

TENSOR FIELD NETWORK (AND OTHER CONVNET GENERALISATIONS)

TDLS - FEB 11. 2019

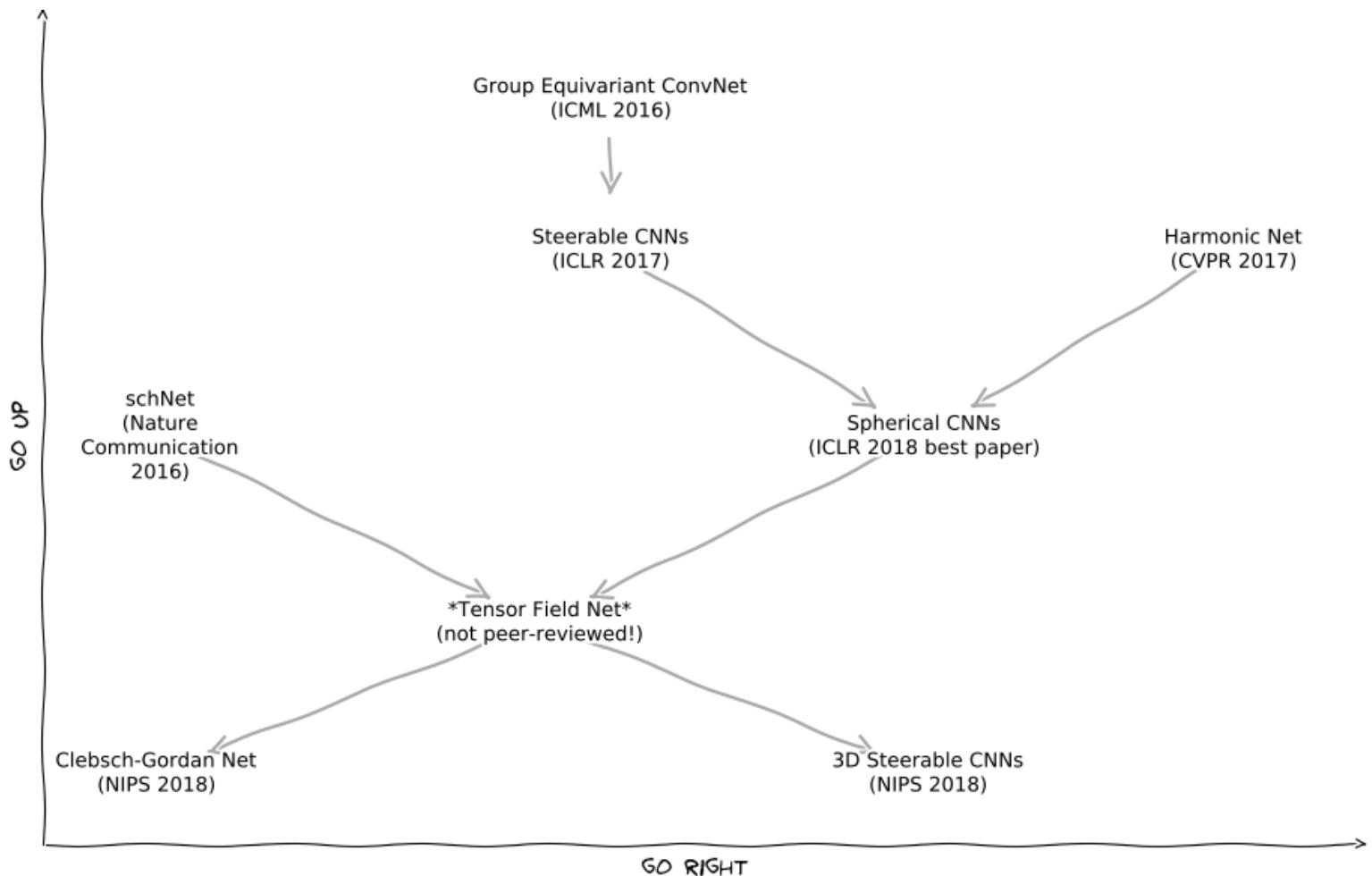
CHRIS DRYDEN

- christopher.paul.dryden@gmail.com
- github.com/chrisdryden

PENG CHENG

- pc175@uowmail.edu.au
- datapassports.com
- github.com/tribblloid

OVERVIEW



PRE-CONVNET (1960-1987)



