Literature Review on :
Equivariance in ConvNet

...

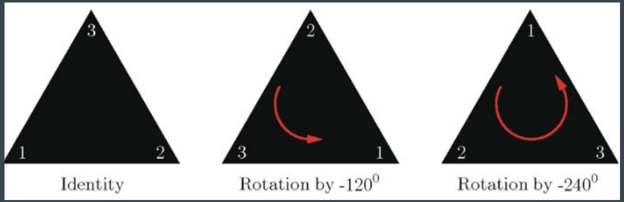
Research Intern
Park Chan Ho

Definitions - Group / Group Symmetry

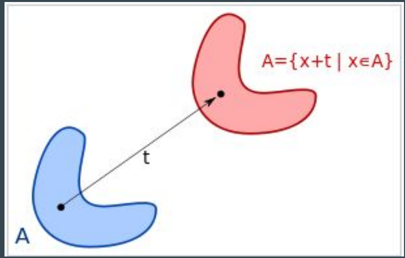
- **Group** is a set G that is associated with a group operator, typically denoted as *, and has the following property
  1) Set is closed under *
  2) * is associative
  3) Every element has an inverse
  4) There exists an identity element

- **We say group symmetry** exists between two objects (x,y) is when there exists a group element g,

  such that: **$g * x = y$**



<Figure 1, rotation group symmetry in triangle>
Rotation group = $\{R_0 , R_{120}, R_{240}\}$



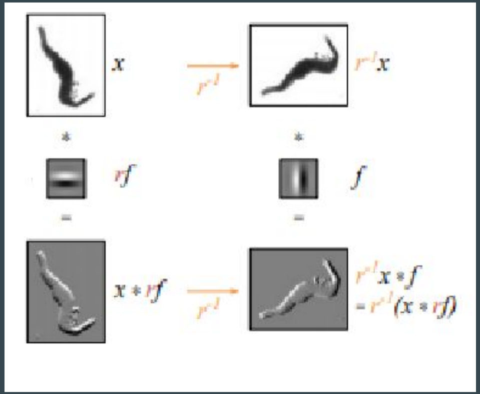
<Figure 2, translation symmetry>
Translation group = $\{(x,y) | x \in Z^2 , y \in Z^2\}$

Definition - Group Equivariance / Invariance

○ **Equivariance** of a representation with respect to a transformation ‘r’ is defined as

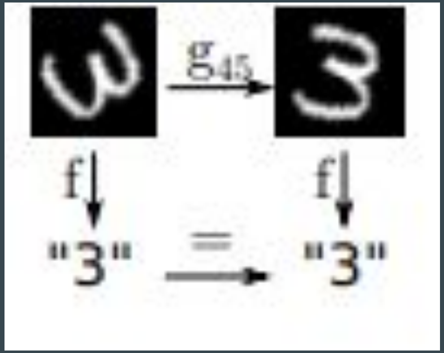
$$\pi_1(r) \circ f(x) = f(\pi_0(r) * x)$$

Equivariance can be seen as the transformation ‘r’ on x can be commuted to the transformation on the output f(x).



<Figure 3, Rotation Equivariance >

○ **Invariance** is a special type of equivariance where $\pi_1(r)$ is the identity map.
Hence it can be summarized as : $f(x) = f(\pi_0(r) * x)$

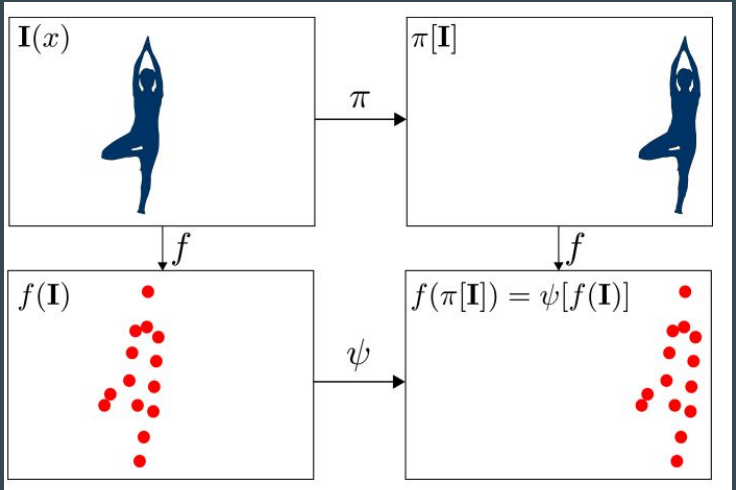


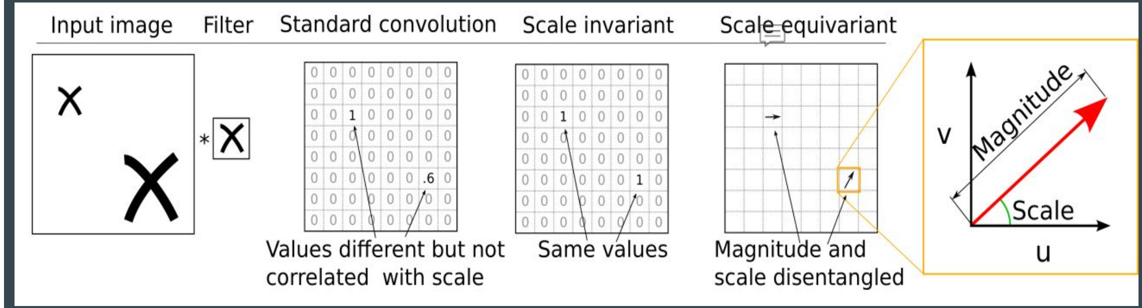
<Figure 4, Invariance>

Why is group equivariance important?

Equivariance is more useful than invariance because it allow us to check if the embeddings are in the right spatial configuration; whether the structure of input is preserved.

⇒ In fact, the **conventional convolution** achieves a group equivariance over **translative group**. Similarly depending on the group, one can design a desired convolution pattern.



