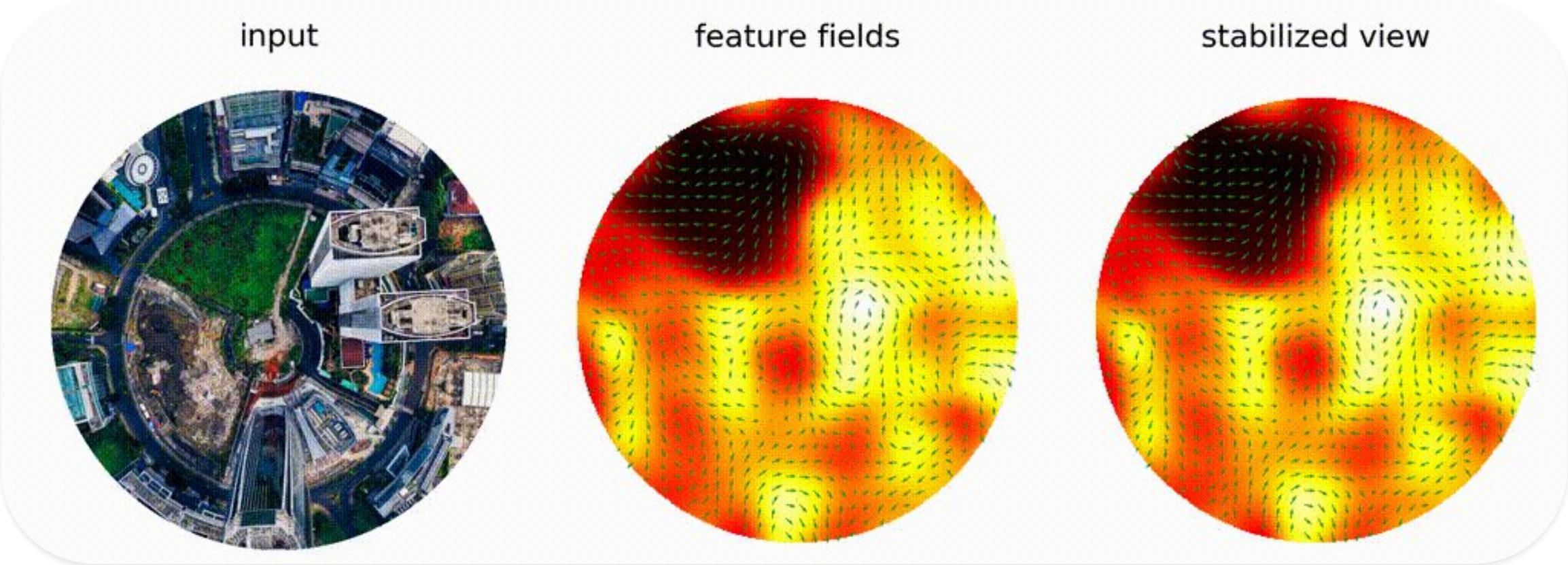
But, CNNs Can Do Better



[*] https://github.com/QUVA-Lab/e2cnn

Generalize CNNs To Exploit Larger Symmetry Groups

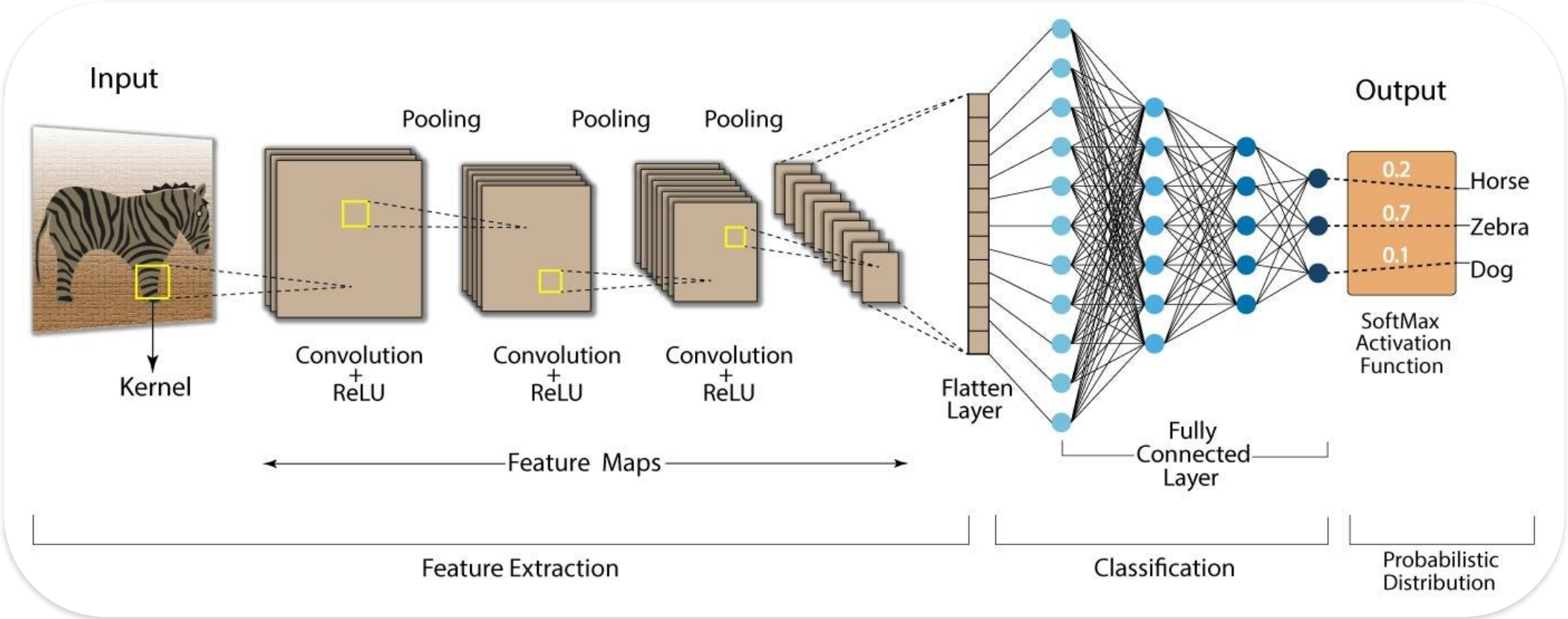
- CNNs handle translation
- Improve CNNs to handle rotations and reflections



[*] https://github.com/QUVA-Lab/e2cnn

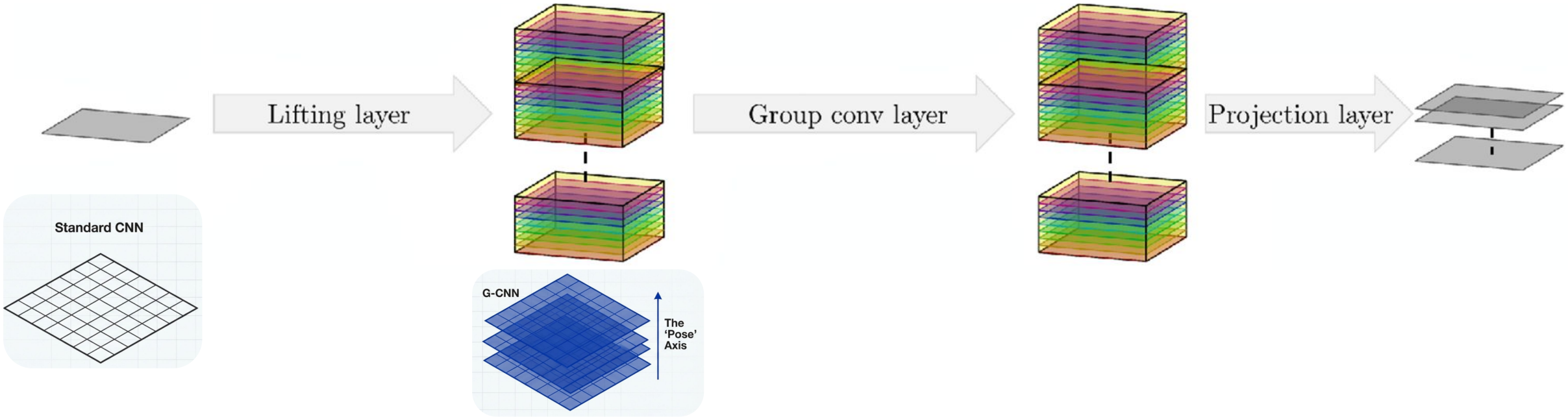
CNN's Status-Quo: Unstructured Representations

- **Minimal internal structure** in representation spaces of standard deep neural networks
- Feature maps are list of scalar numbers
- input = a grid of values => feature map = grid of abstracted representation of values



