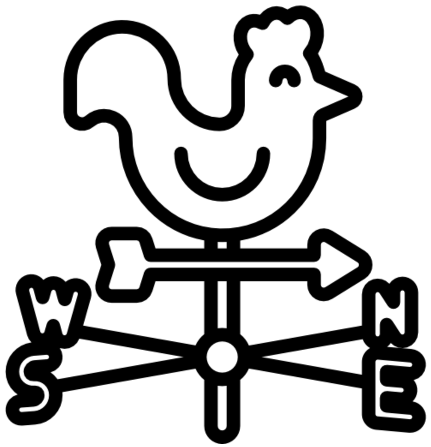
Steerable CNNs

Seoyeon Kim

Cohen, Taco S., and Max Welling. "Steerable cnns.” (2016).

Introduction

- What does ”Steerable” mean? Capable of being controlled
- It takes name from filters.
- Steerable Filters = Wind Vane!



Wind Vane: Rotates to align with the wind direction, so we don’t need a sensor for every possible wind direction.

Steerable Filters: Orientable and adaptable to different orientation of the input image without requiring a separate filter for every possible orientation of the input.

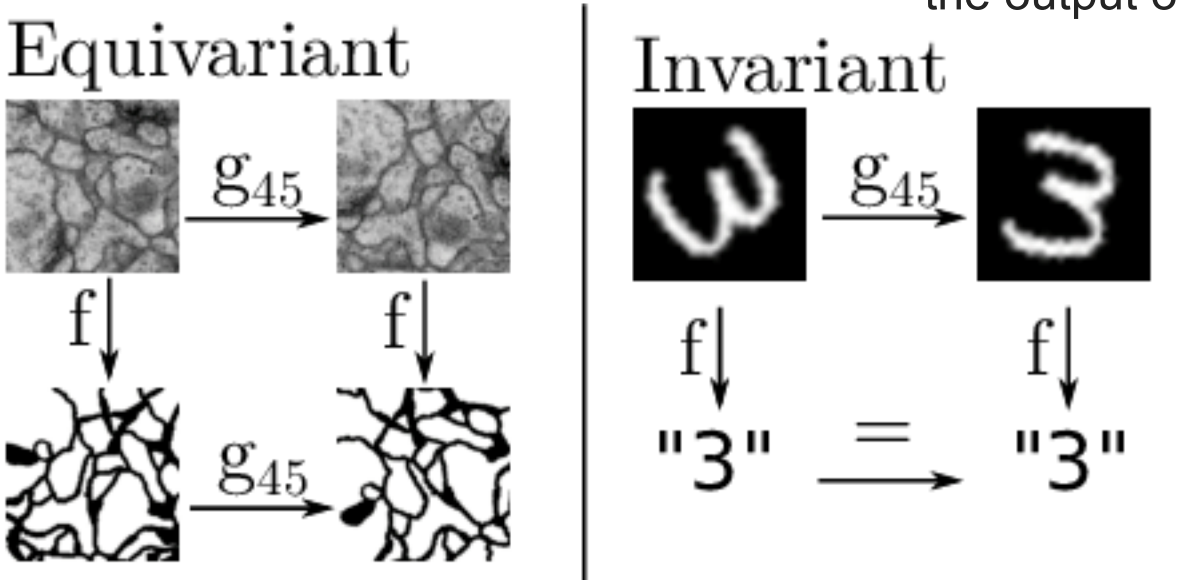
Introduction

- Motivation
  - **Symmetry** in real-world data ex. Rotational, reflection symmetries
  - Limitation of traditional CNNs : **Equivariant** to translation, hard to capture symmetries -> Extensive data augmentation, high computational cost
  - Achieve **invariance** to these symmetries by incorporating **steerable filters** and symmetry groups to reduce the parameter space

•In Euclidean geometry, translation is a geometric transformation that means moving an image or every point in space the same distance in the same direction.

Background

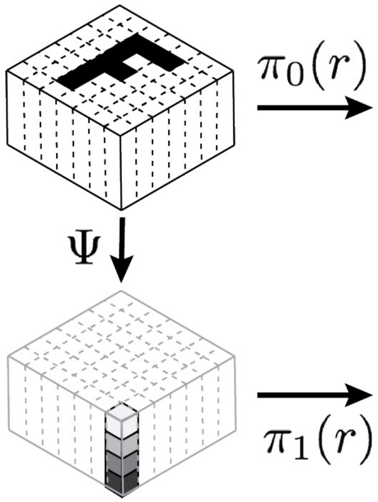
- Equivariant convolutional networks: The representations transform in a predictable linear manner under transformations of the input
- Equivariance: When the input is acted on by the transformation, it also acts on the output.
  -> The application of the transformation can be applied before or after the model's application with no change in overall behaviour.
- Invariance: In whatever way the input is transformed, the output always remains the same.
  -> Useful for object recognition: an input image is rotated, but the output of the filter is the same.