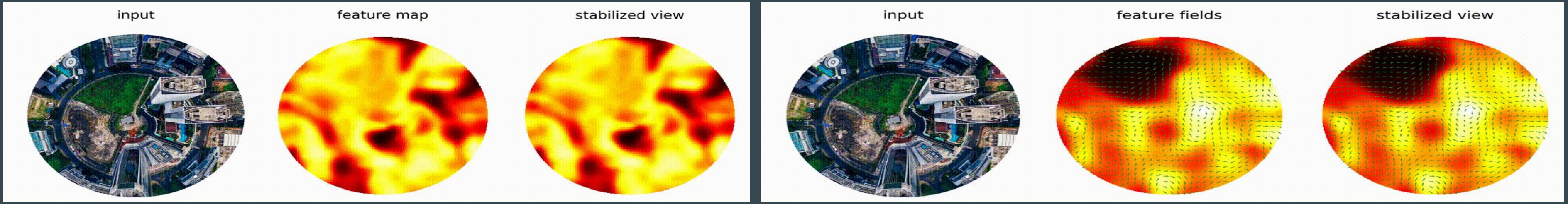


<Figure 5, Translational Equivariance>

<Figure 6, comparison of invariance/ equivariance>

Why is group equivariance important?

Exploits group symmetry on **various input** (spherical, 3D etc) and generates a **more stable** feature map



<Figure 7: Comparison of feature map of Conventional CNN and Rotation Group Equivariant CNN>

High expressive capacity compared to number of parameter; this is because group convolution leads to more weight sharing than the traditional CNN convolution.

Goal of presentation

i) Categorize each equivariant network by category (group, scale, rotation equivariance)

ii) Understand how equivariance is constructed (implicit, explicit)

iii) Limitation of each model

iv) Propose a new direction for improvement

Group Equivariant CNN (2016)

- **Aim of model**

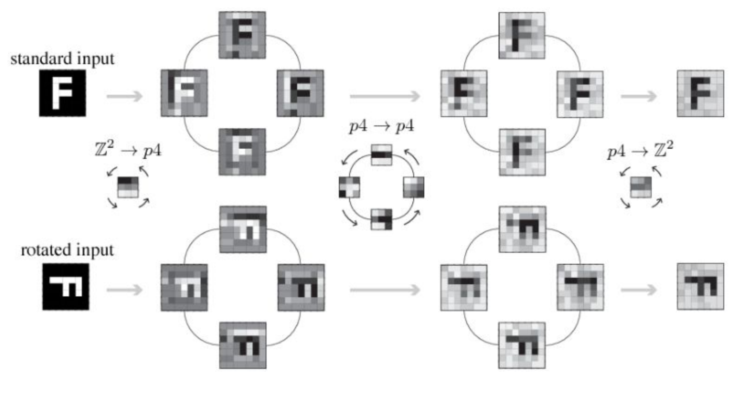
  Construct representations of images that have structure of G-equivariance, for some chosen group G.

  Example: p4 group - 90 degree rotation in input image leads to 90 degree rotation in representation

- **General Flow of Algorithm:**

  Input image ⇨ Translation of filter ⇨ expansion and transformation of filters based on group ⇨ convolution ⇨ next layer

-



<Figure 8: Rotation Equivariance in Group Equivariant CNN>

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

<P4 group representation>

Group Equivariant CNN

- **How it achieves group equivariance**:

  The method below states the G-convolution. The definition of the G-convolution creates an effect of augmenting transformed images while keeping the intra-class structure.

$$\underbrace{[[L_u f]}_{\text{rotation on input}} \star \psi](g) = \underbrace{[L_u[f}_{\text{rotation on feature map}} \star \psi]](g)$$

Lu = rotation

f = previous feature map

φ = kernel

★ = convolution

- **Limitation**

1. Only applicable to discrete groups and as size of group increases the computation rapidly.

2. Kernel offset is not defined; hence it is not invariant to local transformations.

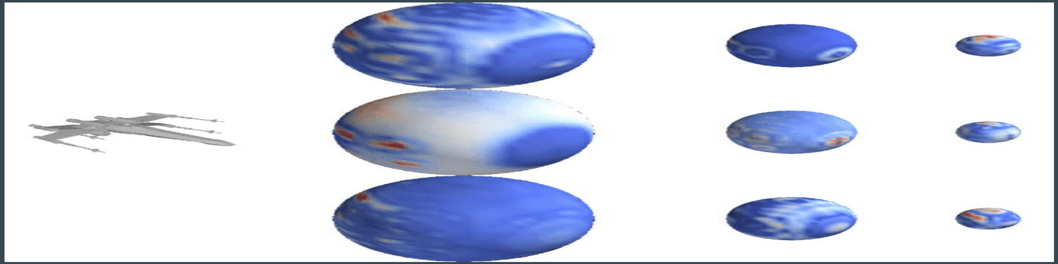
Spherical CNNs (Rotation Equivariance)

- **Aim of model**

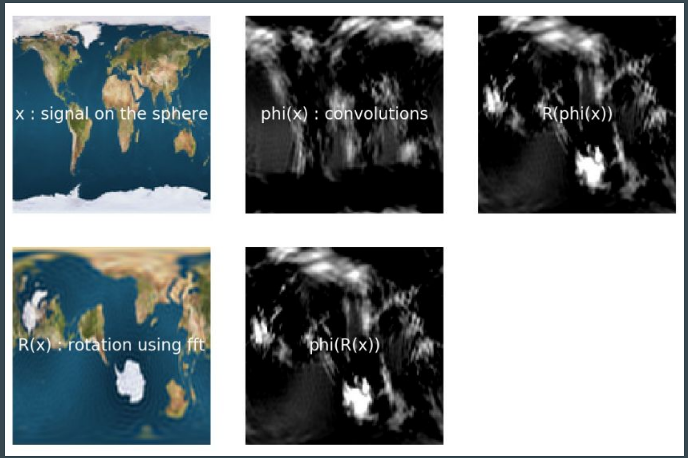
Develop a convolutional network on spherical signals by SO(3) group convolution.
This is important because spherical signals projected to a 2D plane will be distorted.