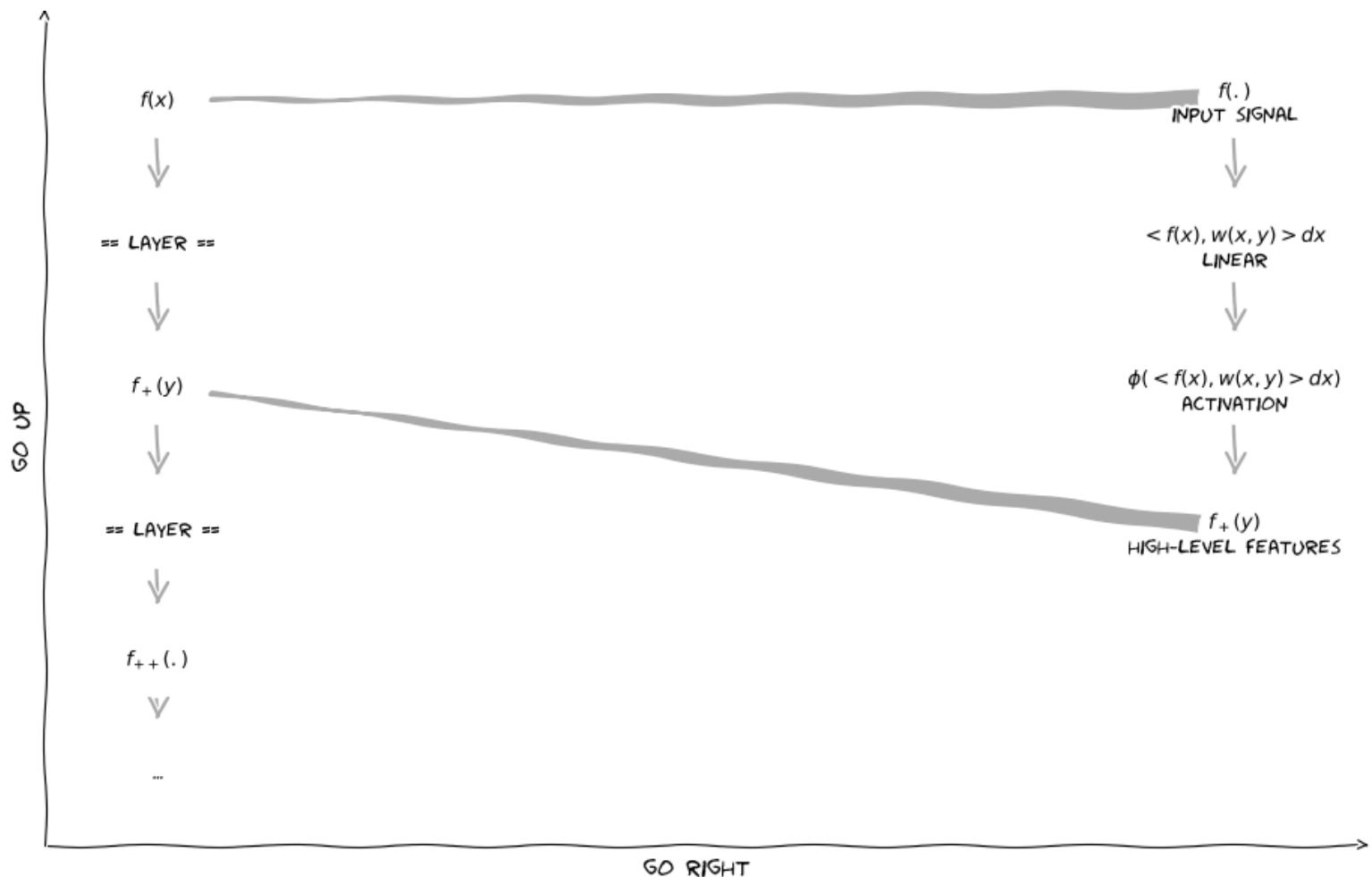
PRE-CONVNET - LINEAR/FULLY CONNECTED/DENSE/PERCEPTRON LAYER

In pursuing of unbounded representation/approximation power

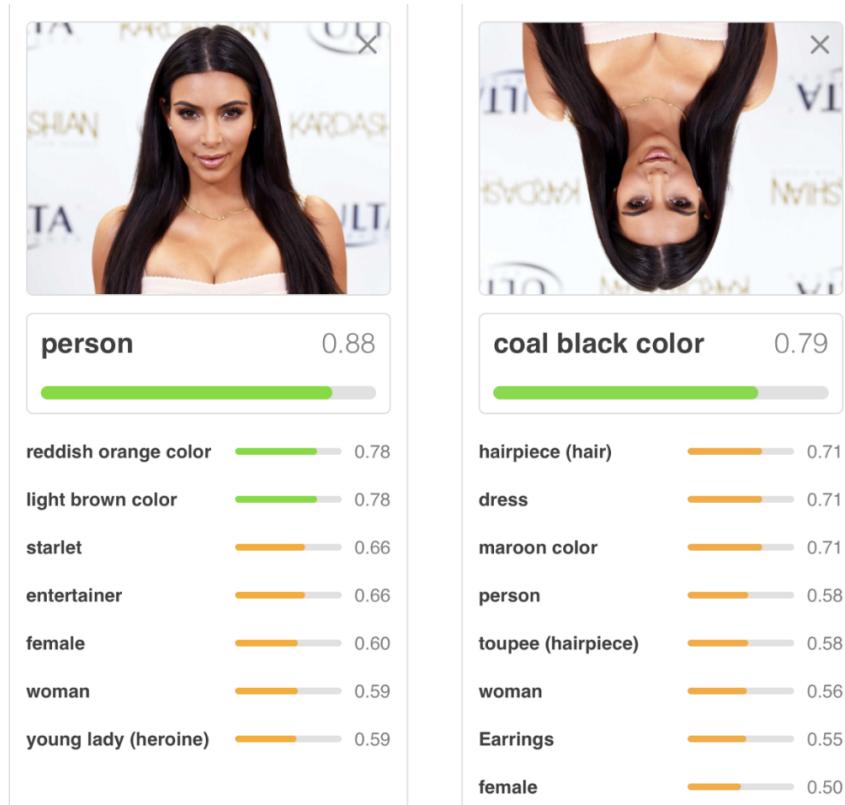
$$\begin{aligned} f_+(y) &= \Phi(f(x)) &= \phi\left(\sum_{x \in \text{domain}} f(x)w(x, y)\right) \\ &= \phi\left(\langle f(x), w(x, y) \rangle_x\right) \end{aligned}$$