The Paradigm Shift: Linear G-Space

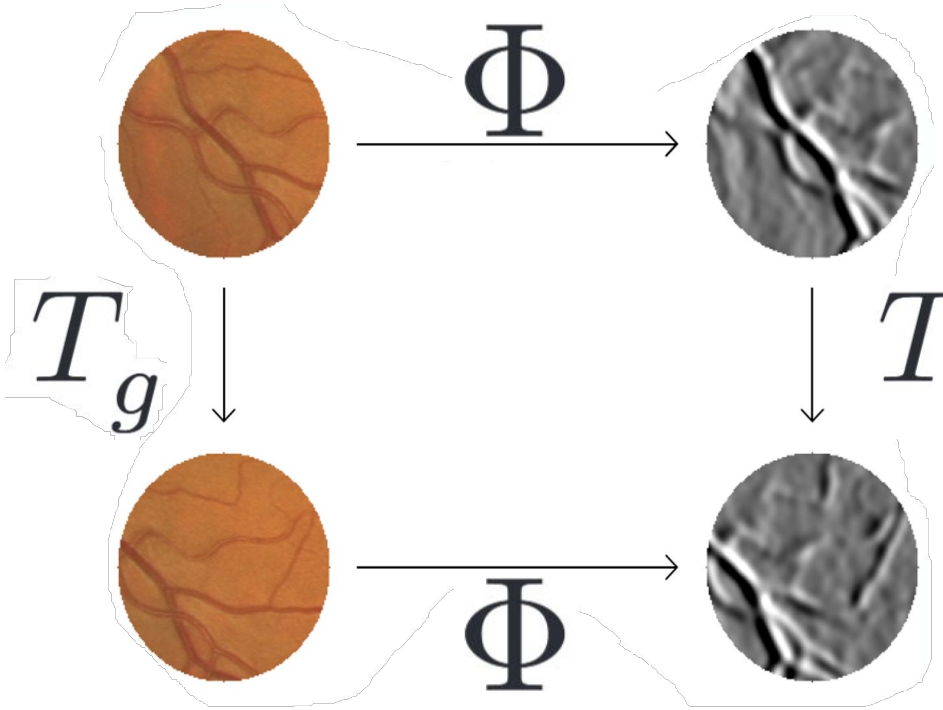
- CNNs operate in the G-Space of 2D translation group (Z^2)
  - Convolution operation -> Sharing weight across the Z^2 group
- Main idea:
  - Choose a more **advanced symmetry group**
  - Lift the input to contain “pose” vectors to be **transformed by the group elements**
  - Use “G-Convolution Layers” to **preserve the input structure**



Equivariance: A Must for the G-Convolution Layers

- Layer Φ should be structure preserving
- This can be achieved by Φ being equivariant:
  - Transform -> Φ = Φ -> Transform
- T and T’ can be different but below should hold:
  - As long as the T is a linear operator
  - T (gh) = T(g)T(h)

(T_g and T'_g represent the same group action,
but can be different physical operations)



$$\Phi(T_g\, x) = T'_g\, \Phi(x)$$

Symmetry Groups

- Symmetry: Transformation leaving an object invariant
- Example:

$$\mathbb{Z}^2 = \{(n, m) \mid n, m \in \mathbb{Z}\}.$$

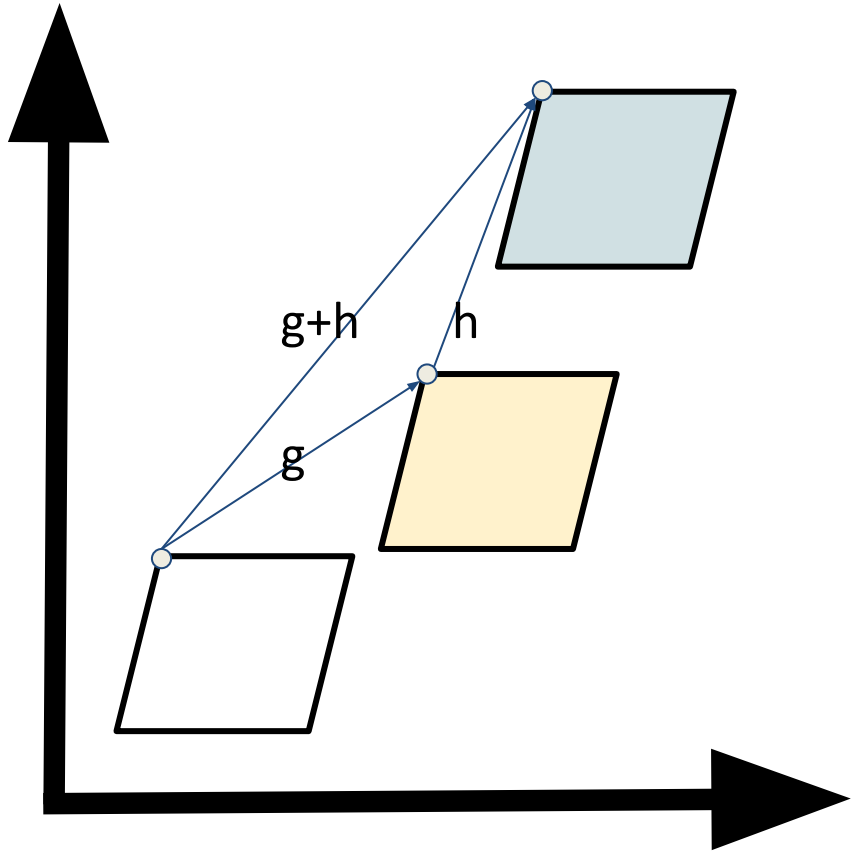
$$(n, m) + (p, q) = (n + p, m + q).$$

Closure: if $g, h \in \mathbb{Z}^2$, then $g + h \in \mathbb{Z}^2$.

Associativity: $(a + b) + c = a + (b + c)$.

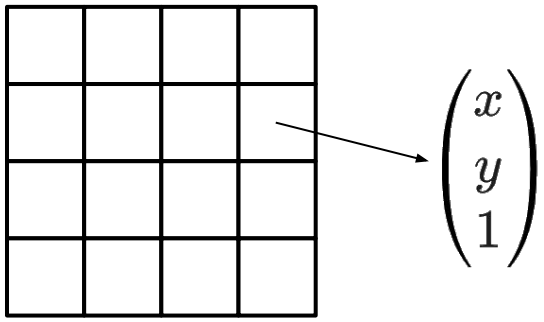
Identity: $e = (0, 0)$ such that $a + e = a$.

Inverse: for $a = (n, m)$, $-a = (-n, -m)$ and $a + (-a) = e$.



p4 and p4m Groups

- p4: rotation + translation on a grid, p4m: rotation + translation + reflection on a grid
- $m \in \{0, 1\}$, $0 \leq r < 4$ and $(u, v) \in Z^2$



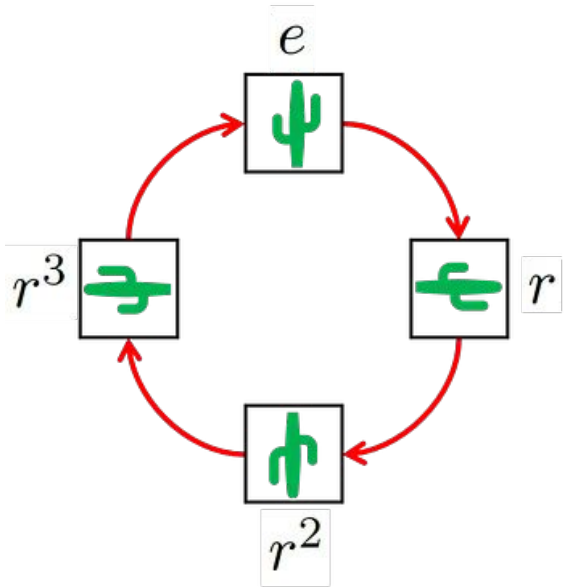
$$\underbrace{\begin{bmatrix} \cos(r\pi/2) & -\sin(r\pi/2) & 0 \\ \sin(r\pi/2) & \cos(r\pi/2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & u \\ 0 & 1 & v \\ 0 & 0 & 1 \end{bmatrix}}_{g(r,u,v)}$$