- **General Flow of Algorithm:**

Input signal ⇨ Project signal to sphere ⇨ Spherical convolution ⇨ Next layer
(3D or spherical signal on 2D)



<Figure 9, 3D rotational equivariance of Spherical CNN>



<Figure 10, Equivariance on spherical signal>

Spherical CNNs (Rotation Equivariance)

- **How it achieves rotational equivariance:**

  The formula below states the spherical convolution. The definition of convolution creates an effect augments rotated images while keeping the intra-class structure.

$$[\psi \star \overbrace{[L_Q f]}^{\text{Rotation Q on prev feature map}}](R) = \underbrace{\langle L_R\psi, L_Q f\rangle = \langle L_{Q^{-1}R}\psi, f\rangle = [\psi \star f](Q^{-1}R)}_{\text{definition of convolution}} = \overbrace{[L_Q[\psi \star f]]}^{\text{Rotation Q on output}}(R).$$

L_Q = arbitrary rotation on sphere

φ = kernel

★ = convolution

f = previous feature map

- **Limitation**

  1. Computationally expensive to calculate the spherical convolution

  2. Other symmetries like scale or affine are not exploited.

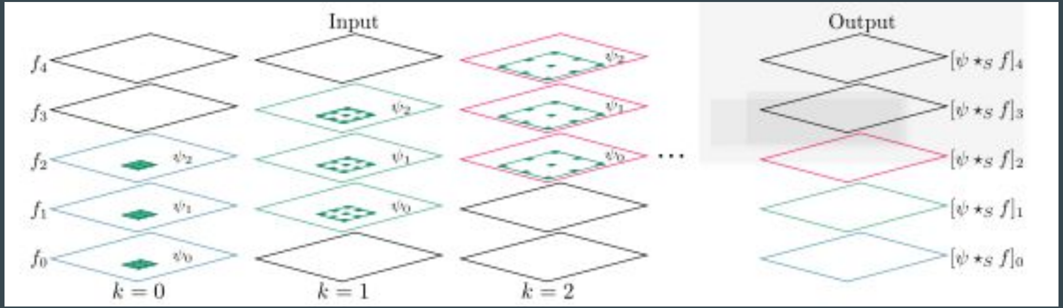
Deep Scale Space (Scale Equivariance)

- **Aim of model**
  - Achieve scale equivariance through defining scale space and scale convolution.
  - Adds another dimension(“lift”) to the input through bandlimited image.

- **General Flow of Algorithm:**

Depending on the scale level considered, kernel will **dilate** according to the scale-space level before convolution.

Input image ➩ Calculated “lifted” signal through gaussian kernel ➩ scale correlation with dilated kernel ➩ next layer



<Figure 11, Convolution in DSS>

Deep Scale Space (Scale Equivariance)

- **How it achieves scale equivariance:**

  Scale equivariant cross-correlation in each layer; this is based on explicit structure of network

Scale operation 't' on previous feature map

Scale operation 't' on output feature map

$$[\psi \star \underline{L_t[f]}](s) = \sum_{x \in X} \psi(x) L_s[L_t[f]](x) = \sum_{x \in X} \psi(x) L_{st}[f](x) = [\psi \star f](st) = \underline{L_t[\psi \star f]}(s)$$

t = arbitrary element from semi-group S

f = map to previous feature map

φ = kernel

★ = convolution

Limitation

1. Dilation of kernel is fixed by the integer scaling factor.
2. Performance may depend on the parameterization of bandlimitting kernel at first.

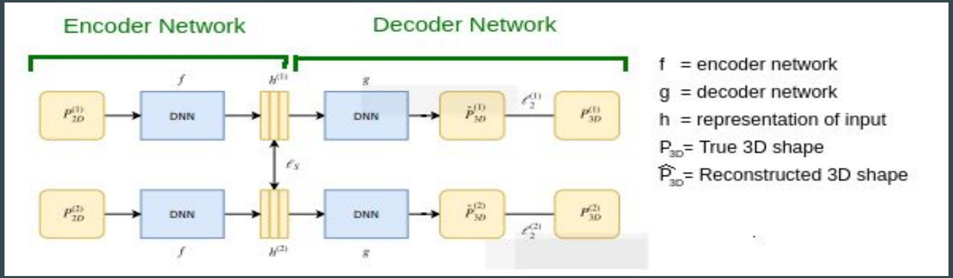
Siamese Equivariant Network(Rotation Equivariance)

- **Aim of model**
  - Develop an autoencoder that learns a geometrically interpretable embedding of human pose.
  - The embedding is rotationally equivariant; so this makes the network robust to new camera views.

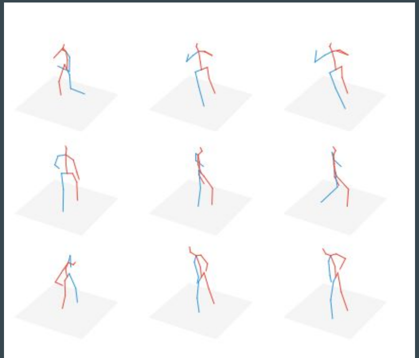
- **General Flow of Algorithm:**

Training data consists images of 3D object projection onto 2D plane from various viewpoints.

Input image pair ⇨ Encoder ⇨ Decoder ⇨ Reconstruction



<Figure 12, Structure of Siamese Autoencoder>



<Figure 13, Generated examples>

Siamese Equivariant Embedding(Rotational Equivariance)

- **How it achieves pose equivariance:**

Siamese Network achieves equivariance through regularization.; the implicit structure.
Setting regularization as equivariance error allows continuous rotation group to be learnt

$$\ell_S = \left|\underbrace{\left\|R_2 R_1^{-1} h_1 - h_2\right\|}_{\text{Regularization on representations to be rotationally equivariant}} - \lambda_1 \underbrace{\left\|P^{(1)}_{3DA} - P^{(2)}_{3DA}\right\|}_{\text{Regularization on decoder network produces the same output as input}}\right|^2 .$$

R1, R2 = linear representation of rotation associated with inputs
P_{3DA} = reconstructed 3D pointcloud
h1, h2 = representation of inputs
λ = regularizing parameter

- **Limitations**

1. The model is specifically designed for finding equivariant representation for one class (human)
2. Large number of parameters as there is less weight sharing than group convolution.

Scale-equivariant steerable convNet(SESN)

- **Aim of model**

  Develop building blocks and define scale group convolution that preserves scale-equivariance.

  Scale convolution is a group convolution based on the scale-translation group. H={(s,t): s ∈ scales, t∈ Z^2}

- **General Flow of Algorithm:**

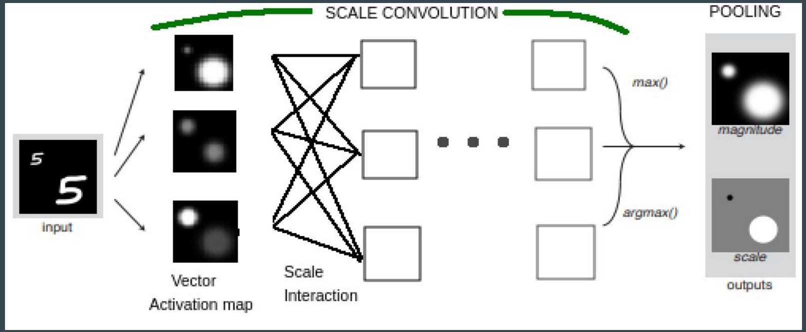
Kernels are scale steerable function; a linear combination of steerable basis functions.