Marcos, Diego & Volpi, Michele & Komodakis, Nikos & Tuia, Devis. (2016). Rotation equivariant vector field networks.

Equivariance

- Map $\boldsymbol{\psi}$: $X \rightarrow Y$, $X \subset \mathbb{R}^d$ and $Y \subset \mathbb{R}^{d1}$
- g, a geometrical transformation belonging to the group G
- $\boldsymbol{\psi}$ is equivariant to G if : $\boldsymbol{\Psi}(\Pi_0(g)\boldsymbol{x}) = \Pi_1(g)\boldsymbol{\Psi}(\boldsymbol{x})$
- $\pi_0(g) : X \rightarrow X'$ and $\pi_1(g): Y \rightarrow Y'$: two linear maps (ex. matrices applied by multiplication) determined by the application of g to x.



- r : rotation of -90° (relace g)
- *$\pi_0(r)$ operates in the domain of X and $\pi_1(g)$ works in the domain of Y .*
- *Commutation:* Applying a transformation and then computing the map produces = calculating the map and then applying the transformation

If $X=\mathbb{R}^2$, 2-D cartesian space, and r is the transformation "clock-wise rotation of 90°", the matrix $\pi_0(r)$ would be equal to a 2x2 Euler matrix with θ=π/2.

$$\begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

Steerable filters

- Let's use rotation to know what steerable filter is.
- $\boldsymbol{\psi}$: $\mathbb{R}^d \rightarrow \mathbb{R}^{d1}$: Convolutional map whose kernel function is k
- Input signal $f(x) \in \mathbb{R}^d$, output signal $f_1(x) \in \mathbb{R}^{d1}$
- $f_1(x) = \boldsymbol{\psi}(f(x))$ -> $f_1(x) = k(x) * f(x)$

- Two conditions to define this filter steerable with respect to rotations:
  - k(x) of each output element can be expressed as a sum of basis functions $\psi_j(x)$, j=1,..M.
  - *The filter's rotation by an arbitrary* angle θ , g_θ, can be expressed in terms of rotations applied to each single basis function.

$$g_\theta(k(x)) = \sum_j^M w_j(\theta)\psi_j(x)$$

We can uniquely orient the filter's response to an input, by modifying the values of w_j

Example of steerable filter

- 2D space, kernel function is a directional derivative of a 2D gaussian
- $k\colon \mathbb{R}^2 \rightarrow \mathbb{R}$ and $x = (x_1, x_2) \in \mathbb{R}^2$

$$k(x_1, x_2) = \frac{\partial(e^{-(x_1^2+x_2^2)})}{\partial x_1} = -2x_1 \cdot e^{-(x_1^2+x_2^2)}$$

$$k_\theta = g_\theta(k(x_1, x_2)) = k(g_\theta^{-1}(x_1, x_2))$$

$$k(g_\theta^{-1}(x_1, x_2)) = k(\begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}^{-1} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}) = k(\begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}^{T} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}) =$$

$$k\begin{pmatrix} cos(\theta)x_1 + sin(\theta)x_2 \\ sin(\theta)x_1 - cos(\theta)x_2 \end{pmatrix} = -2(cos(\theta)x_1 + sin(\theta)x_2) \cdot e^{-(x_1^2+x_2^2)} =$$

$$\underbrace{-sin(\theta)}_{w_1(\theta)} \cdot \underbrace{(-2x_2 \cdot e^{-(x_{11}^2+x_2^2)})}_{\psi_1(x_1,x_2)} \underbrace{-cos(\theta)}_{w_2(\theta)} \cdot \underbrace{(-2x_1 \cdot e^{-(x_1^2+x_2^2)})}_{\psi_2(x_1,x_2)}$$

Power of steerable filters

- linear combination of the convolutions of f with the single basis ψ_1, ψ_2 of k.

$$k_\theta * f = w_1(\theta)(\psi_1 * f) + w_2(\theta)(\psi_2 * f)$$

- We can construct a steerable kernel that ‘steers’ its responses to the orientation of the input.
- When the network encounters data with varying orientations, such as a rotated object in an image, it configures these weights to align the kernel’s responses to the orientation of the input data.
- Enhance the efficiency and outcome with fewer parameters.
- Using steerable properties to handle diverse input orientations