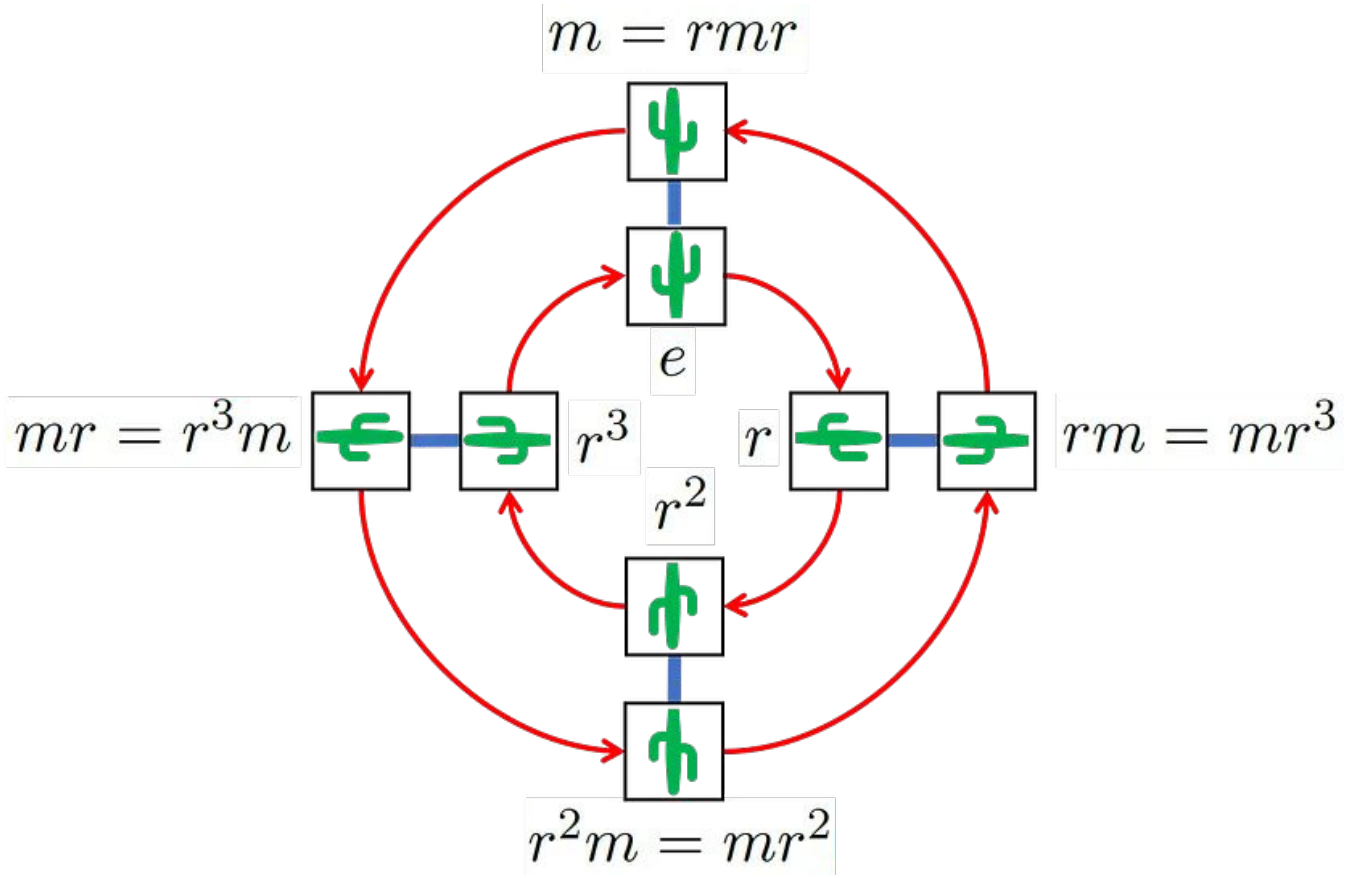
$$g(m, r, u, v) = \begin{bmatrix} (-1)^m & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(r\pi/2) & -\sin(r\pi/2) & 0 \\ \sin(r\pi/2) & \cos(r\pi/2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & u \\ 0 & 1 & v \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} (-1)^m \cos(\frac{r\pi}{2}) & -(-1)^m \sin(\frac{r\pi}{2}) & u \\ \sin(\frac{r\pi}{2}) & \cos(\frac{r\pi}{2}) & v \\ 0 & 0 & 1 \end{bmatrix}$$

p4 and p4m Visualized

- Visualizations use r as rotation and m as reflection:



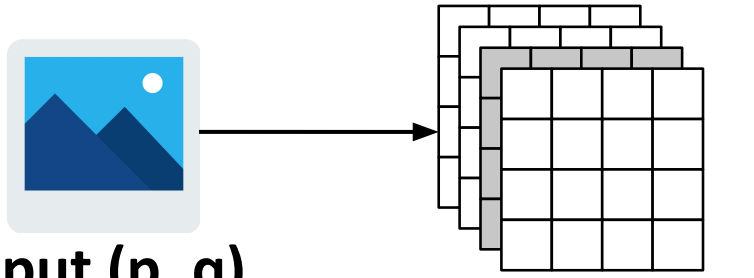


[*] https://medium.com/swlh/geometric-deep-learning-group-equivariant-convolutional-networks-ec687c7a7b41

Functions on Groups: Base Terminology

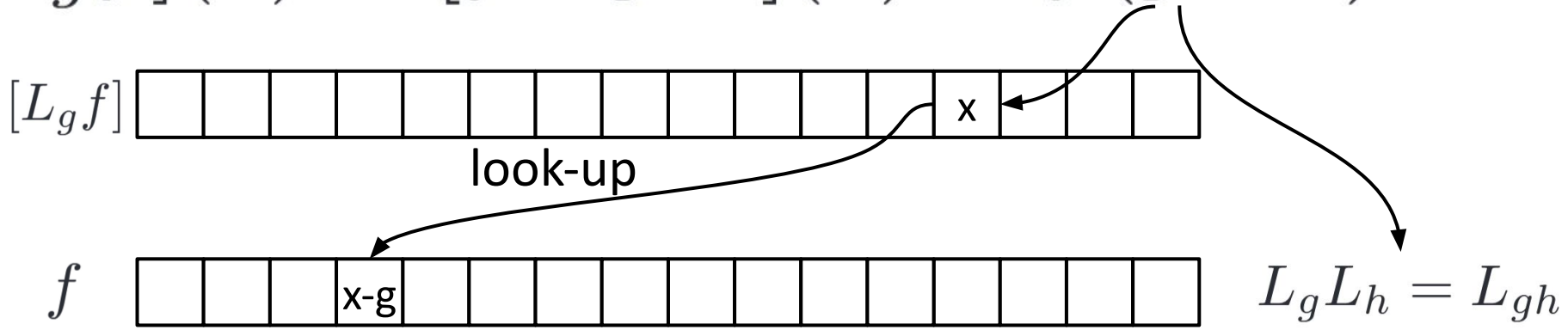
- Model stacks of feature maps in a conventional CNN as a domain bounded function

$$f : \mathbb{Z}^2 \rightarrow \mathbb{R}^K$$



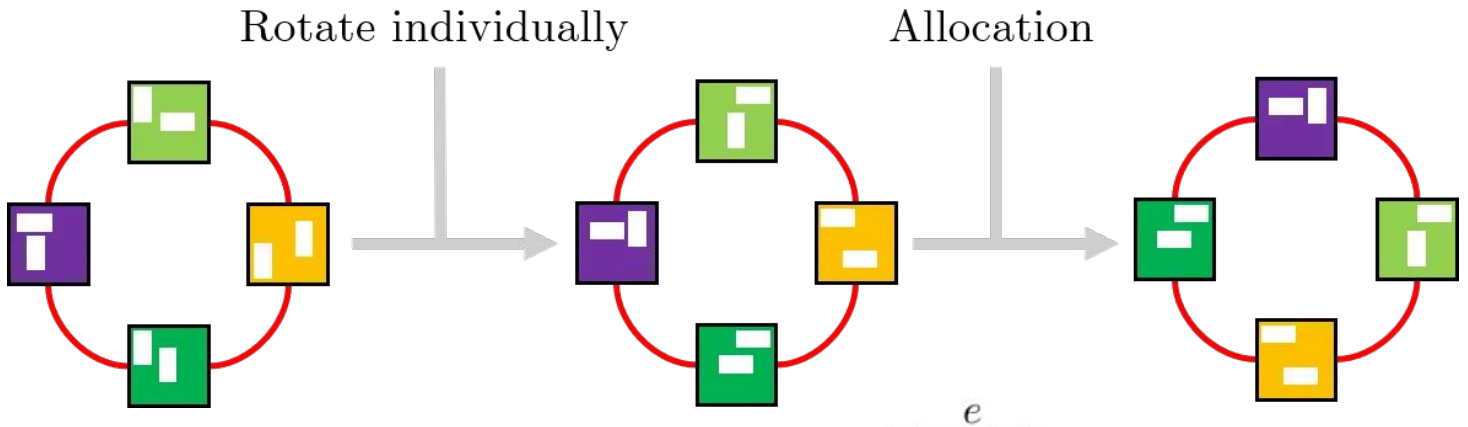
- Use below notation for transformations on feature maps:

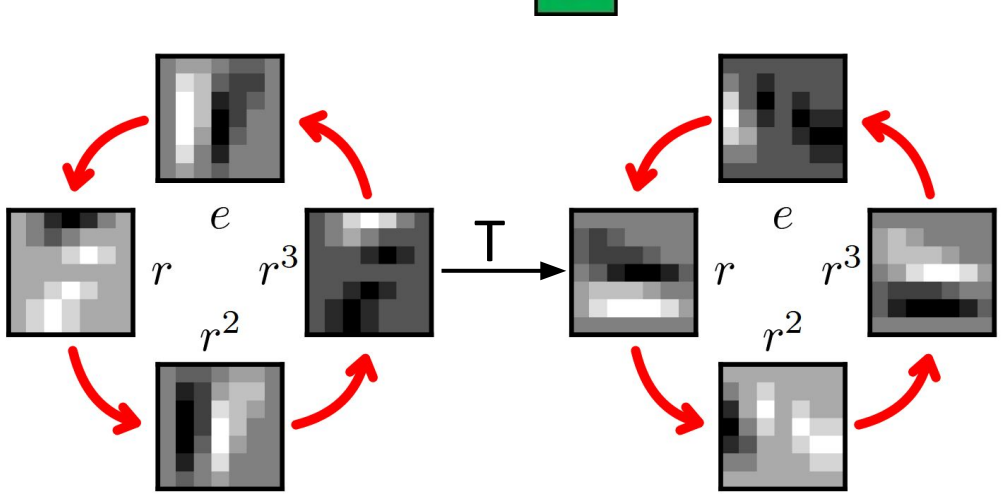
$$[L_g f](x) = [f \circ g^{-1}](x) = f(g^{-1}x)$$

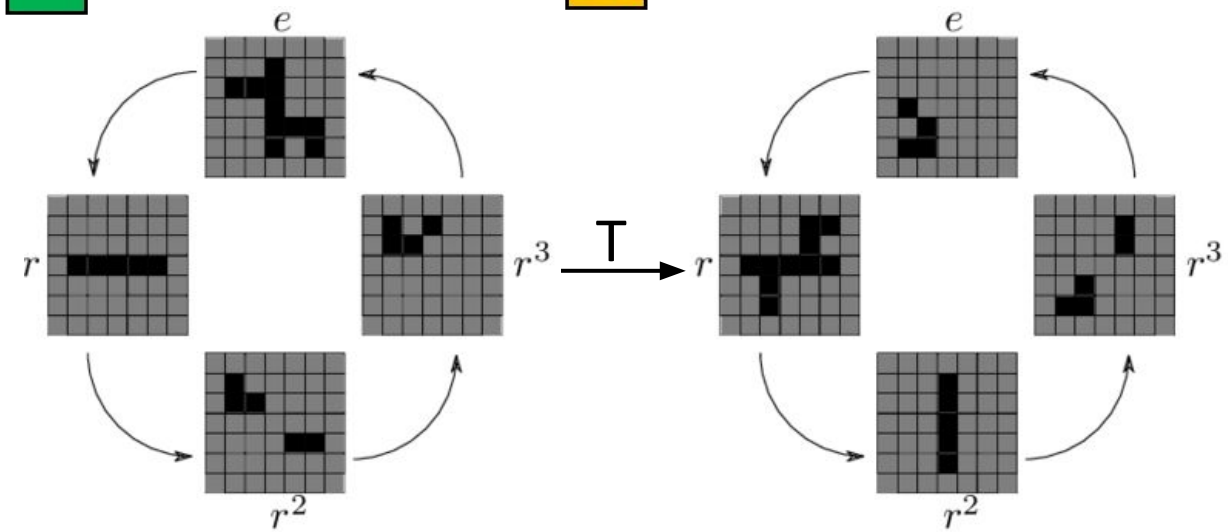


Functions on Groups: p4 Visualization

- Transformation of a structured object (T):



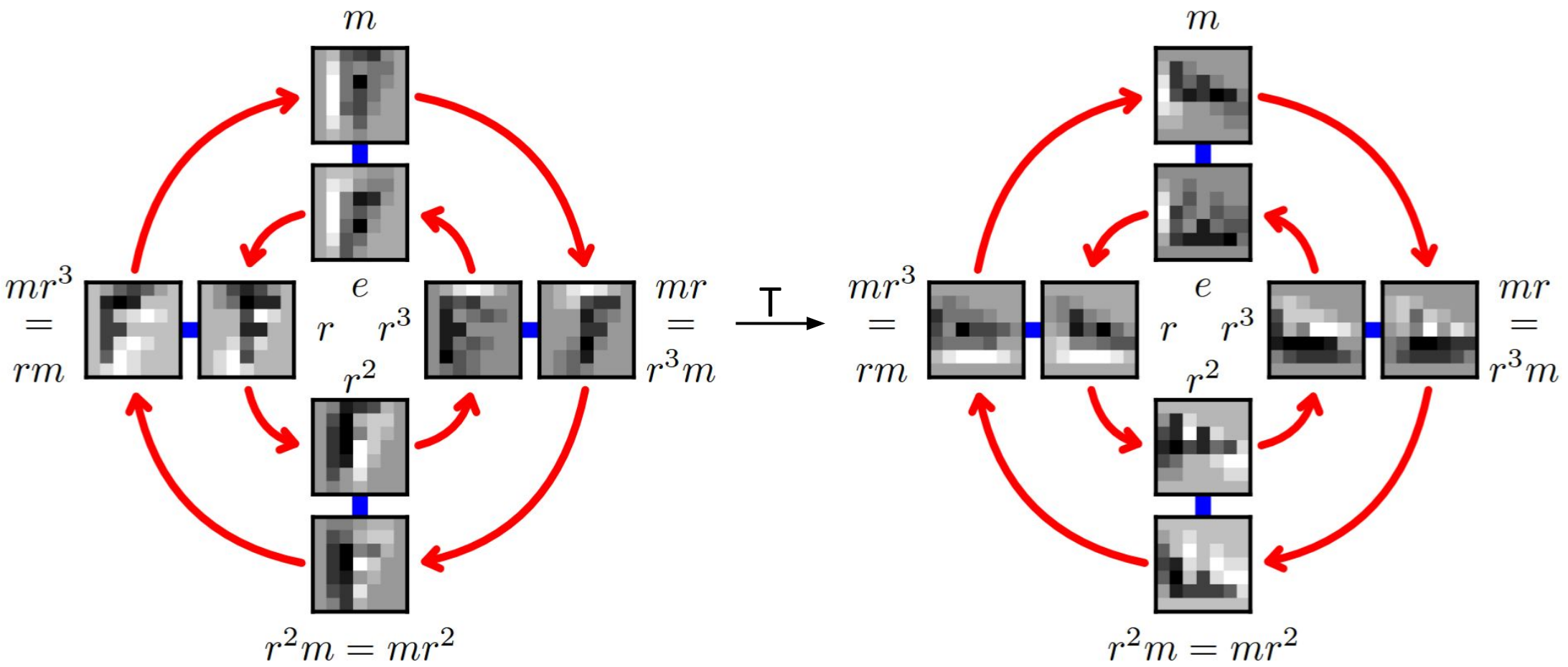




[*] https://medium.com/swlh/geometric-deep-learning-group-equivariant-convolutional-networks-ec687c7a7b41

Functions on Groups: pm4 Visualization

- Again, first transform and then follow the red line:



Quick Recap (EQUIVARIANCE VS. INVARIANCE)

Invariance

- The output remains identical regardless of transformation.

$$\Phi (L_g f) = \Phi (f)$$

Equivariance

- The output transforms in the same way as the input. (e.g., Convolution).

$$\Phi (L_g f) = L_g \Phi (f)$$

- Equivariance is preferred in intermediate layers because it preserves relative spatial information.