(w are weight of neurons)

PRE-CONVNET - LINEAR/FULLY CONNECTED LAYER



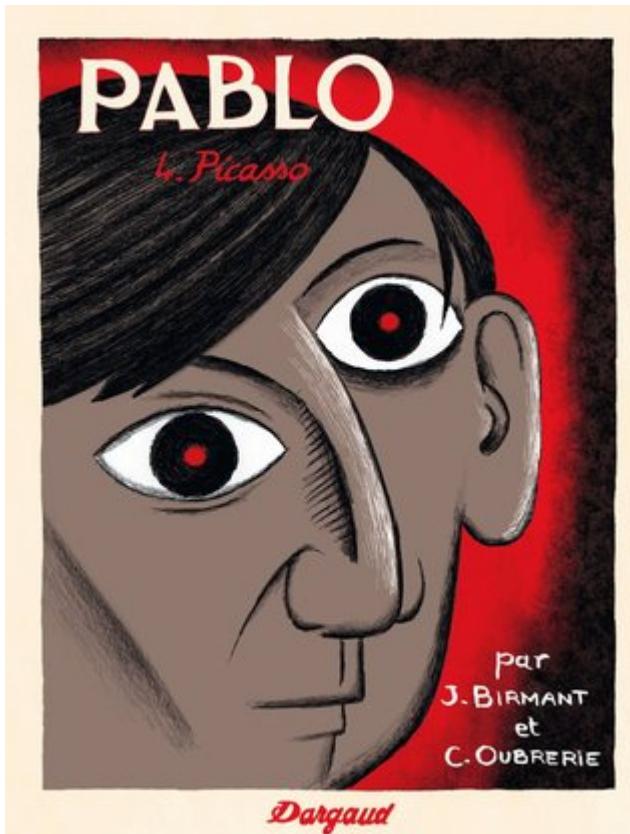
PRE-CONVNET - LINEAR/FULLY CONNECTED LAYER



[*] Image courtesy <https://www.quora.com/What-is-the-difference-between-equivariance-and-invariance-in-Convolution-neural-networks>
[\(https://www.quora.com/What-is-the-difference-between-equivariance-and-invariance-in-Convolution-neural-networks\)](https://www.quora.com/What-is-the-difference-between-equivariance-and-invariance-in-Convolution-neural-networks)

INVARIANT LAYER / BAG-OF-WORDS?

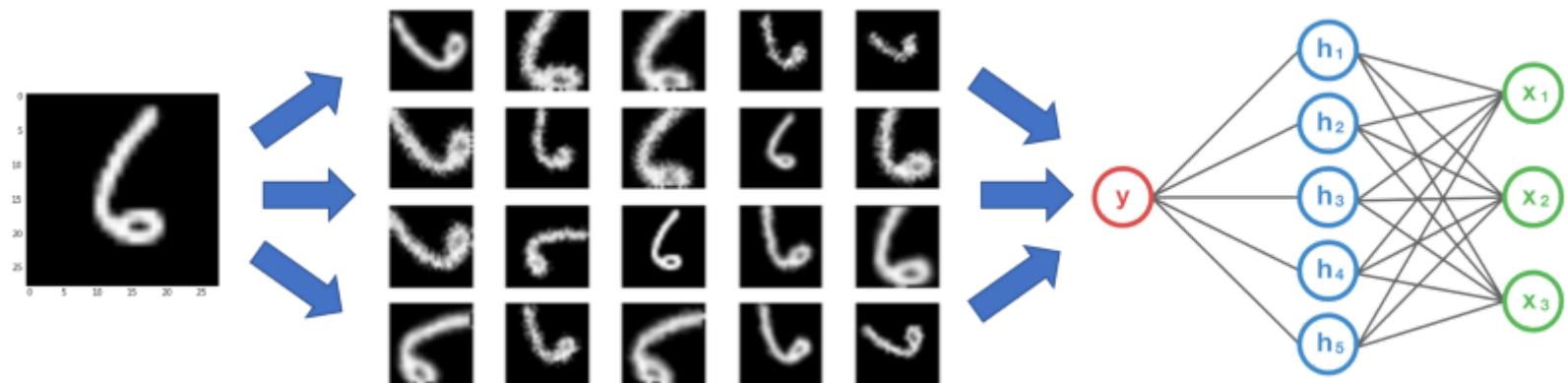
- Don't do this



[*] Image Courtesy: <https://www.amazon.ca/Pablo-Art-Masters-Julie-Birmant/dp/1906838941> (<https://www.amazon.ca/Pablo-Art-Masters-Julie-Birmant/dp/1906838941>)