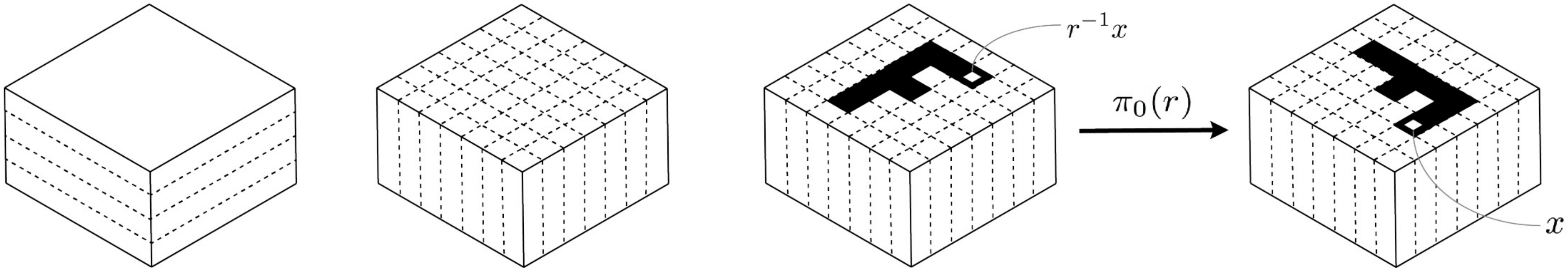
Reduction in Computational cost

- Parameter sharing
  - Traditional CNN : Each filter must learn to detect features at different orientations independently
  - Steerable CNN : Share parameters across different orientations, reducing the number of learnable parameters

- Rotational Equivariance
  - The responses are consistent under rotations of the input data
  - No need for extensive computation at different orientations

Theory: Feature maps and fibers

- 2D signal $f\colon \mathbb{Z}^2 \to \mathbb{R}^K$ with K channels
- Stack of feature maps f_k (k=1,…,K)
- Bundle of fibers F_x at position x : K-dimensional vector space that spans all channels at a given position