Properties of CNNs

- Formulation of Convnet at:
  - layer l
  - Input feature map $f_k(y) \quad f : \mathbb{Z}^2 \rightarrow \mathbb{R}^{K^l}$
  - Filter (or Kernel) ψ_k

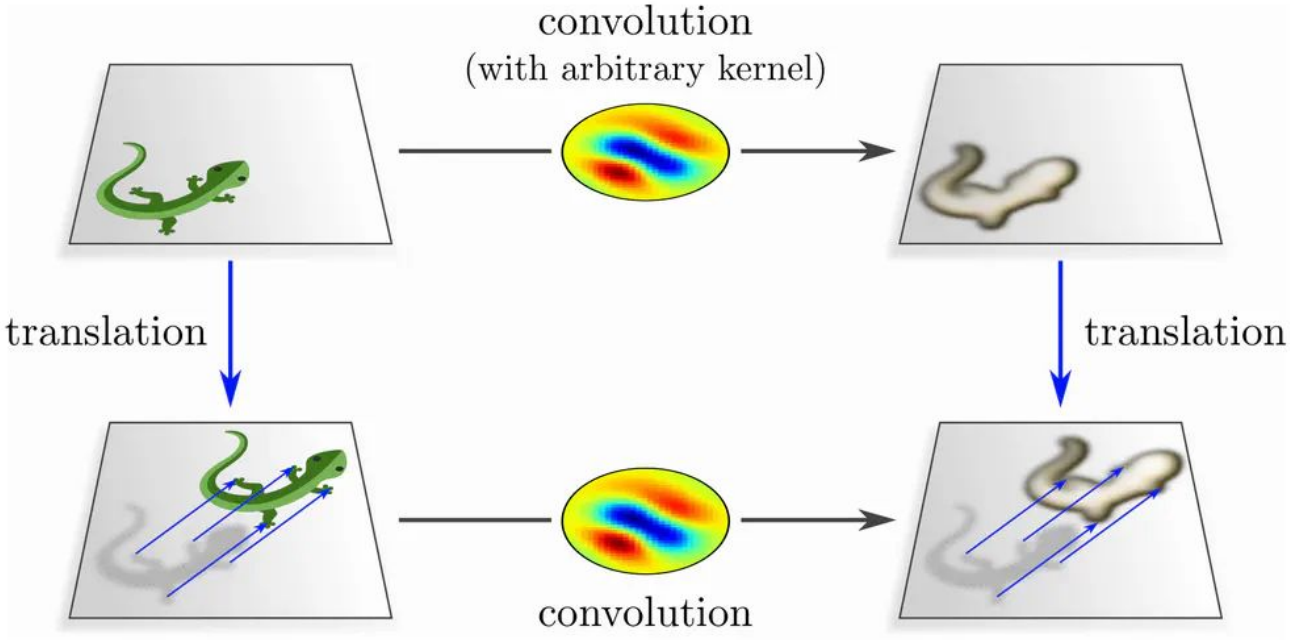
$$[f * \psi^i](x) = \sum_{y \in \mathbb{Z}^2} \sum_{k=1}^{K^l} f_k(y) \psi_k^i(x - y)$$

$$[f \star \psi^i](x) = \sum_{y \in \mathbb{Z}^2} \sum_{k=1}^{K^l} f_k(y) \psi_k^i(y - x) \tag{7}$$

Properties of CNNs (translation)

- Correlation is an equivariant map for the **translation group**

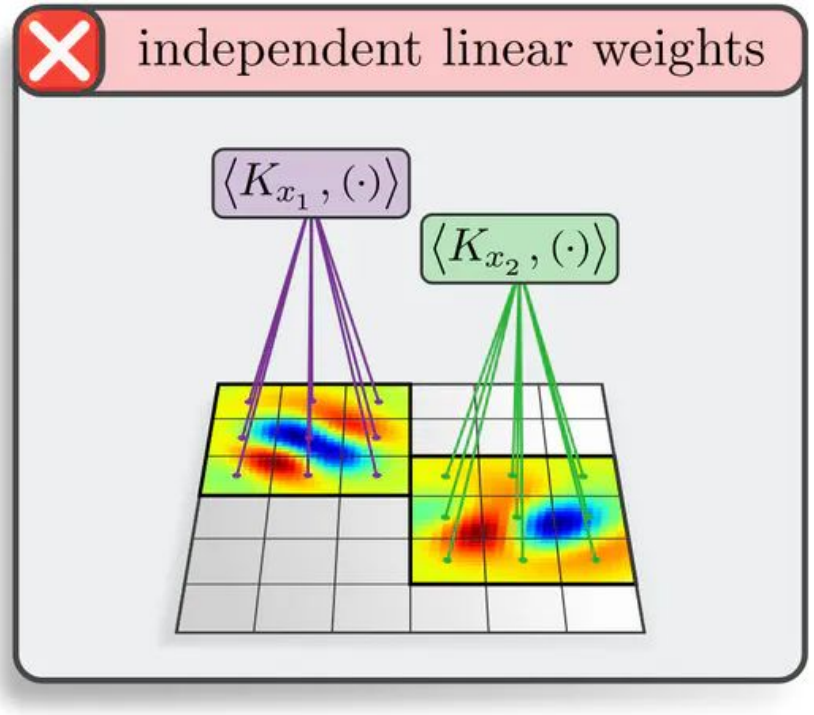
$$L_t : y \rightarrow y + t$$

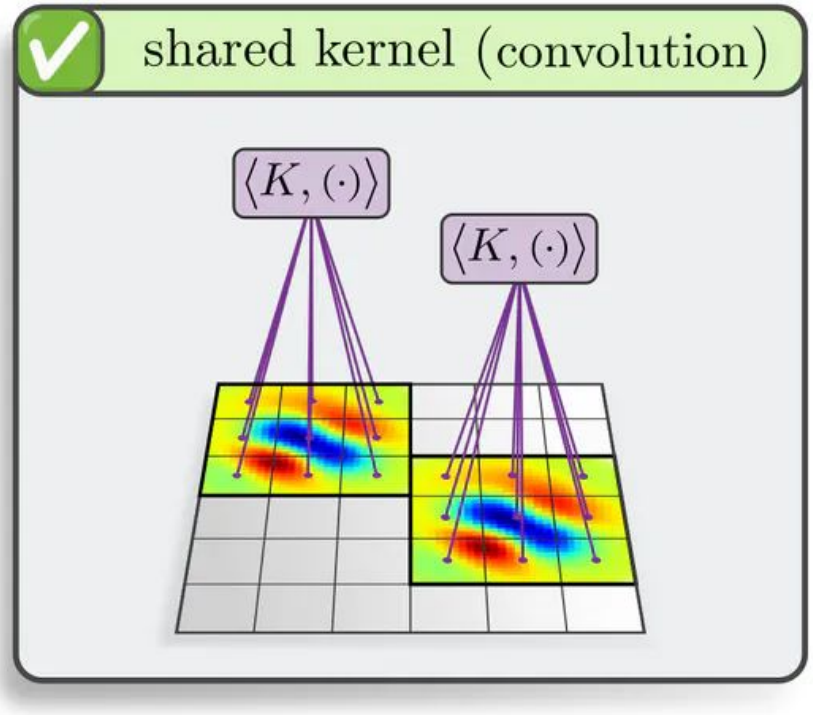


[*] https://maurice-weiler.gitlab.io/blog_post/cnn-book_2_conventional_cnns/

Properties of CNNs (translation)

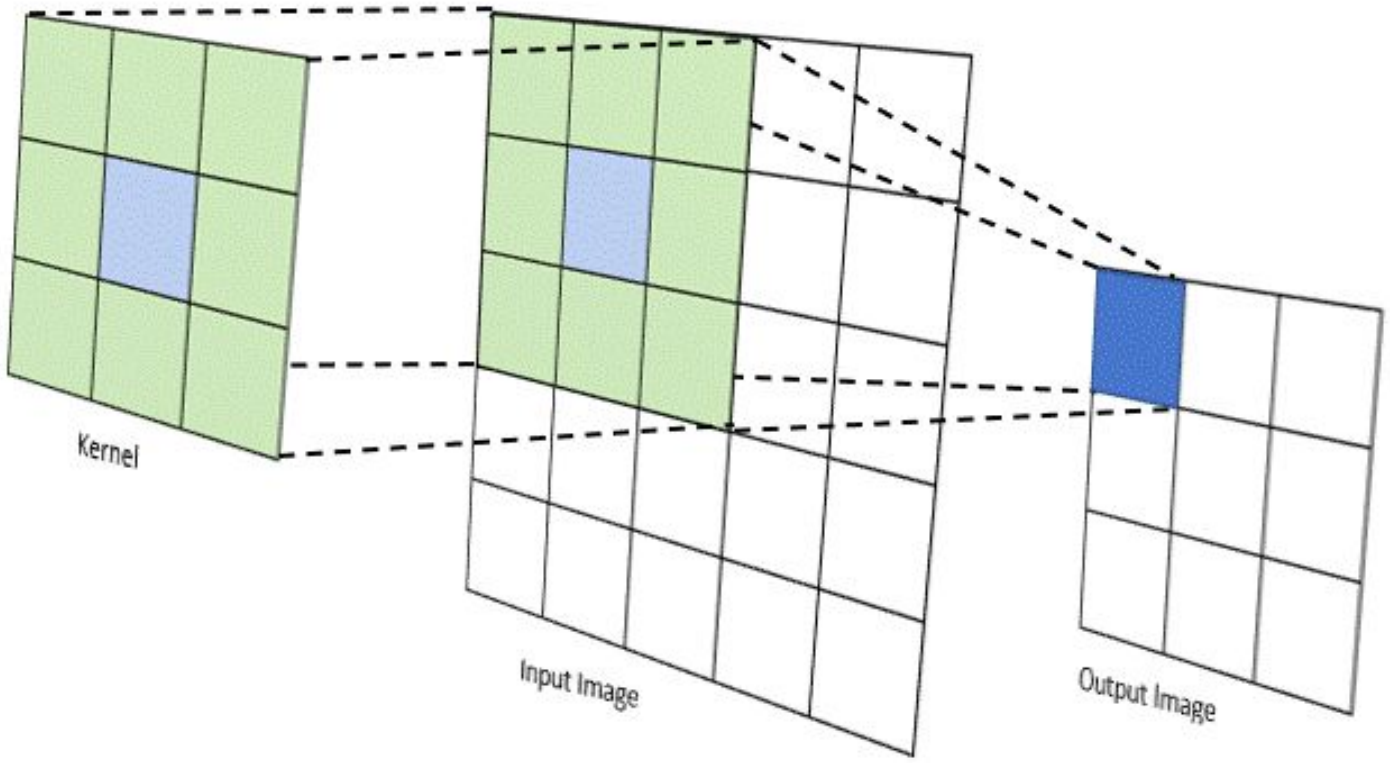
Weight sharing (using the same filter at every pixel) is what provides translation equivariance. However, standard filters are **not shared** across rotations.