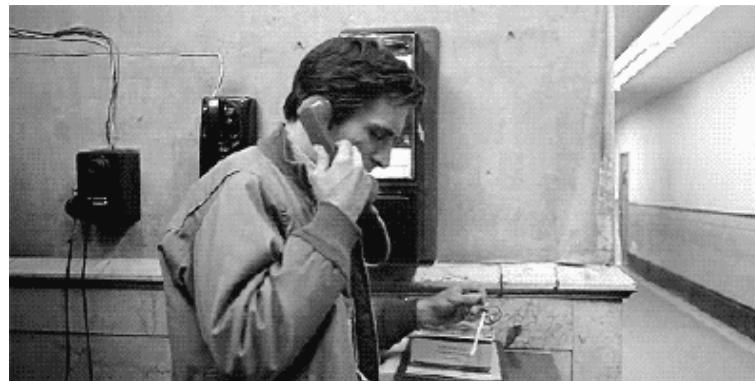
DATA AUGMENTATION

- Good catch



DATA AUGMENTATION

- Too slow in practice
 - In **convex case** SGD "theoretically probably" converges equally fast
 - otherwise it "kind of works" but with much less efficiency
 - Time & space complexity increase exponentially with the dimensionality of augmentation
-



2D translation

DATA AUGMENTATION

- Time & space complexity increase exponentially with the dimensionality of augmentation



2D translation x 1D rotation, no gravity

DATA AUGMENTATION

- Time & space complexity increase exponentially with the dimensionality of augmentation



2D translation \times 1D rotation, gravity perpendicular to domain

DATA AUGMENTATION

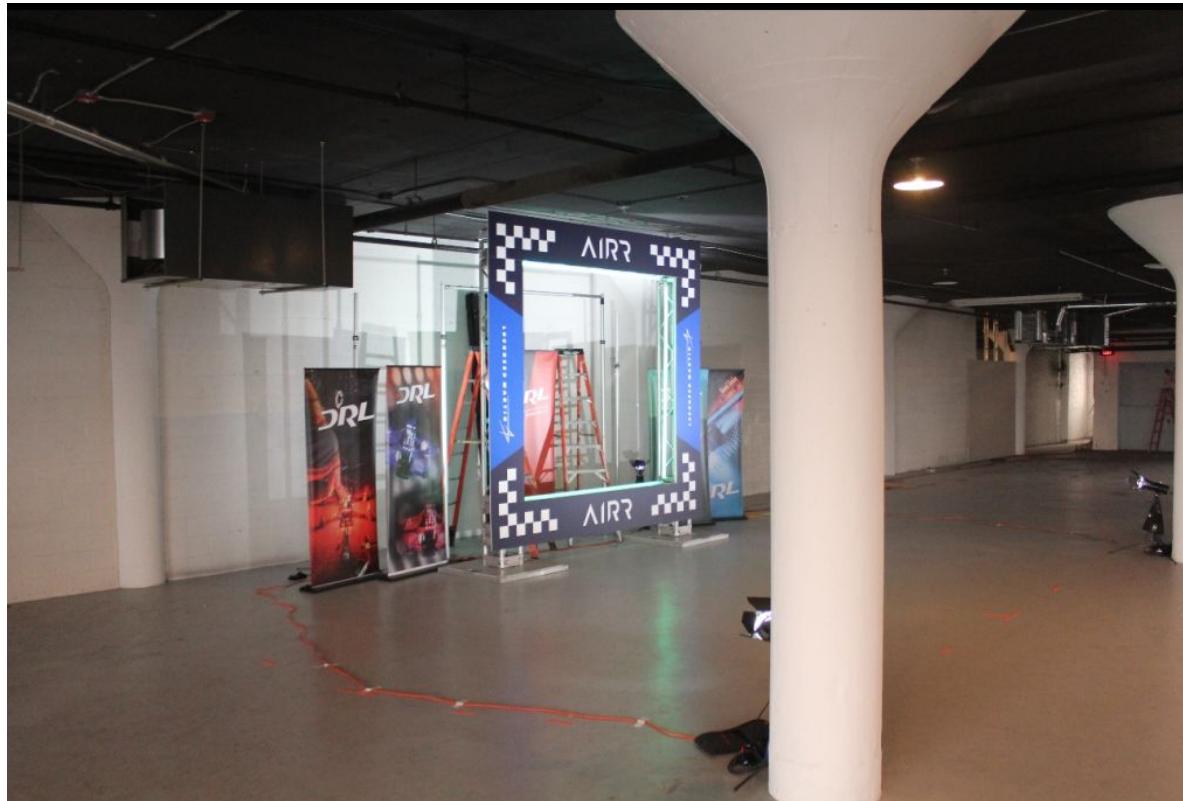
- Time & space complexity increase exponentially with the dimensionality of augmentation



3D rotation

DATA AUGMENTATION

- Time & space complexity increase exponentially with the dimensionality of augmentation