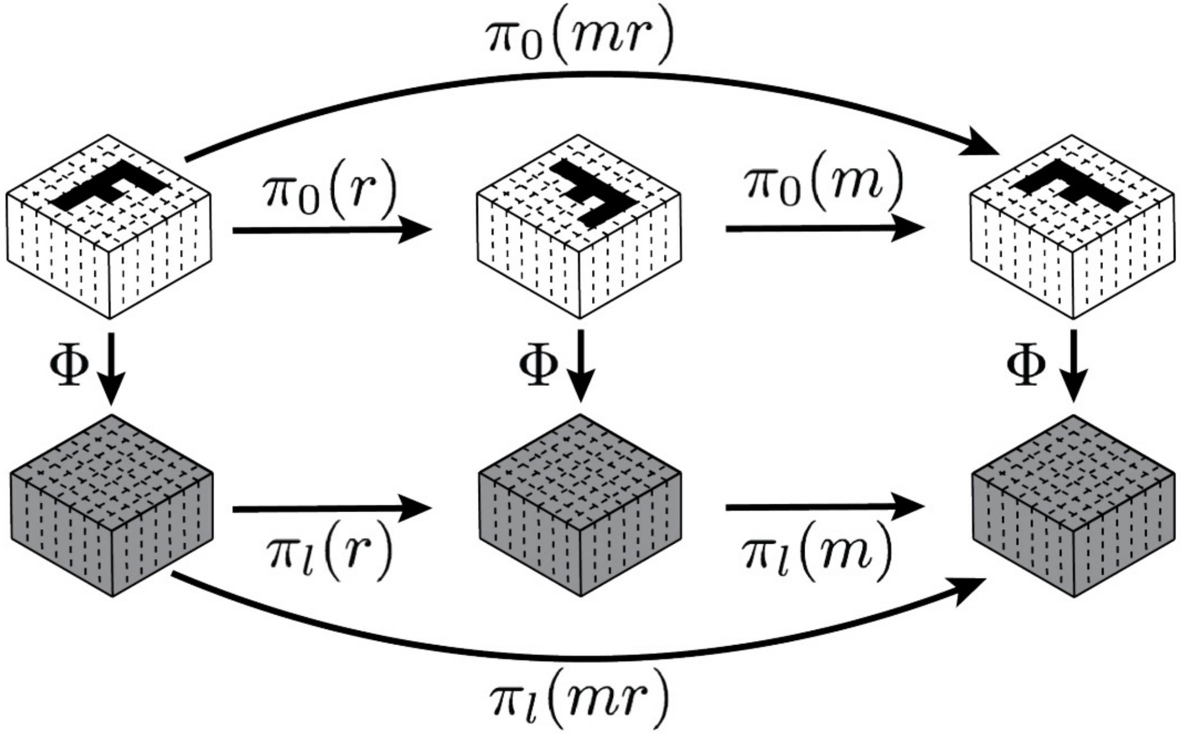


(a) The feature space $\mathcal{F}$ is decomposed into a *stack of feature maps* (left) and a *bundle of fibers* (right).

(b) An image $f \in \mathcal{F}_0$ is rotated by r using $\pi_0(r)$.

$[\pi_0(g)f]\,(x) = f(g^{-1}x)$ The pixel at $g^{-1}x$ gets moved to x by the transformation $g \in G$

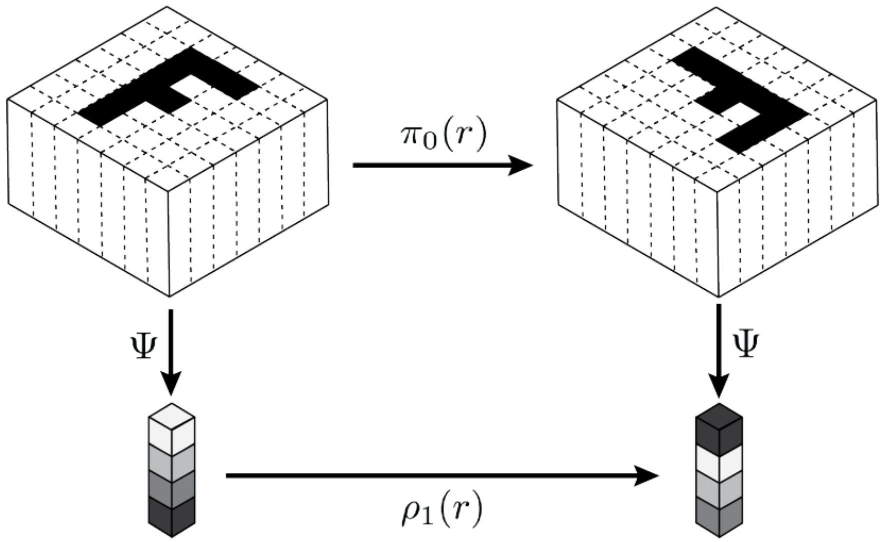
Theory: Steerable representations



- The result of following any path depends only on the beginning and endpoint.
- It is independent of the path itself because of the steerability $\Phi\pi_0(r) = \pi_l(r)\Phi$.

Theory: Equivariant Filter banks

- Let's construct a filter bank that generates H-steerable fibers
- Convolution with this filter bank produces a feature space that is steerable w.r.t G.
- Filter bank Ψ is H-equivariant (*H excludes translation)



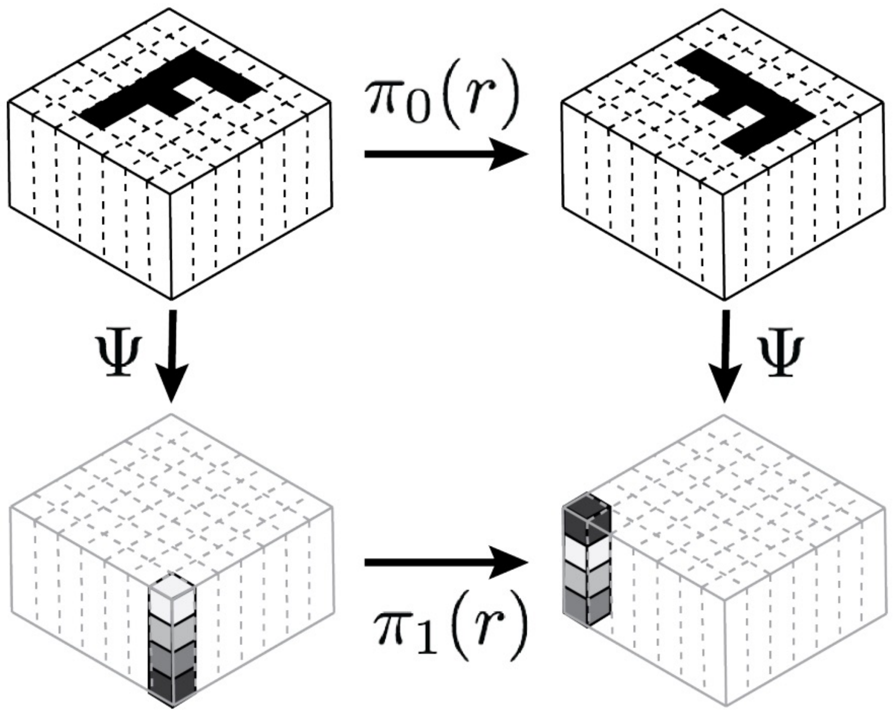
$$\rho(h)\,\Psi = \Psi\,\pi(h), \qquad \forall h \in H$$