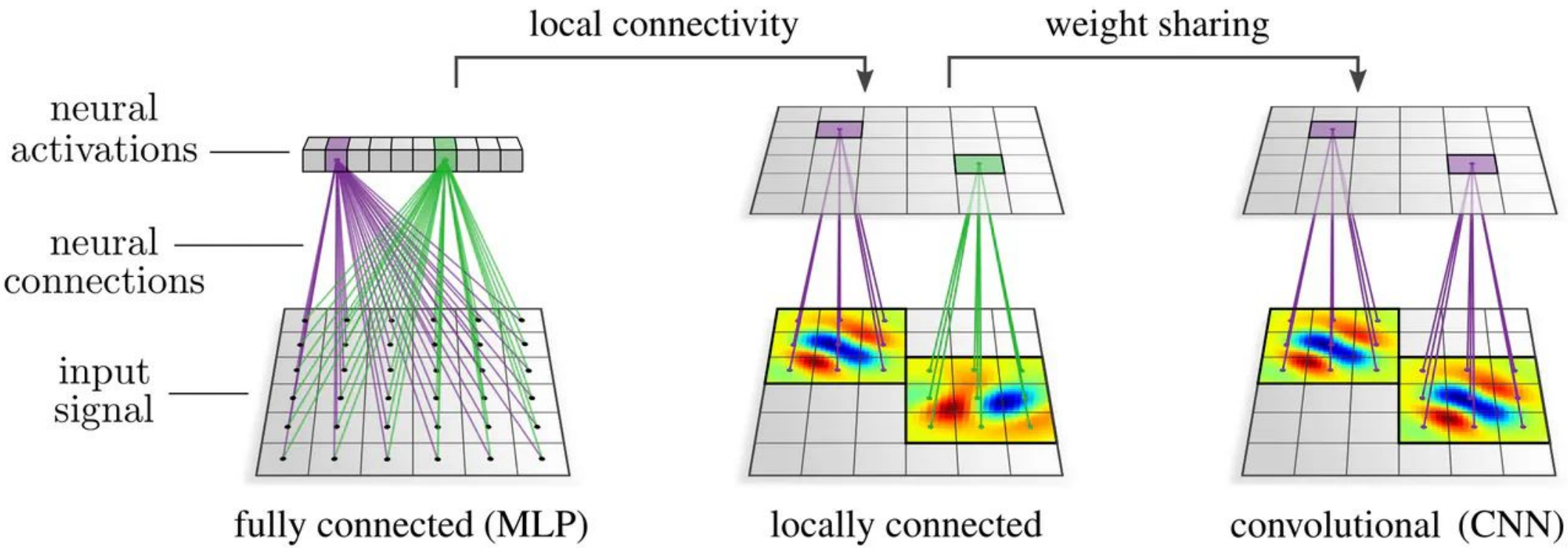
[*] https://maurice-weiler.gitlab.io/blog_post/cnn-book_2_conventional_cnns/

Properties of CNNs (translation)



[*] https://medium.com/@timothy_terati/image-convolution-filtering-a54dce7c786b

Properties of CNNs (translation)



[*] https://maurice-weiler.gitlab.io/blog_post/cnn-book_2_conventional_cnns/

Properties of CNNs (rotation)

- Correlation is not equivariant map for the **translation group**

In a standard CNN, the filter (kernel) is fixed. Imagine you have a filter that detects horizontal edges.

- If you give it an image of a horizontal line, it lights up.
- If you rotate the image 90 degrees, that line is now vertical.
- Since your filter is still looking for horizontal edges, it now detects nothing.

Properties of CNNs (rotation)

- How did we handle this before?

$$[[L_r f] \star \psi](x) = L_r[f \star [L_{r-1}\psi]](x) \qquad (9)$$

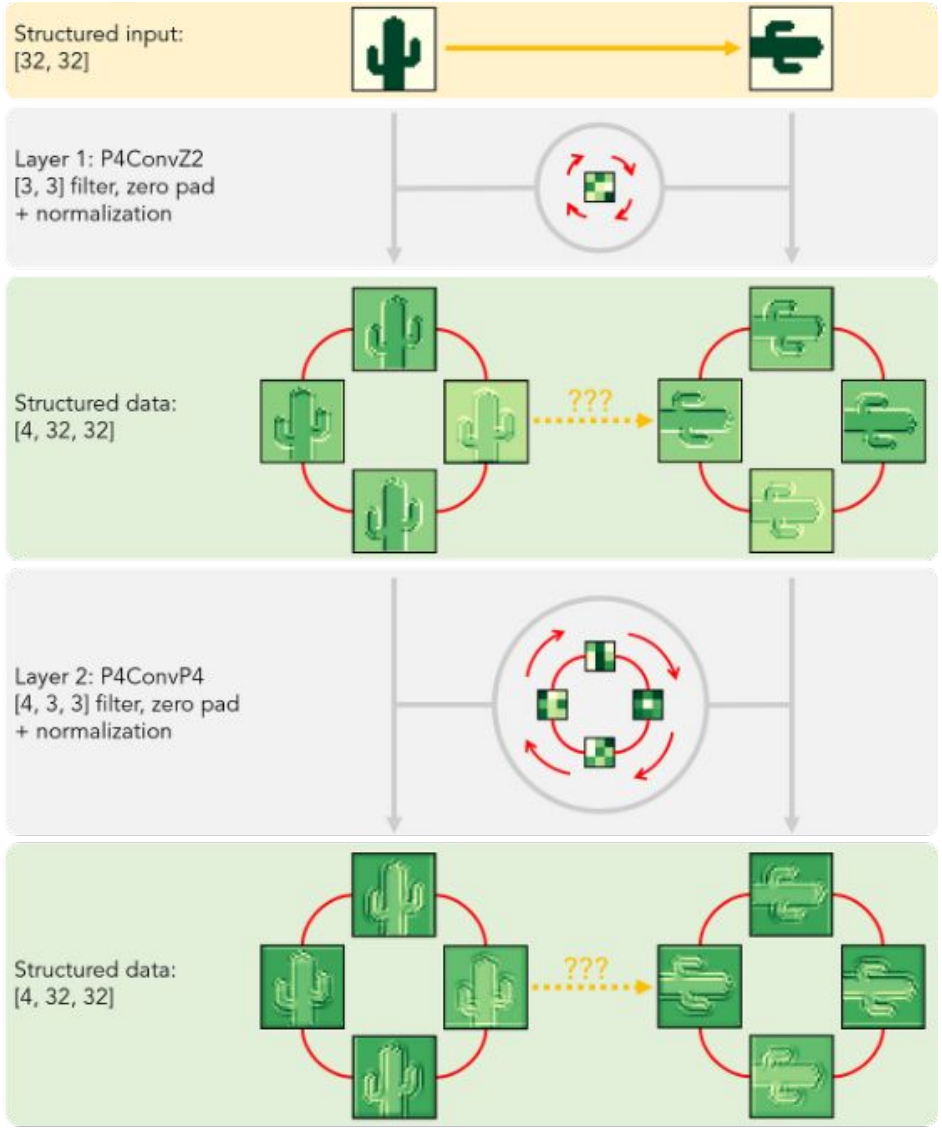
If an ordinary CNN learns rotated copies of the same filter, the stack of feature maps is equivariant, although individual feature maps are not.