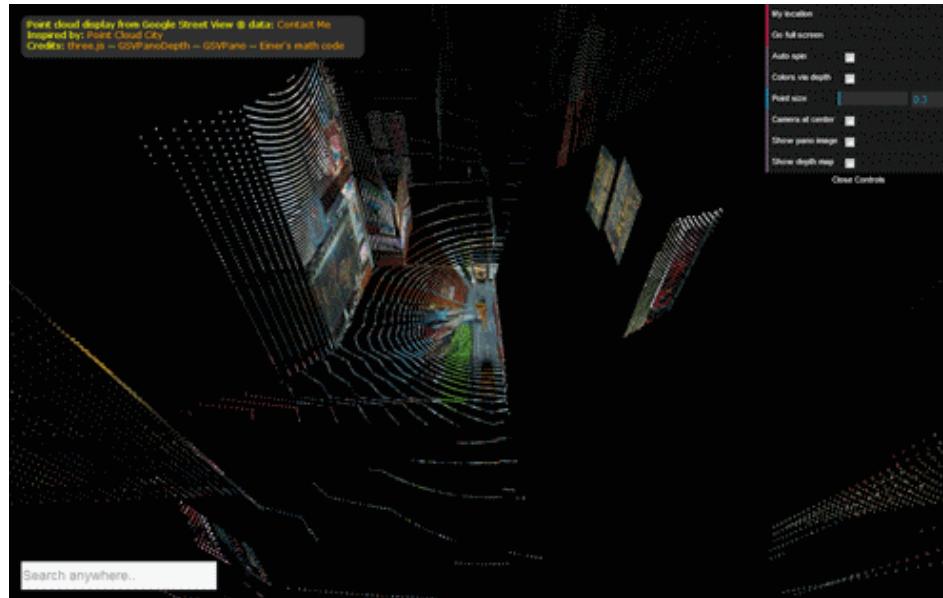


4D affine transformations

[*] Image Courtesy: AIRR <https://thedroneracingleague.com/airr/> (<https://thedroneracingleague.com/airr/>)

DATA AUGMENTATION

- Time & space complexity increase exponentially with the dimensionality of augmentation



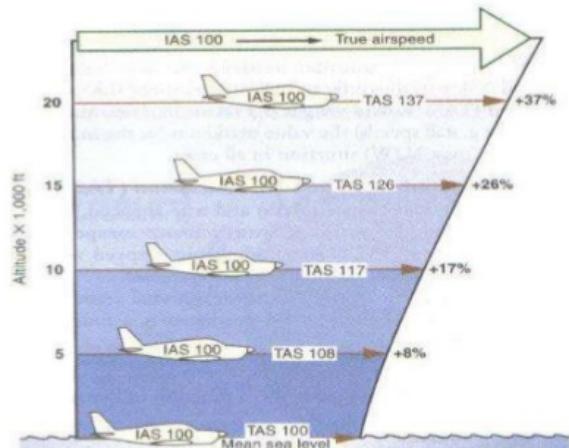
3D translation × 3D rotation

DATA AUGMENTATION

- Time & space complexity increase exponentially with the dimensionality of augmentation

IAS and TAS

Rules of Thumb: Increase your IAS at from MSL by 2% (or 1.8%) per 1000ft increase to obtain the Gross TAS:

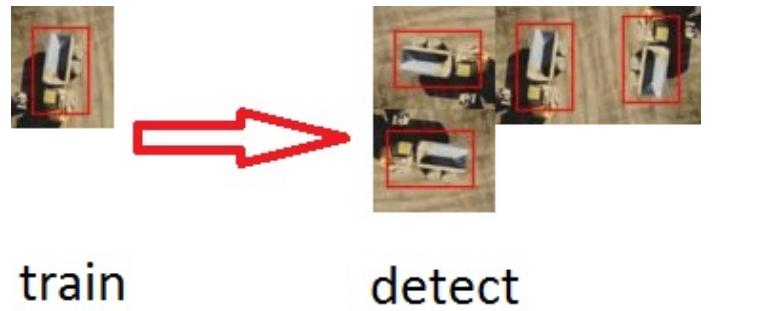


■ Figure 25-52 With IAS constant, TAS increases with increase in altitude

DATA AUGMENTATION

How about a better idea?

- Instead of augmenting, we hard-bake such prior knowledge into the network to yield identical result!



Augmentation types	Answer
2d translation	ConvNet
others	G-ConvNet
- 2d translation + 90° rotation	Group Equivariant CNNs
- 2d translation + rotation	Harmonic Net
- 3d rotation	Spherical CNNs
- 3d translation + rotation	Tensor Field Net

Let's do a hello-world experiment on MNIST dataset:



... for which each image can be augmented until all cases are covered



START LEARNING!

2 LAYERS ONLY

1. Highway only bypassing Linear/FC/~~Dense/Perceptron~~

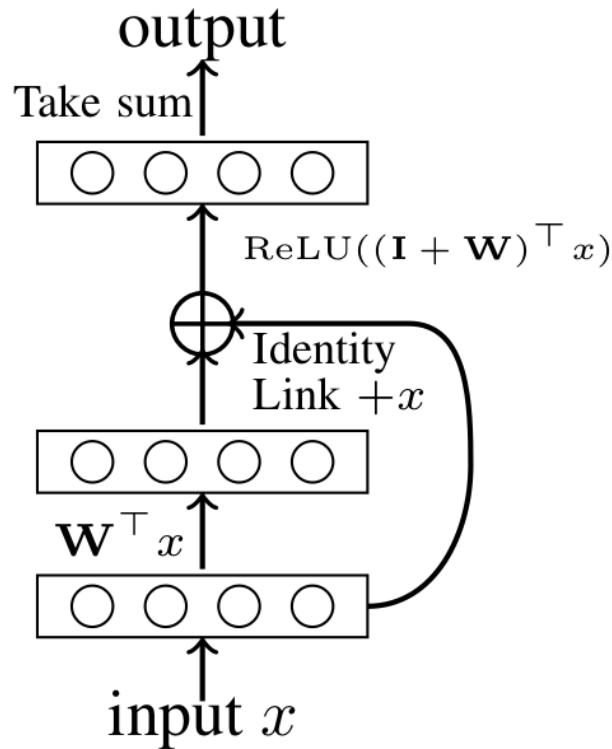
- Designed to break symmetry at saddle points*
- WITHOUT initialisation (all weights start with 0)
- $10 \times 10 \Rightarrow 10 \times 10 \Rightarrow \text{ReLU}$