Calculate steerable basis functions ➪ Input Image ➪ Scale convolution(➪ Scale, magnitude pooling) ➪ Next layer

-

$$L_s[f] \star \psi_\sigma = L_s[f \star \psi_{s^{-1}\sigma}]$$

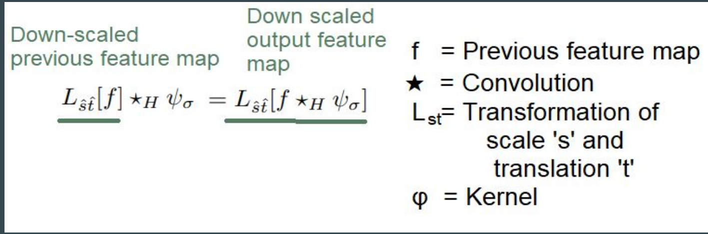
<Figure 13, Steerable filter of SESN>



<Figure 14, Flow of SESN>

Scale-equivariant steerable convNet(SESN)

- **How it achieves scale equivariance:** scale equivariance in each layer



- **Advantage compared to Deep Scale Space**

1. No use of gaussian kernel for bandlimiting; an assumption of creating scale space.

2. Not restricted to discrete integer scale factor.

- **Limitation**
-

1. If there are scales changes by large magnitude, it known to underperform

2. There are no offsets for the kernel; may not be invariant to local transformation .

Summary

Year	Model	Equivariance				Description/ How	Offset of kernel
		Rotation	Scale	Group	Continuous?		
2016	Group Convnets	✓	✗	✓	✗	Group equivariance through p4 group convolution/pooling	Kernel shape is fixed / rotation action applied
2017	Harmonic Networks	✓	✗	✗	-	Patchwise equivariance through circular harmonic filters (similar to SO(2) group)	Dependent on a function (radial profile) / rotation action applied
2018	Siamese Network[7]	✓	✗	✓	✓	Equivariant embedding of 3D pose through paired image training; pairs of same class but rotated pose	Kernel shape is fixed
2019	Deep Scale space	✗	✓	✓ (semi)	✗	Scale-space through bandlimiting kernel. Feature map and filter are transformed through semi-group $H = \{(A,z): A \in \text{discrete dilation}, z \in \text{shift}\}$	Kernel is dilated dependent on the dilation factor(A)
2019	E(2)-equivariant steerable convNet[8]	✓	✗	✓	✗	O(2) Group-equivariance and group restriction on last few layers achieves state of art result	Kernel shape is fixed / O(2) subgroup actions applied

Summary

Year	Model	Equivariance				Description/ How	Offset of kernel
		Rotation	Scale	Group	Continuous?		
2019	Self-supervised Scale Equivariant Network for Weakly Supervised Semantic Segmentation	✗	✓	✗	-	- Implicit construction of equivariance through regularization. - Multi-scale feature extraction using multiple convNets to predict segmentation map. - While forcing equivariance through regularization, the features are still effective as it was trained on predicting segmentation map.	Kernel fixed on all branches of the convNet
2019	SESN (Scale-Equivariant steerable convNet)	✗	✓	✓	✓	Scale equivariance through steerable kernels and group convolution on H={(s,t): s $\in$ scales, t $\in$ Z^2}	Depends on the scale element applied to steerable filter

Improvement for better scale equivariant model?

- Shape of the kernel is fixed / changing with a fixed pattern corresponding to group

  ⇒ Not invariant to local transformation

  ⇒ Combine with modules that has kernel changing shape

Deformable Convolutional Network (DCN, 2017)

○ **Problem:**

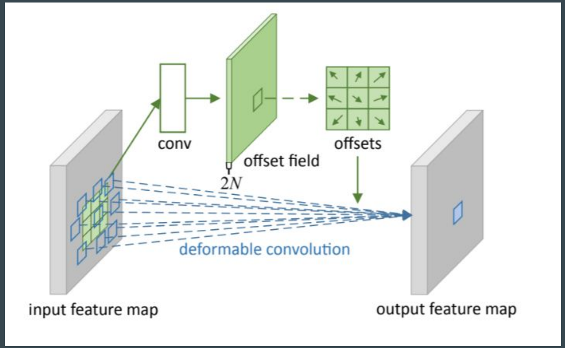
- Conventional CNN or models discussed before have **fixed kernel** shape or transforming depending on the group the kernel is associated with.

⇒ lacks internal mechanism to handle the **local geometric transformation**

○ **Suggested solution:**

- Offset learning through additional convolution branch

- This allows receptive fields to be effective as shown in Figure 16.



<Figure 15, convolution of DCN>

Figure 16, receptive field of DCN[9]:
Green point indicates the position of the kernel

Deformable Convolutional Network (DCN, 2017)

○ Deformable Convolution

$$[[f \circ \phi] \star \varphi](x) = \sum_{y \in R} \sum_{i=1}^{K^l} \varphi_i(y) \cdot f_i(x + y + \phi(y))$$

φ = kernel
Φ = offset function
f = previous feature map
K^l = # of feature maps in 'l'th layer
R = kernel grid

○ Limitation

1. Symmetries are not being exploited.
2. Embeddings will not be equivariant to scale or rotation, which implies lack of internal structure of feature space.
3. Depending on the number of object class, model becomes complex rapidly due to number of parameters