Properties of CNNs (rotation)

Static Filters

In standard CNNs, filters are fixed in orientation. If the image rotates, the filter response changes completely.

- A horizontal edge detector won't detect a vertical edge.
- The network must learn redundant rotated filters.
- Sample complexity increases linearly with pose variety.



[*] https://medium.com/swlh/geometric-deep-learning-group-equivariant-convolutional-networks-ec687c7a7b41

Equivariance Properties of CNNs

- Using the substitution: $y \longrightarrow y + t$
- This substitution is a Translation function L_t

$$
\begin{aligned}
[[L_t f] \star \psi](x) &= \sum_y f(y - t)\psi(y - x) \\
&= \sum_y f(y)\psi(y + t - x) \\
&= \sum_y f(y)\psi(y - (x - t)) \\
&= [L_t[f \star \psi]](x).
\end{aligned} \tag{8}
$$

G-CNNs

- CNNs Inherently exploit translation symmetry. This is a desirable trait.
- Now how do we generalize this to larger symmetry groups (e.g., rotations).
- The network is built from layers that are **internally equivariant** to the group.

What we need to do:

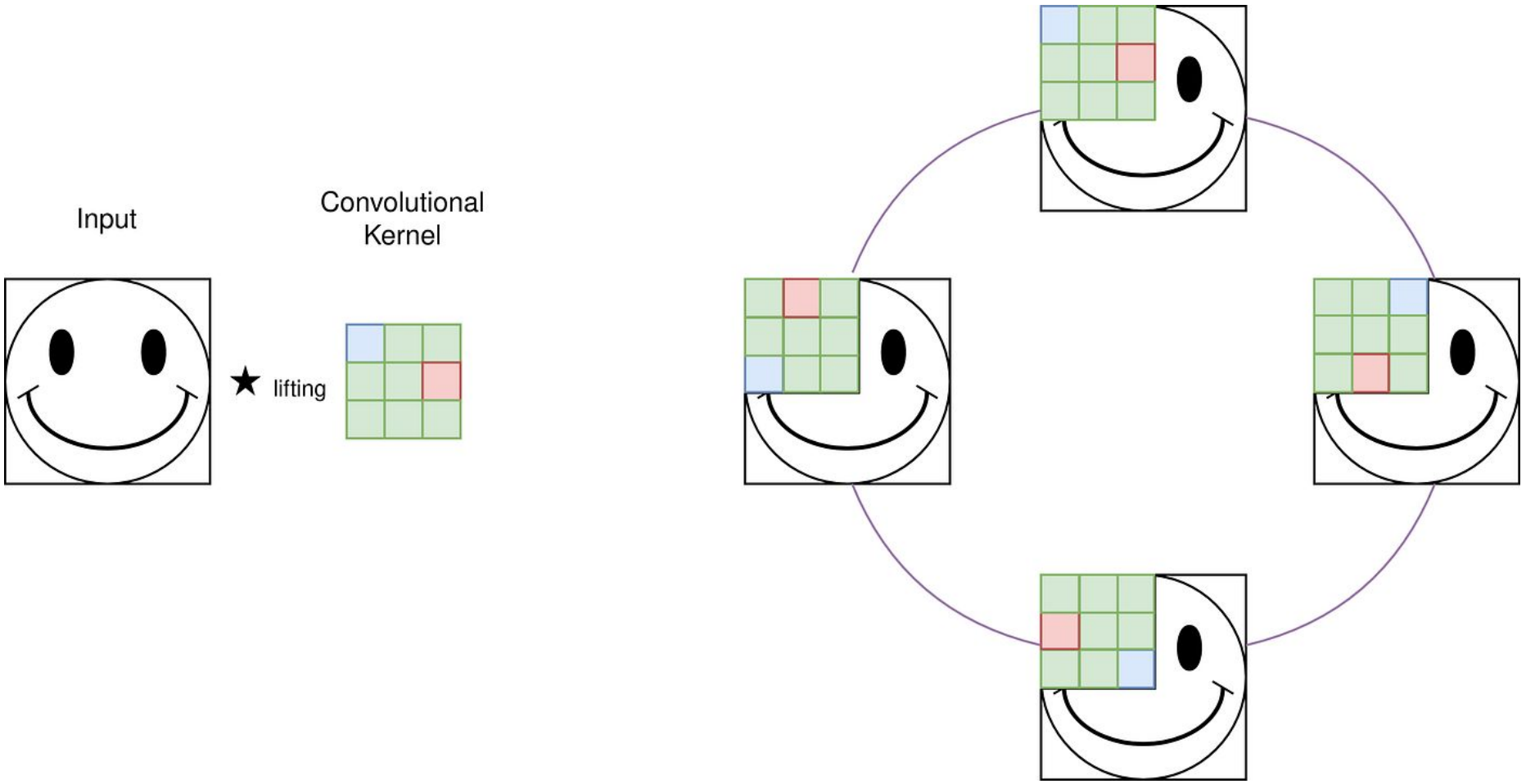
➢ Standard CNN: A feature map is a 2D grid where each value tells you where a feature (like an eye) is located in the image.

➢ G-CNN: A feature map is a 3D stack defined on a group (like $p4$). Each layer in the stack tells you not just where a feature is, but also in which orientation (0°, 90°, 180°, or 270°) it was found.

Step 1: The Lifting Convolution

The first layer must "lift" the input from the 2D grid into the Group space G.

- Input: 2D Image
- Output: Group Function $Z : G \to \mathbb{R}^L$
- The filter is rotated and applied to the image repeatedly.



[*] https://medium.com/data-science/group-equivariant-self-attention-19e47f0b786e

The difference

- Standard Convolution: Takes a 2D image and slides a single filter across it to produce a single 2D output.

- Lifting Convolution: Takes the 2D image but applies the filter in multiple orientations (e.g., rotated at 0°, 90°, 180°, and 270°).

Step 2: Hidden Layers (G to G)

Once data is on the group manifold, we must process it with layers that stay on the group.

- Input: Function on the group G.
- Operator: **Group-to-Group Correlation**.
- Filter: Also defined on the group G.
- Outcome: Hierarchical features that understand relative orientation.

The difference

Standard CNN

Weights shared only over spatial translations . (u, v)

G-CNN (p4m)

Weights shared over translations **AND** 8 orientations.

Instead of just looking at neighboring pixels (left, right, up, down), the filter now looks at neighboring orientations too. It asks: "Is there a vertical edge here AND a horizontal edge in the layer representing 90 degrees?"

By doing this, the network can learn complex patterns (like a corner or a face) regardless of how they are rotated, because the filter itself "rotates" through the stack as it convolved. This ensures that the rotational information is preserved all the way to the final layers of the network.

Massive Weight Sharing Benefits

Standard CNN: Shares weights across (x, y).

G-CNN: Shares weights across (x, y) AND all rotations r and mirror.

Efficiency: Learns one detector, gets 4 (or 8) orientations for free.

Accuracy: Structural prior prevents learning redundant features.

Step 3: Pooling in G-Space

Once data is on the group manifold, we must process it with layers that stay on the group.

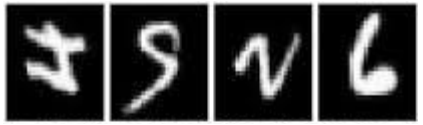
- Input: Function on the group G.
- Operator: **Group-to-Group Correlation**.
- Filter: Also defined on the group G.
- Outcome: Hierarchical features that understand relative orientation.

SUBGROUP POOLING

To produce a classification label, the network must eventually become invariant to pose. This is done by pooling across the group channels.

- Example: Pool over rotations but keep translations.
- Result: A standard 2D feature map that is rotation-invariant.