2. Plain old Linear/FC

- $10 \times 10 \Rightarrow \text{one-hot } 1 \sim 10 \Rightarrow \text{ReLU}$

[*] Y. Li and Y. Yuan, "Convergence Analysis of Two-layer Neural Networks with ReLU Activation" NIPS 2017, pp. 1-11.

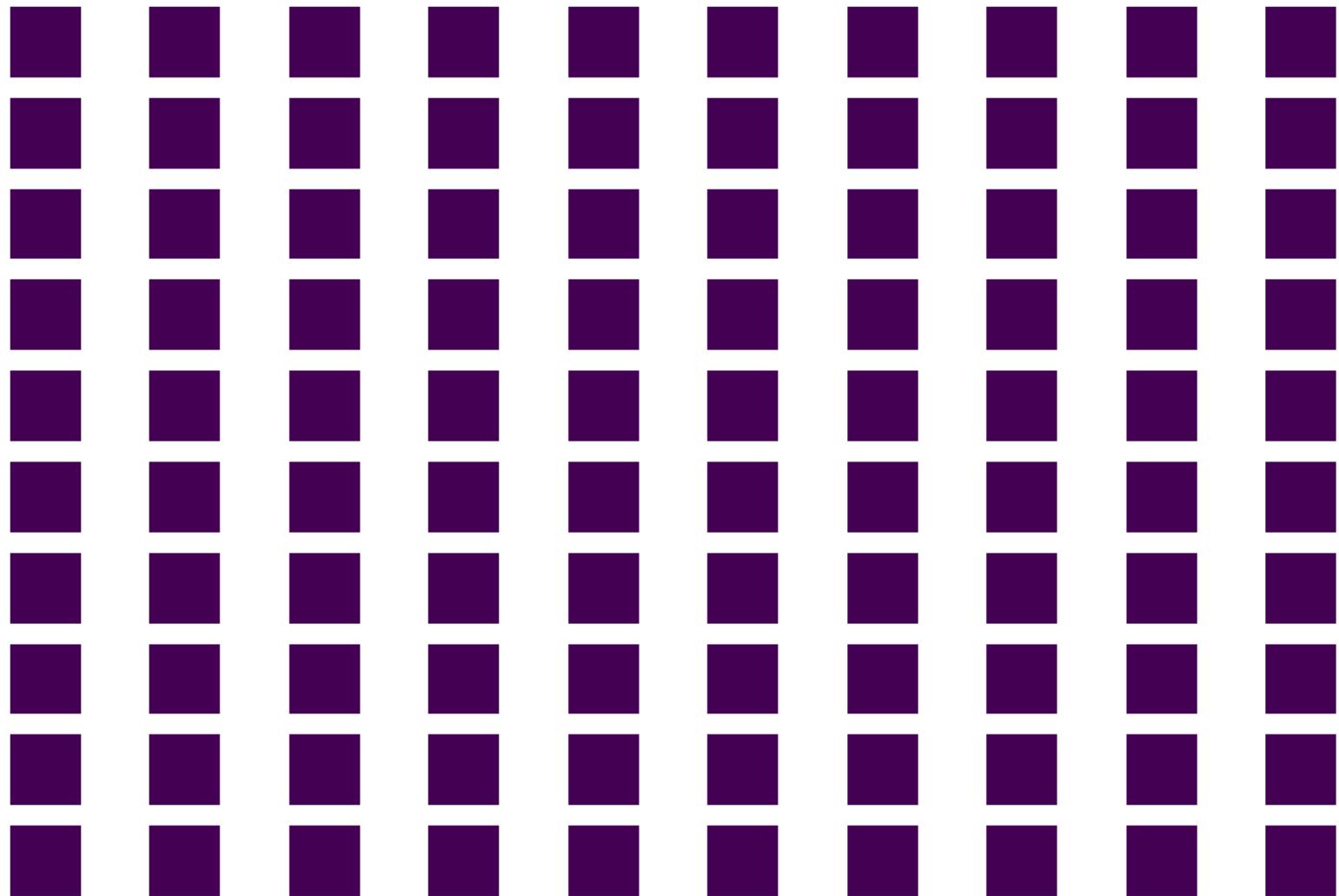
START LEARNING!