Direction of new methodology

Aim

1. Scale equivariant representation of images

2. Offset learning based on the object class

Motivation

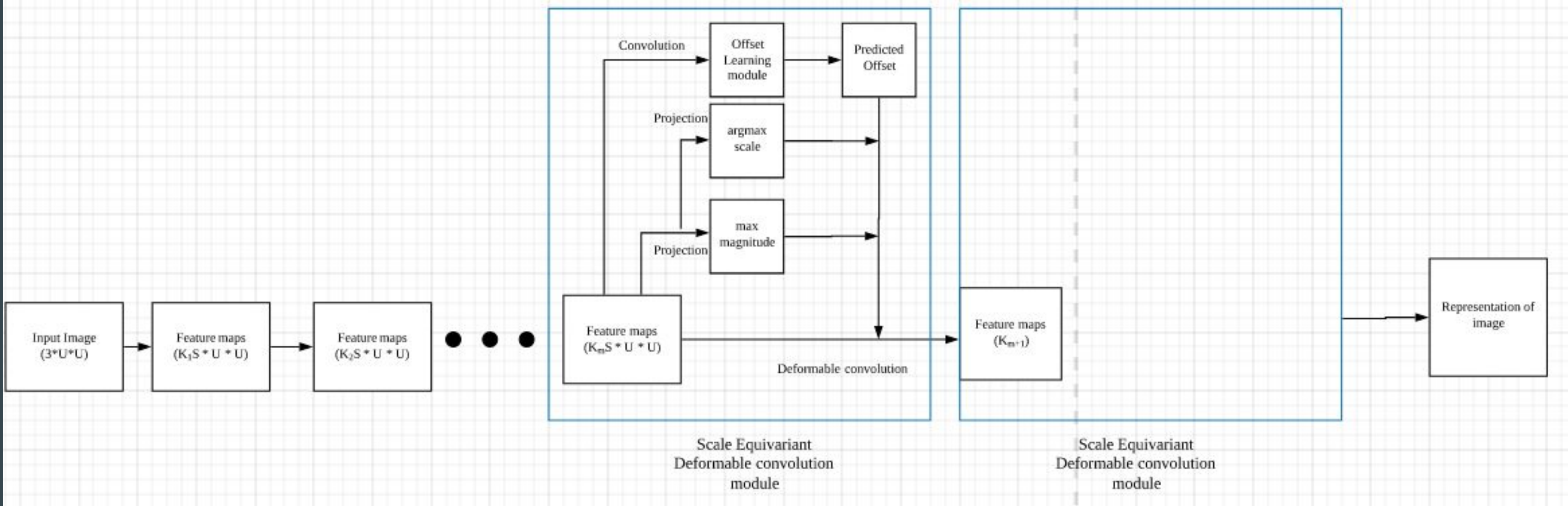
1. Deformable convolution is applied to the last few convolutional layers. This is because if the deformable convolution is applied to lower or middle layers, the spatial structures are susceptible to fluctuations.[9]

2. SESN applies scale projection at the end of convolutional block. This is because the representation of projection is invariant to input variation towards the end of the network.[10]

Proposed Methodology

DCN v2 scales the offset learnt by a learnable modulation factor. Instead of modulation factor, we replace this with the scale achieved from scale projection.

Architecture: We keep SESN as our backbone structure and replace the last few layers with the new deformable convolution layer.



Challenges 1. Maintaining scale equivariance with the offset learning module

2. Computational complexity when number of object class is large

Equivariance in new method

$[L_{st}[g] \star \varphi](x) = L_{st}[g \star \varphi](x)$

$g(s, x) = f(s, x + \Delta x)$ where Δx is the offset learnt by the offset convolution layer

$\Delta x = \phi(\Psi(f(s, x)))$ where Ψ is the separate steerable convolution on the feature map

$\Psi(f(s, x))$ is the offset learnt by the deformable layer

ϕ is offset adjustment based on the scale, magnitude of object on x

f = previous feature map

$\Delta x = \phi(\Psi(f(x)))$

$\star$ = Convolution

φ = Kernel

L_{st} = Scale group action on feature map (scale by 's' translation by 't')

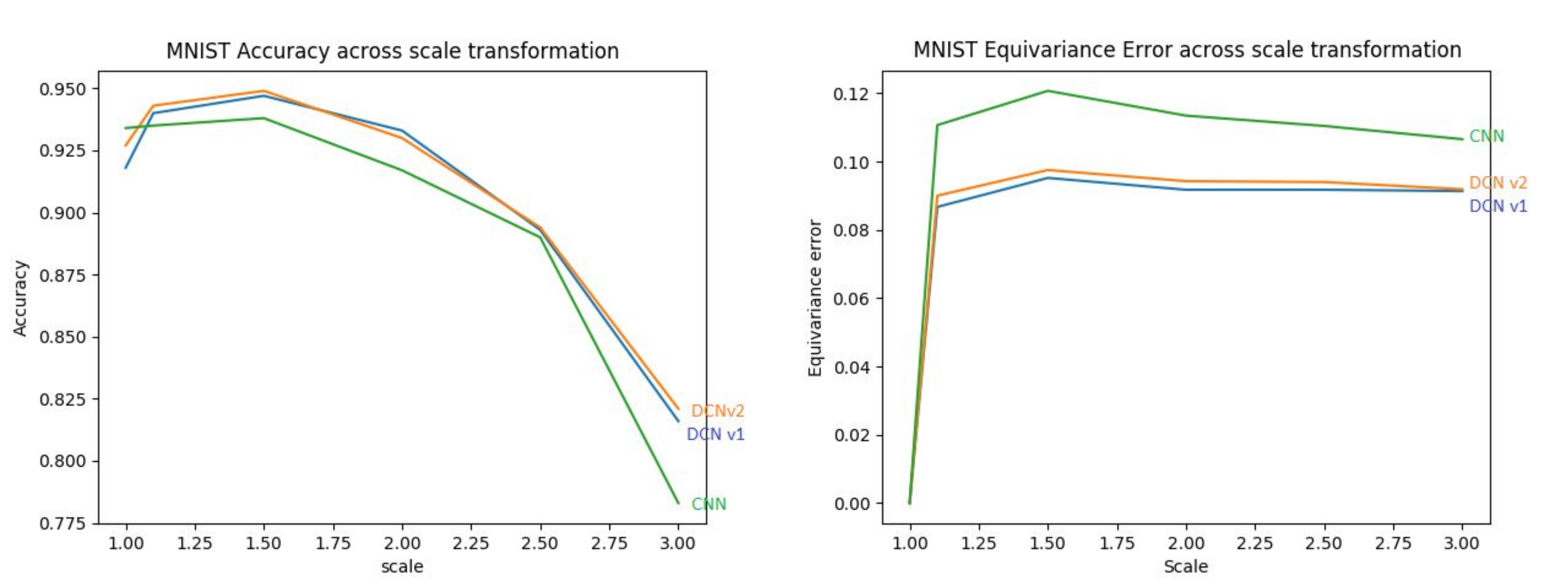
Attached in appendix

1. Proof of preservation of steerability in kernel
2. Proof of group action onto feature map
3. Proof of equivariance of representations