Experiments: Rotated MNIST



- P4CNN: replace each conv by a p4-conv and added max-pooling over rotations **after last conv layer**
- P4CNNRotationPooling: replace each conv by a p4-conv followed by coset max-pooling over rotations
- → **Premature invariance is undesirable** in a deep architecture

Network	Test Error (%)
SVM (2007)*	10.38 ± 0.27
Transformation-invariant RBM (TIRBM) (2012)**	4.2
Rotation invariant C-RBM (RC-RBM) (2012)***	3.98
Z2CNN	5.03 ± 0.0020
P4CNNRotationPooling	3.21 ± 0.0012
P4CNN	**2.28 ± 0.0004**

[*] Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. (2007) An empirical evaluation of deep architectures on problems with many factors of variation. (ICML)
[**] Sohn, K., & Lee, H. (2012). Learning Invariant Representations with Local Transformations. (ICML)
[***] Schmidt, U., & Roth, S. (2012). Learning rotation-aware features: From invariant priors to equivariant descriptors. (CVPR)

Experiments: CIFAR-10

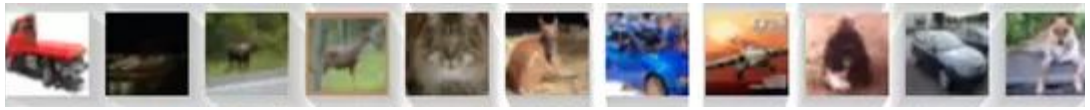
- Replace each conv by a p4/p4m-conv and reduce #filters in each layer
  → approximately invariant #parameters

Network	G	CIFAR10	CIFAR10+	Param.
All-CNN*	$\mathbb{Z}^2$	9.44	8.86	1.37M
	$p4$	8.84	7.67	1.37M
	$p4m$	7.59	7.04	1.22M
ResNet44**	$\mathbb{Z}^2$	9.45	5.61	2.64M
	$p4m$	**6.46**	**4.94**	2.62M

Table 2. Comparison of conventional (i.e. $\mathbb{Z}^2$), $p4$ and $p4m$ CNNs on CIFAR10 and augmented CIFAR10+. Test set error rates and number of parameters are reported.

[*] Springenberg, J.T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for Simplicity: The All Convolutional Net. (ICLR), 2015.
[**] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Identity Mappings in Deep Residual Networks. (ECCV), 2016.

Experiments: CIFAR-10



Network	*G*	Test Error (%)	Param.
Wide ResNet-26 w/ moderate data augmentation	Z2	5.27	~7.2M
	p4m	4.19	**7.2M**
WRN-28-10 (2016)*		4.17	36.5M

[*] Zagoruyko, S. and Komodakis, N. Wide Residual Networks. (BMVC), 2016.

Discussions & Implications

- p4 and p4m-conv can be used as **drop-in replacement** of standard conv

- G-CNNs benefit from **data augmentation** in the same way as convolutional networks
  - As long as augmentation comes from group larger than *G*

- There need not be **full symmetry** in the dataset for G-CNNs to be beneficial

Strengths & Weaknesses

- From unstructured to **structured** representations
  - Concrete example of how adding mathematical structure to a representation can improve NN's ability to see abstract similarities between superficially different concepts

- p4/p4m only works for **discrete** groups
  - Convolution on continuous groups may be hard to approximate in an equivariant manner
  - Full enumeration of transformations in group may not be feasible if group is large

Future Works

- p4/p4m only works for discrete groups
  - General E(2) - Equivariant Steerable CNNs (2019)*
- G-CNNs for 3D space groups
  - Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds (2018)**
  - SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks (2020)***
- Hexagonal lattices with more symmetries than square grids
  - HexaConv (2018)****

[*] Maurice Weiler, Gabriele Cesa. General E(2) - Equivariant Steerable CNNs. (NIPS), 2019.
[**] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, Patrick Riley. Tensor field networks: Rotation- and translation- equivariant neural networks for 3D point clouds. 2018.
[***] Fabian B. Fuchs, Daniel E. Worrall, Volker Fischer, Max Welling. SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks. (NIPS), 2020.
[****] Emiel Hoogeboom, Jorn W.T. Peters, Taco S. Cohen, Max Welling. HexaConv. (ICLR), 2018.

Takeaway

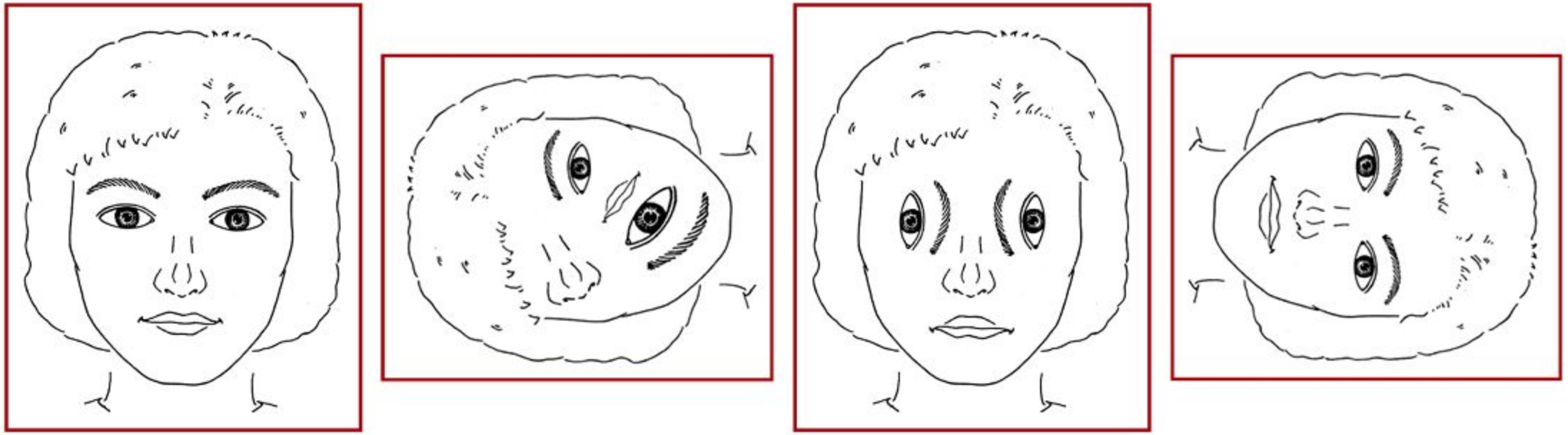
By adding structure to representation space (i.e. linear *G*-space) and enforcing **equivariance** to larger groups of symmetries (i.e. rotations, reflections), the model **shares weights** across transformed features, **reducing redundancy** and **improving generalization**

END

BACKUP

Slide Title

- Points
  - More points

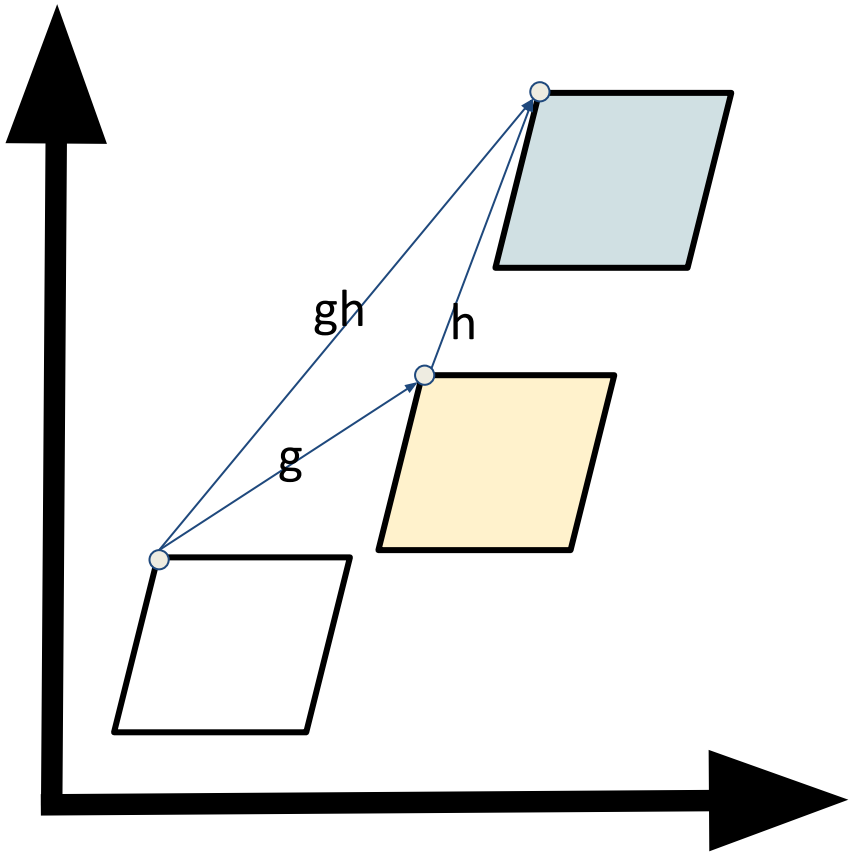


[*] Reference 1
[**] Reference 2
[⁂] Reference 3
[⁂ ⁂] Reference 4

Slide Title

- Points
  - More points

[*] Reference 1
[**] Reference 2
[⁎] Reference 3
[⁎ ⁎] Reference 4


gh
h
g

Slide Examples

Divider