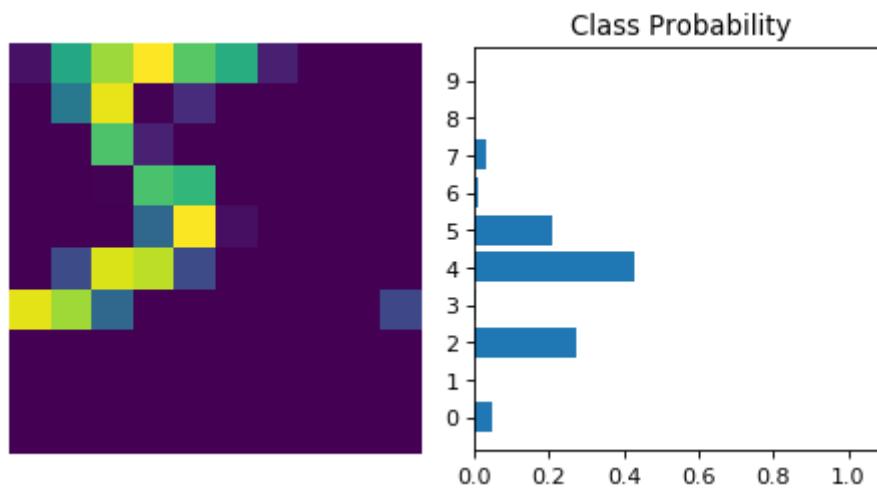
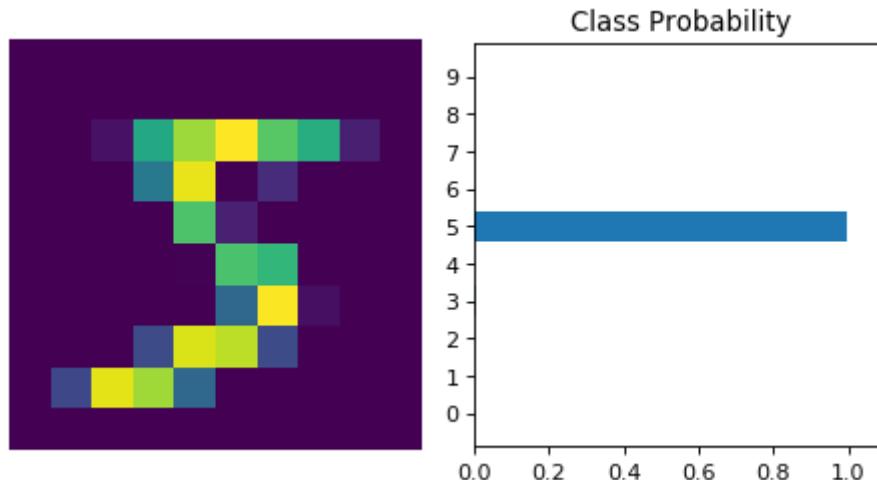
), with identity mapping (right)

[*] Y. Li and Y. Yuan, "Convergence Analysis of Two-layer Neural Networks with ReLU Activation" NIPS 2017, pp. 1-11.