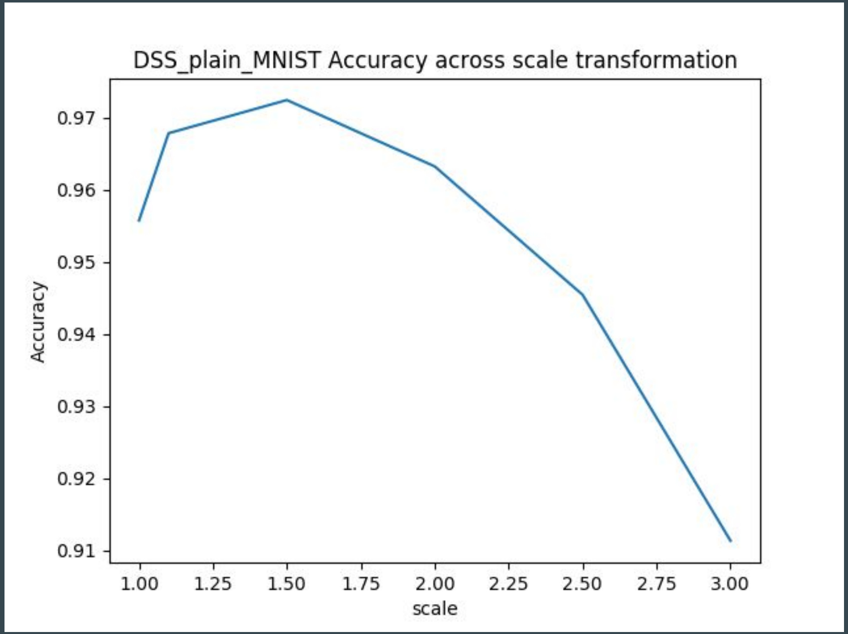
Plan of action

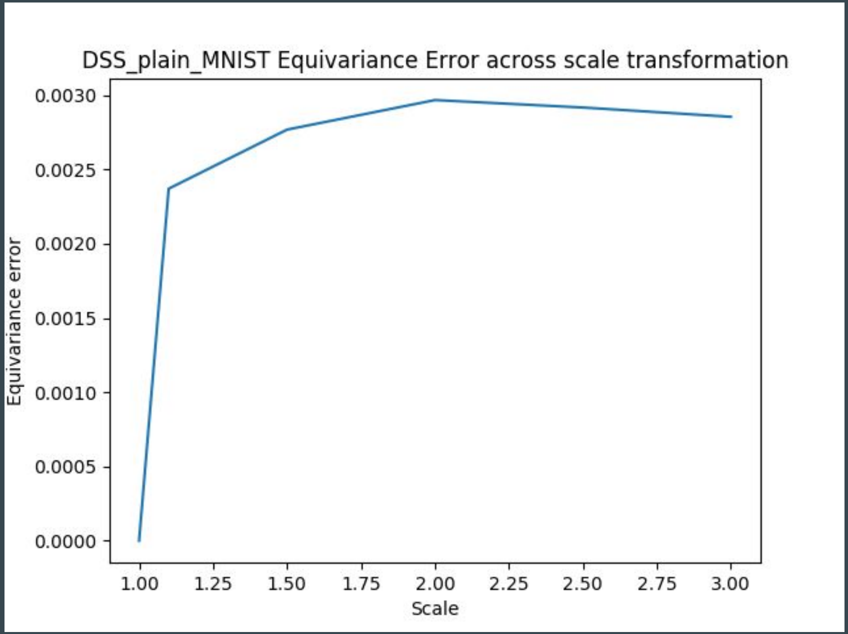
	How?	Why?
Check scale equivariance in existing models	Take random scale MNIST images and apply random scale symmetric action and calculate the equivariance error $\| L_s\Phi(f) - \Phi L_s(f) \|^2$	Check equivariance in deformable convolutional network
Implement new method	Check the convergence of the learning	Check stability of architecture
	Hyper-parameter tuning	Check performance of model
	Plot receptive fields at different points	Check the accuracy of offset learning
	Check the equivariance error $\| L_s\Phi(f) - \Phi L_s(f) \|^2$	Ensure equivariance
	Compare time performance with existing models	Check efficiency