ρ_1 represents the 90-degree rotations r by a permutation matrix that cyclically shifts the 4 channels.

$$\Psi = \sum_i \alpha_i \psi_i$$

Theory: Induction

- Let's show how H-steerability of fibers leads to G-steerability of the whole feature space F'
- π' is the representation of G induced by the representations ρ of H



$$\Psi \star \pi(g) f = \pi'(g) \Psi \star f$$

$$[\pi'(tr) f](x) = \rho(r) \left[f((tr)^{-1} x) \right]$$

t: translation, r: rotation

Transport to new location and then acted on by ρ_1.

The representation π_1 induced from the permutation representation ρ_1.

Related Works

- **Equivariant CNNs** (Cohen & Welling, 2016; Dieleman et al., 2016) : Only enforce equivariance to small groups of transformations like rotations by multiples of 90 degrees ; Impractical since the computational cost of this methods scales with the size of the group.
- **Steerability and group representation theory**: Lenz (1989); Koenderink & Van Doorn (1990); Teo (1998); Krajsek & Mester (2007)
- **Equivariant kernels**: Reisert (2008); Skibbe (2013)
- **Invariant and equivariant CNNs**: Gens & Domingos (2014); Kanazawa et al. (2014); Dieleman et al. (2015; 2016); Cohen & Welling (2016); Marcos et al. (2016)

Methods: Image Classification Experiment

- Goal: assess steerability as an inductive bias and compare capsules in small-data scenarios
- Dataset: CIFAR 10
- Baseline Network: 20-layer ResNets tuned for 2k samples
- Steerable CNNs, Replacing convolution layers by steerable convolution layers
- Single-type capsule architectures :30-40% error compared to ResNets (30% error)
- Regular representation capsules : outperforming standard CNNs with 26.75% error. Why? It encompasses all representations, allowing them to learn diverse spatial patterns.
- Mix of capsules (First representation-quotient capsules (regular, qm, qmr2, qmr3) +ReLU, second representation-irreducible capsules (A1, A2, B1, B2, E(2x))+CReLU) : 24.48% error
- Capsules: Specialized convolutional layers that are designed to capture and process specific orientation information.

Results

- When tested on the full CIFAR10 and CIFAR100 dataset, the steerable CNN substantially outperforms the ResNet (He et al., 2016) baseline and achieves state of the art results.

Net	Depth	Width	#Params	#Labels	Dataset	Test error
Ladder	10	96		4k	C10ss	20.4
steer	14	(280, 112)	4.4M	4k	C10	23.66
steer	20	(160, 64)	2.2M	4k	C10	24.56
steer	14	(280, 112)	4.4M	4k	C10+	16.44
steer	20	(160, 64)	2.2M	4k	C10+	16.42
ResNet	1001	16	10.2M	50k	C10+	4.62
Wide	28	160	36.5M	50k	C10+	4.17
Dense	100	2400	27.2M	50k	C10+	3.74
steer	26	(280, 112)	9.1M	50k	C10+	3.74
steer	20	(440, 176)	16.7M	50k	C10+	3.95
steer	14	(400, 160)	9.1M	50k	C10+	**3.65**
ResNet	1001	16	10.2M	50k	C100+	22.71
Wide	28	160	36.5M	50k	C100+	20.50
Dense	100	2400	27.2M	50k	C100+	19.25
steer	20	(280, 112)	6.9M	50k	C100+	19.84
steer	14	(400, 160)	9.1M	50k	C100+	**18.82**