Experiment on Scale MNIST -DCN

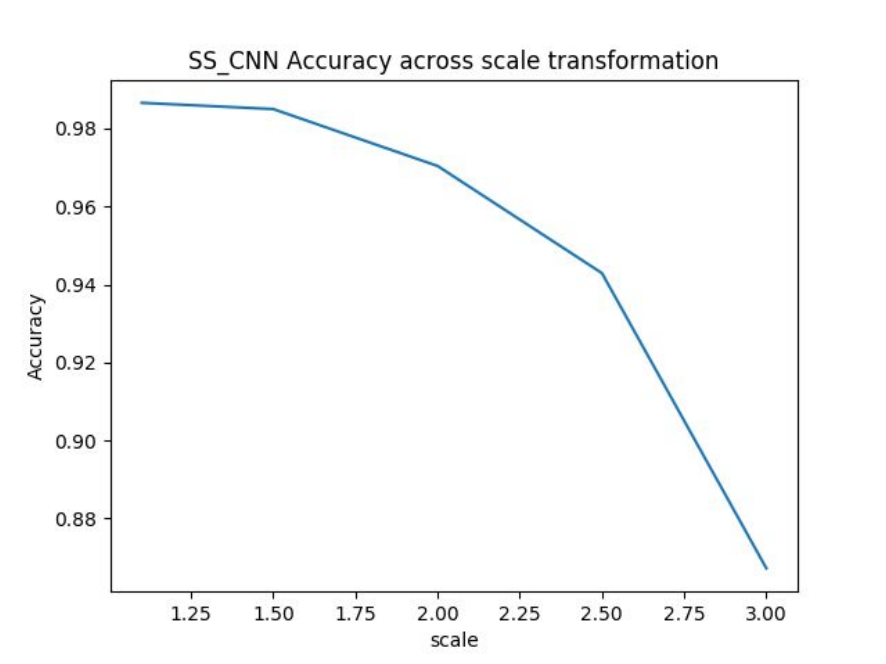


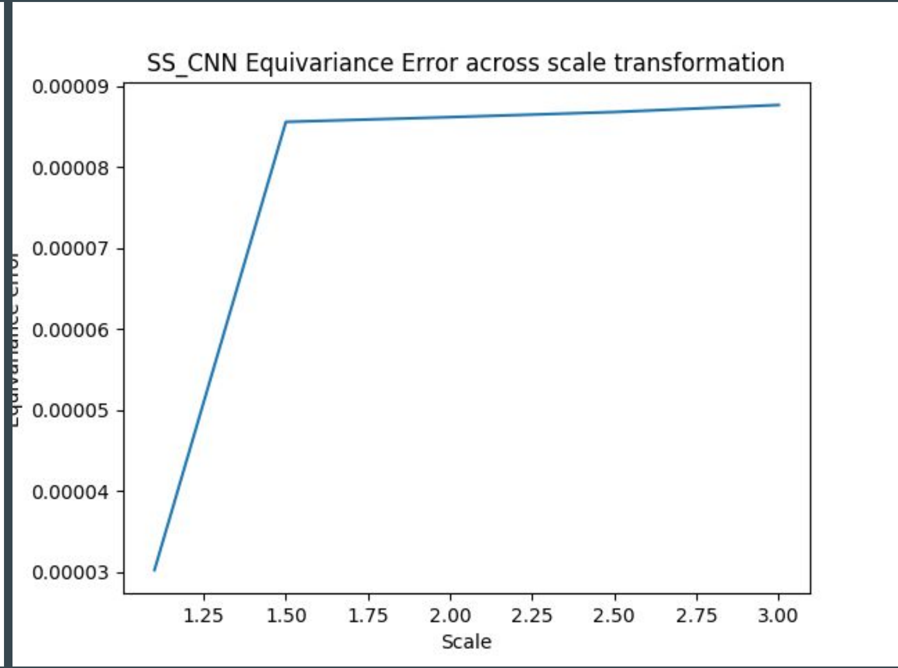
Experiments on Scale MNIST - DSS





Experiment on Scale MNIST - SS_CNN





Conclusion

Appendix: proof

$g(s, x) = f(s, x + \Delta x)$ where Δx is the offset learnt by the offset convolution layer

$\Delta x = \phi(\Psi(f(s, x)))$ where Ψ is the separate steerable convolution on the feature map

$\Psi(f(s, x))$ is the offset learnt by the deformable layer

ϕ is offset adjustment based on the scale, magnitude of object on x

$$\phi(\Psi(f(s, x))) = proj_1 f(s, x) \cdot \Psi(f(s, x))$$

$$proj_1 f(s, x) = max_s f(s, x)$$

Remark $s_2^{-1}\Delta x = \Delta s_2^{-1} x$ (i.e. $s_2^{-1}\phi(\Psi(f(s, x))) = \phi(\Psi(f(s_2^{-1}s, s_2^{-1}x))))$)

Proof.

$$s_2^{-1}\Delta x = s_2^{-1}\phi(\Psi(f(s, x))) = s_2^{-1} \cdot proj_1 f(s, x) \cdot \Psi(f(s, x))$$

$$= s_2^{-1} \cdot proj_1 f(s, x) \cdot s_2 \cdot \Psi_{s_2}(f(s_2^{-1}s, s_2^{-1}x))(\because \Psi \text{ is a steerable filter})$$

$$= proj_1 f(s, x) \cdot \Psi_{s_2}(f(s_2^{-1}s, s_2^{-1}x))$$

$$= ???\ proj_1 f(s_2^{-1}s, s_2^{-1}x) \cdot \Psi_{s_2}(f(s_2^{-1}s, s_2^{-1}x))\ (\because proj_1 = max_s = max_{s_2^{-1}s})?$$

$$= \phi(\Psi_{s_2}(f(s_2^{-1}s, s_2^{-1}x))))$$

$$= \Delta s_2^{-1} x$$

A. Proof of $L_{s_2}[g](s, x) = g(s_2^{-1}s, s_2^{-1}x)$

$$L_{s_2}[g](s, x) = L_{s_2}[f(s, x + \Delta x)]$$

$$= f(s_2^{-1}s, s_2^{-1}(x + \Delta x))$$

$$= f(s_2^{-1}s, s_2^{-1}x + s_2^{-1}\Delta x)$$

$$= f(s_2^{-1}s, s_2^{-1}x + \Delta s_2^{-1}x)(\because s_2^{-1}\Delta x = \Delta s_2^{-1}x)$$

$$= g(s_2^{-1}s, s_2^{-1}x)$$

.

Appendix: proof

B. Proof of $L_t[g](s,x) = g(s, x-t)$

$$L_t[g](s,x) = L_t[f(s, x+\Delta x)]$$

$$= f(s, x - t + \Delta x)$$

$$= f(s, (x-t) + \Delta(x-t))) \ (\because ???)$$

$$= g(s, x-t)$$

- Proof of preservation of steerability of kernel on feature map $g(x)$

1. $$[L_t[g] \star \varphi] = L_t[g \star \varphi]$$

$$[L_t[g] \star \varphi](x) = \int_{\mathbb{R}} L_t[g](x')\varphi(x'-x)dx'$$

$$= \int_{\mathbb{R}} g(x'-t)\varphi_(x'-x)dx'$$

$$= \int_{\mathbb{R}} g(x'')\varphi(x''+t-x)dx'' \ (\because \text{Change in variable } x'' = x'-t)$$

$$= \int_{\mathbb{R}} g(x'')\varphi(x'-(x-t))dx''$$

$$= L_t[g \star \varphi]$$

Appendix: proof

2. $[L_s[g] \star \varphi] = L_s[g \star \varphi]$

$$[L_s[g] \star \varphi](x) = \int_{\mathbb{R}} L_s[g](x')\varphi(x'-x)dx'$$

$$= \int_{\mathbb{R}} g(s^{-1}x')\varphi(x'-x)dx'$$

$$= \int_{\mathbb{R}} g(x'')\varphi(sx''-x)sdx'' \text{ } (\because \text{Change in variable } x'' = s^{-1}x' \Rightarrow dx' = sdx'')$$

$$= s\int_{\mathbb{R}} g(x'')\varphi(s(x''-s^{-1}x))dx''$$

$$= s\int_{\mathbb{R}} g(x'')L_{s^{-1}}[\varphi](x''-s^{-1}x)dx''$$

$$= sL_s[g \star L_{s^{-1}}[\varphi]](x)$$

$$= sL_s[g \star s^{-1}[\varphi_{s^{-1}}]](x) \text{ } (\because \text{ steerable filter } L_{s^{-1}}[\varphi](x) = \varphi(sx) = s^{-1}\varphi_{s^{-1}}(x))$$

$$= L_s[g \star \varphi_{s^{-1}}]$$

- Translation Equivariance

$$[L_{t'}[g] \star \varphi_\sigma](s,t) = \sum_{s'}[L_{t'}[g](s',\cdot) \star \varphi_{s\sigma}(s^{-1}s',\cdot)](t) \text{ } (\because \text{definition of scale convolution})$$

$$= \sum_{s'} L_{t'}[g(s',\cdot) \star \varphi_{s\sigma}(s^{-1}s',\cdot)](t) \text{ } (\because \text{scale steerable filter } \varphi)$$

$$= L_{t'}\sum_{s'}[g(s',\cdot) \star \varphi_{s\sigma}(s^{-1}s')](t)$$

$$= L_{t'}[g \star \varphi_\sigma](s,t)$$

- Scale Equivariance

$$[L_{s^*}[g] \star \varphi_\sigma](s,t) = \sum_{s'}[L_{s^*}[g](s',\cdot) \star \varphi_{s\sigma}(s^{-1}s',\cdot)](s,t) \text{ } (\because \text{definition of scale convolution})$$

$$= \sum_{s'} L_{s^*}[g(s^{*-1}s',\cdot) \star \varphi_{s^{*-1}s\sigma}(s^{-1}s',\cdot)](t) \text{ } (\because \text{scale steerable filter } \varphi)$$

$$= \sum_{s''}[g(s'',\cdot) \star \varphi_{s^{*-1}s\sigma}(s^*s^{-1}s''),\cdot](s^{*-1}t) \text{ } (\because \text{1.Change of variable } s'' = s^*$$

$$\text{2. } L_{s*} \text{ applied on feature map and kernel } \varphi)$$

$$= [f \star \varphi_\sigma](s^{*-1}s, s^{*-1}t)$$

$$= L_{s^*}[f \star \varphi_\sigma](s,t)$$

References

1. Ivan Sosnovik, Michal Szmaja: Scale-equivariant steerable network

2. Daniel E Worrall and Max Welling. Deep scale-spaces: Equivariance over scale

3. Diego Marcos, Michele Volpi : Rotation Equivariant vector field networks

4. Daniel E Worrall, Stephan J Garbin. Harmonic networks: Deep translation and rotation equivariance.

5. Taco Cohen and Max Welling. Group equivariant convolutional networks

6. Taco Cohen, Mario Geiger, Jonas Kohler, and Max Welling. Convolutional networks for spherical signals.

7. Véges, M., Varga, V., and Lörincz, A. (2018). 3D human pose estimation with siamese equivariant embedding

8. Maurice Weiler, Gabriele Cesa: General E(2)- Equivariant steerable CNNs

9. Jifeng Dai, Haozhi Qi: Deformable Convolutional Networks

Analysis of object detectors

Year	Model Name	Description	Scale-inv	Rot-inv	Equivariance
2014	SPP-net	Single stage detector with single scale feature map (Spatial Pyramid Pooling, multi-level spatial bins)	✓	✗	✗
2015	SSD (Single Shot Detector)	Single stage detector with multi-scale feature pyramid predictions	✓	✗	✗
2015	Faster-RCNN (Fast-RCNN + RPN)	Double stage detector with single Scale feature map (anchor boxes)	✓	✗	✗
2016	FPN (Feature Pyramid Network)	Single Stage detector with Multi-scale feature pyramid predictions (bidirectional & lateral)	✓	✗	✗
2017	RetinaNet	Single stage detector with multi-scale feature map (FPN + anchor boxes)	✓	✗	✗
2017	STN (Spatial Transformer Network)	Spatial Transform module performs affine transformation on feature map	✓	✓	✗

Representation Learning - 1

Understanding image representation by measuring their equivariance and equivalence (K.Lenc ,2015)

- By assuming a group action to be a linear transformation on the feature space, the paper introduces a way to generate a sparse matrix representation of the group; the loss function constructs a group equivariant representation of inputs.

$$E(A_g, \mathbf{b}_g) = \lambda \mathcal{R}(A_g) + \frac{1}{n}\sum_{i=1}^{n} \ell(\phi(g\mathbf{x}_i), A_g\phi(\mathbf{x}_i) + \mathbf{b}_g),$$

$$E(A_g, \mathbf{b}_g) = \lambda \mathcal{R}(A_g) + \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \phi_2 \circ (A_g, \mathbf{b}_g) \circ \phi_1(g^{-1}\mathbf{x}_i)).$$

Here, A_g and b_g each represents the linear transform and bias term of the matrix representation of the groups. Regularisation ensures sparsity. First loss function is used in the earlier layers ensuring equivariance; second is used in later layers ensuring target-oriented representation.

Learning the irreducible representations of commutative lie group (T.Cohen, 2014)

- By representing Toroidal Subgroup through block-diagonalization, each group element is represented by angles(φ) in transformed coordinate axis; transformation of axis is based on orthogonal matrix(W)

$$\begin{aligned} d^2(\mathbf{x}, \mathbf{y}) &= \min_{\varphi} \|\mathbf{y} - \mathbf{W}\mathbf{R}(\varphi)\mathbf{W}^T\mathbf{x}\|^2 \\ &= \sum_j \min_{\varphi_j} \|\mathbf{v}_j - \mathbf{R}(\varphi_j)\mathbf{u}_j\|^2 \\ &= \sum_j \|\mathbf{v}_j - \mathbf{R}(\hat{\mu}_j)\mathbf{u}_j\|^2, \end{aligned}$$

Here, v_j and u_j are each coordinates of x and y in the new coordinate system. Element g acting on x can be replaced by R(φ), a block diagonal matrix, multiplied to u_j

This idea can be applied to estimate the parameter of group or for subtask like classification

Representation Learning - 2

Transformation Properties of learned visual representation (T.Cohen, 2015)

- Learning the latent space representation of a class(Z_n) and applying group action(T) onto the latent vector to learn the class representation.
- Through the idea generation of object from an unseen viewpoint is possible.

$$p(x^{n,v} \mid z^n, g^{n,v}) = \mathcal{N}(x^{n,v} \mid f_\theta(\hat{T}(g^{n,v})\, z^n), \sigma_x^2)$$

- $x^{n,v}$ represents 'v'th view on the same object instance 'n'
- f_θ represents the neural network learning the mapping from the latent vector $T(g^{n,v})z^n$; $g^{n,v}$ represents the vth action applied to the 'n'th class latent vector z^n