Demonstration of the code

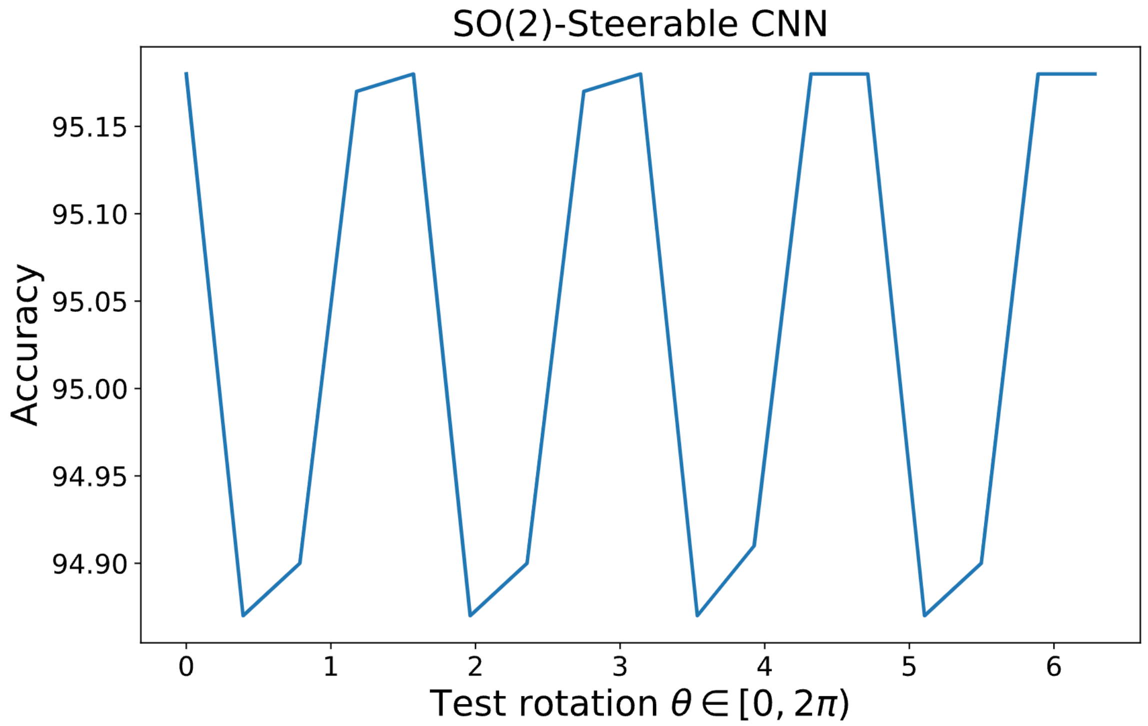
- Dataset: MNIST
- Model: SO(2) equivariant model and C4 equivariant model
- Equivariance Test: Feed N=20 rotated versions of the first image in the test set and print the output logits of the model for each of them.

```
##########################################################################################
angle  |      0      1       2       3       4       5      6       7      8       9
   0.0 : [ 1.856 -0.193  1.68  -1.481  0.48  -2.156 -0.38   0.727  0.41   1.071]
  18.0 : [ 1.858 -0.189  1.659 -1.464  0.457 -2.156 -0.399  0.717  0.427  1.072]
  36.0 : [ 1.856 -0.193  1.675 -1.48   0.486 -2.153 -0.421  0.727  0.43   1.055]
  54.0 : [ 1.872 -0.191  1.676 -1.477  0.5   -2.151 -0.416  0.738  0.432  1.054]
  72.0 : [ 1.875 -0.191  1.672 -1.49   0.5   -2.154 -0.394  0.746  0.407  1.055]
  90.0 : [ 1.856 -0.193  1.68  -1.481  0.48  -2.156 -0.38   0.726  0.41   1.071]
 108.0 : [ 1.858 -0.189  1.659 -1.464  0.457 -2.156 -0.399  0.717  0.427  1.072]
 126.0 : [ 1.856 -0.193  1.675 -1.48   0.486 -2.153 -0.421  0.727  0.43   1.055]
 144.0 : [ 1.872 -0.191  1.676 -1.477  0.5   -2.151 -0.416  0.738  0.432  1.054]
 162.0 : [ 1.875 -0.191  1.672 -1.49   0.5   -2.154 -0.394  0.746  0.407  1.056]
 180.0 : [ 1.856 -0.193  1.68  -1.481  0.48  -2.156 -0.38   0.726  0.41   1.071]
 198.0 : [ 1.858 -0.189  1.659 -1.464  0.457 -2.156 -0.399  0.717  0.427  1.072]
 216.0 : [ 1.856 -0.193  1.675 -1.48   0.486 -2.153 -0.421  0.727  0.43   1.055]
 234.0 : [ 1.872 -0.191  1.676 -1.476  0.5   -2.151 -0.416  0.738  0.432  1.054]
 252.0 : [ 1.875 -0.191  1.672 -1.49   0.5   -2.154 -0.394  0.746  0.407  1.056]
 270.0 : [ 1.856 -0.193  1.68  -1.481  0.48  -2.156 -0.38   0.726  0.41   1.071]
 288.0 : [ 1.858 -0.189  1.659 -1.464  0.457 -2.156 -0.399  0.717  0.427  1.072]
 306.0 : [ 1.856 -0.193  1.675 -1.48   0.486 -2.153 -0.421  0.727  0.43   1.055]
 324.0 : [ 1.872 -0.191  1.676 -1.477  0.5   -2.151 -0.416  0.738  0.432  1.054]
 342.0 : [ 1.875 -0.191  1.672 -1.49   0.5   -2.154 -0.394  0.746  0.407  1.055]
##########################################################################################
```

- The output of the model is already almost invariant but there are small fluctuations in the outputs.
- This is the effect of the discretization artifacts (e.g. the pixel grid can not be perfectly rotated by any angle without interpolation) and can not be completely removed.

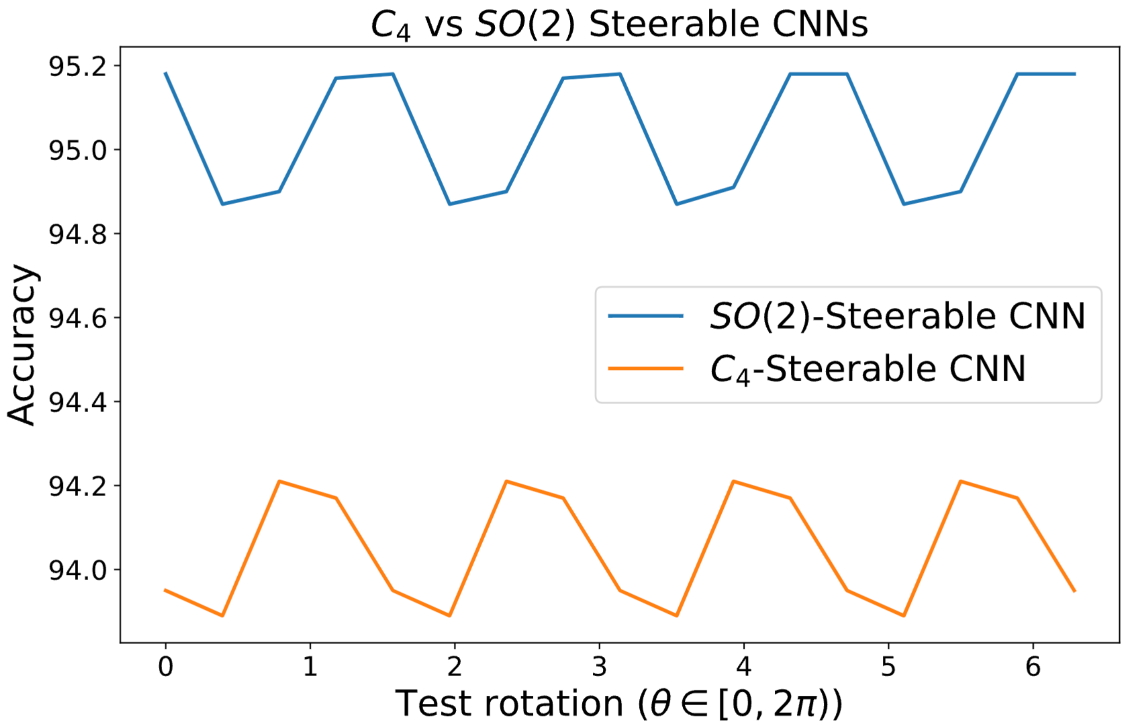
Train the SO(2) equivariant model

```
epoch 0 | test accuracy: 87.82799999999999
epoch 10 | test accuracy: 93.28999999999999
epoch 20 | test accuracy: 94.612
Test accuracy: 94.612
```



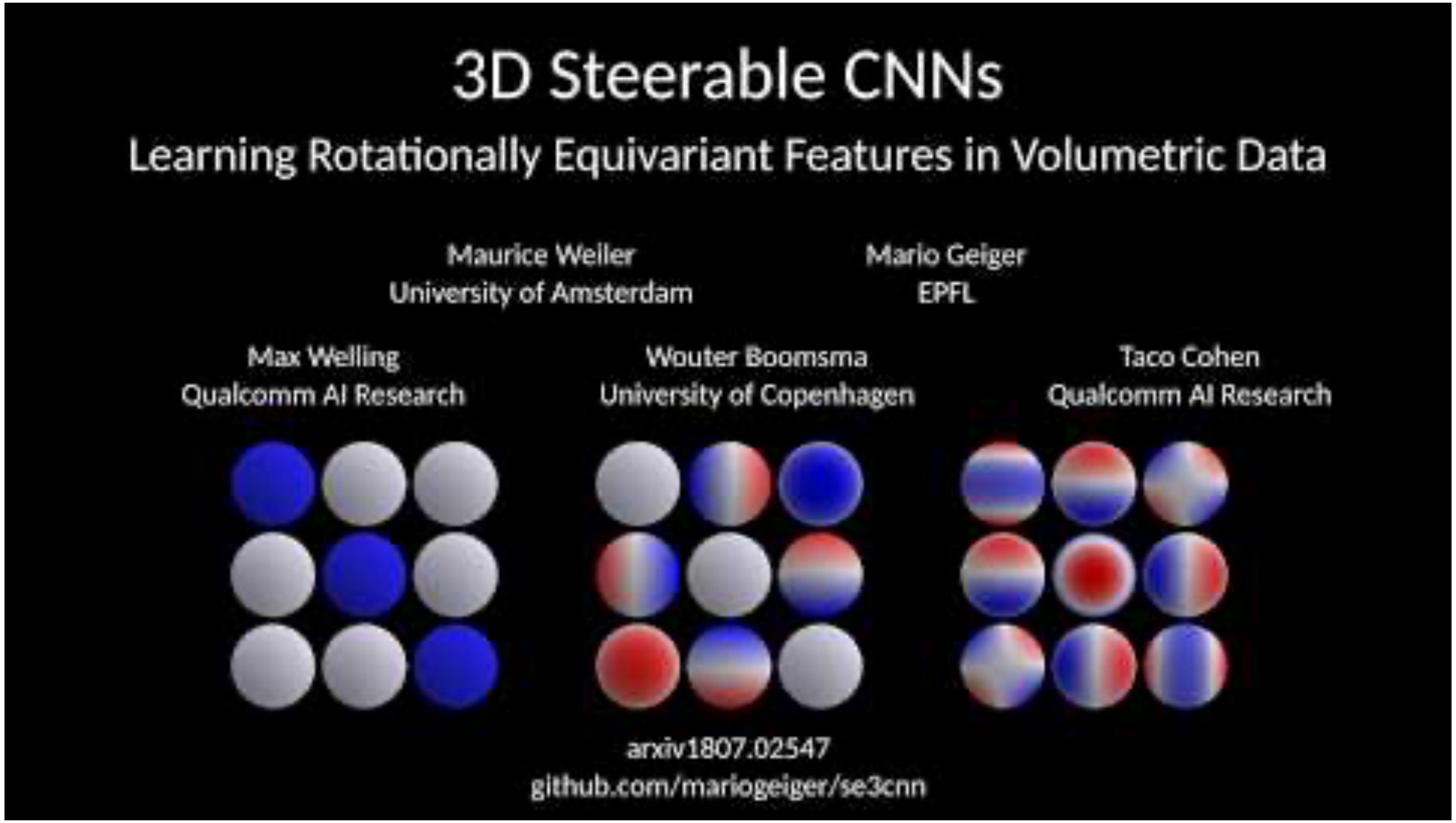
Train the C4 equivariant model

```
epoch 0 | test accuracy: 80.27
epoch 10 | test accuracy: 90.424
epoch 20 | test accuracy: 93.63
Test accuracy: 93.63
```



The accuracy of SO(2) architecture is stable to rotations than C4 architecture.

Application: 3D model classification



https://youtu.be/ENLJACPHSEA

Weiler, Maurice, et al. "3d steerable cnns: Learning rotationally equivariant features in volumetric data." *Advances in Neural Information Processing Systems* 31 (2018).

References

[1] Cohen, Taco S., and Max Welling. "Steerable cnns." (2016).

[2] Marcos, Diego & Volpi, et al. “Rotation equivariant vector field networks.” (2016).

[3] Weiler, Maurice, et al. "3d steerable cnns: Learning rotationally equivariant features in volumetric data." *Advances in Neural Information Processing Systems* 31 (2018).

[4] Ciprian, M. “*A gentle introduction to Steerable Neural Networks*.” Towards Data Science. https://towardsdatascience.com/a-gentle-introduction-to-steerable-neural-networks-part-1-32323d95b03f (2023).

Thank You