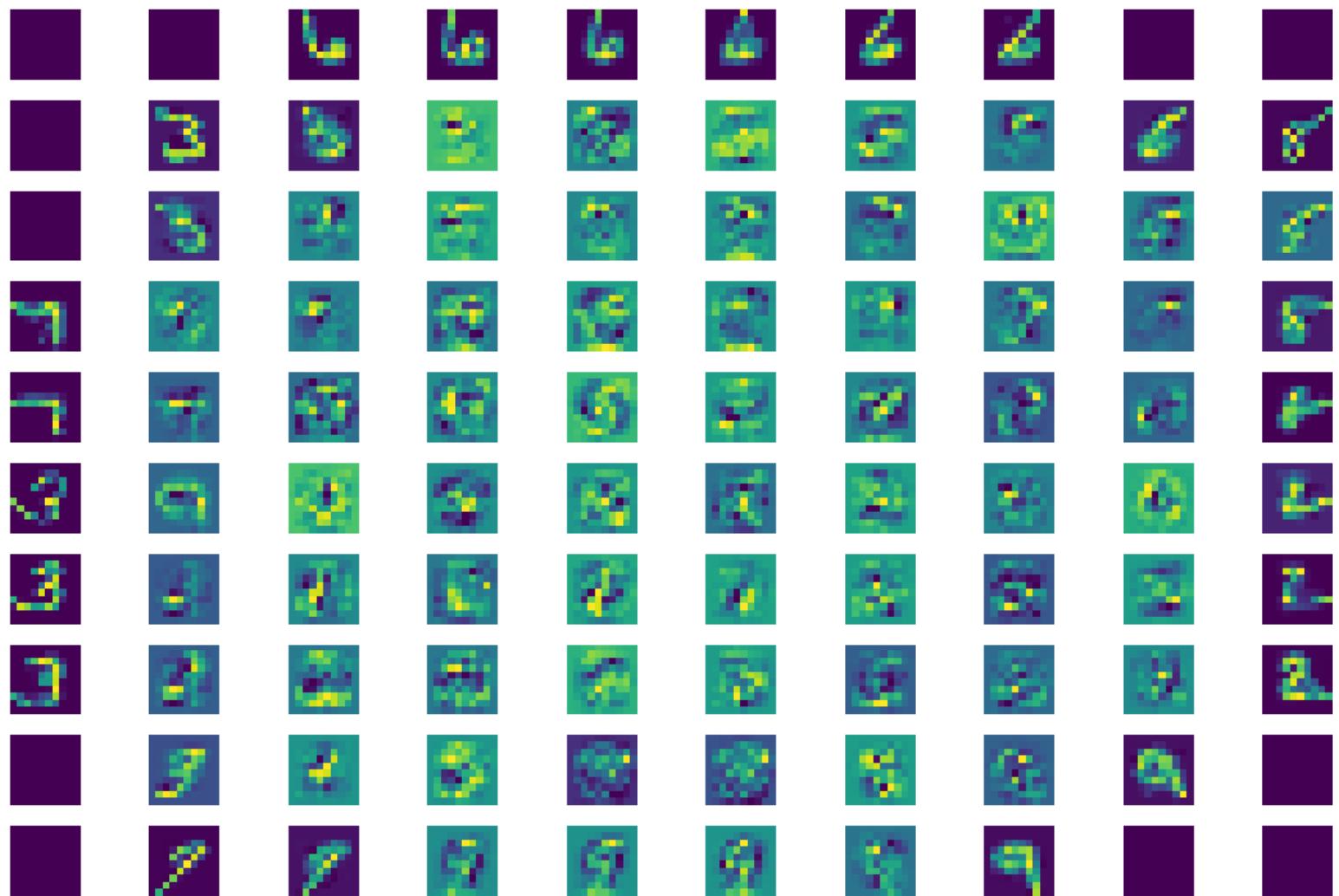
Weight map before training



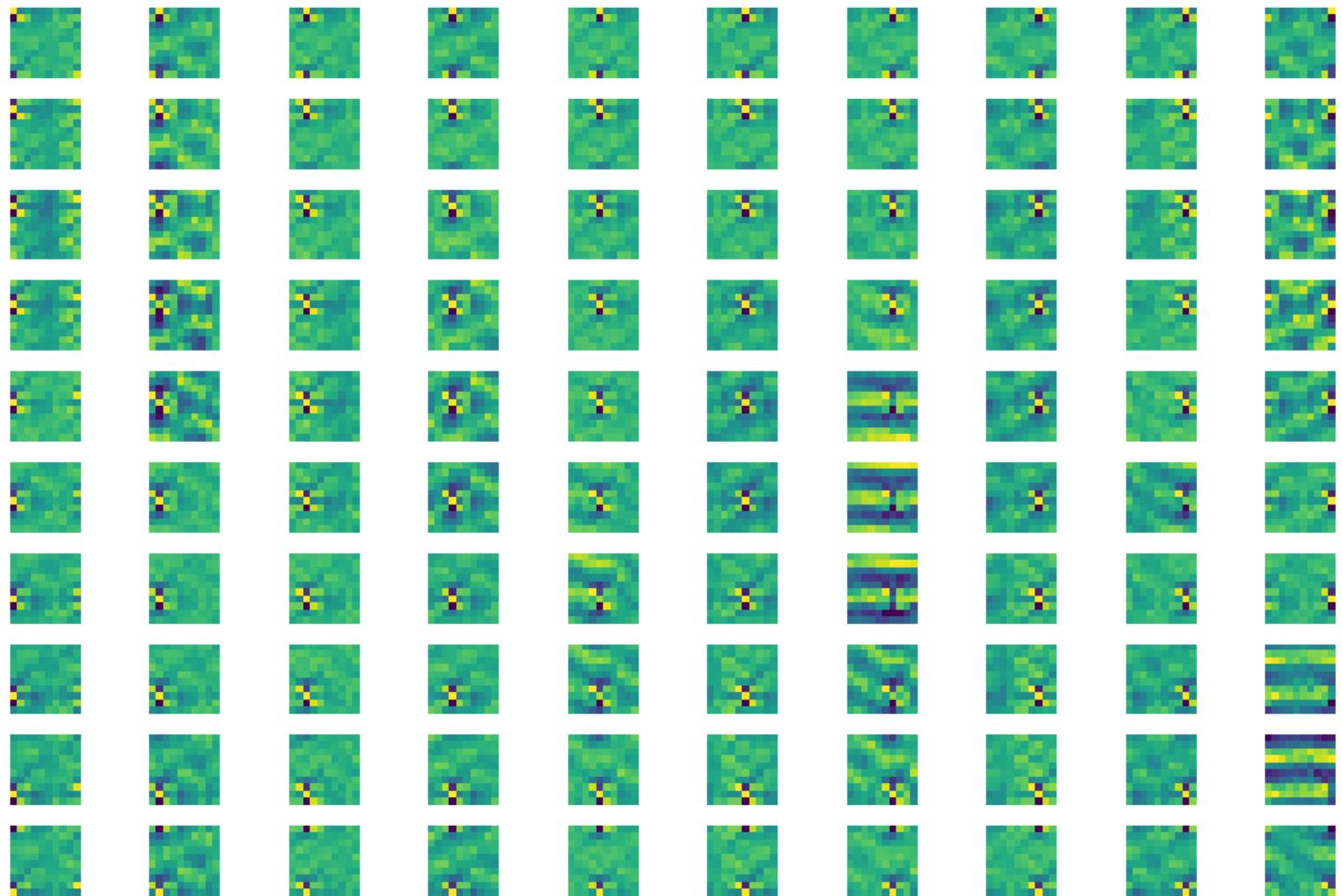
Trained on raw MNIST dataset



Weight map after training on raw MNIST dataset



Weight map after training on AUGMENTED MNIST dataset



DATA AUGMENTATION

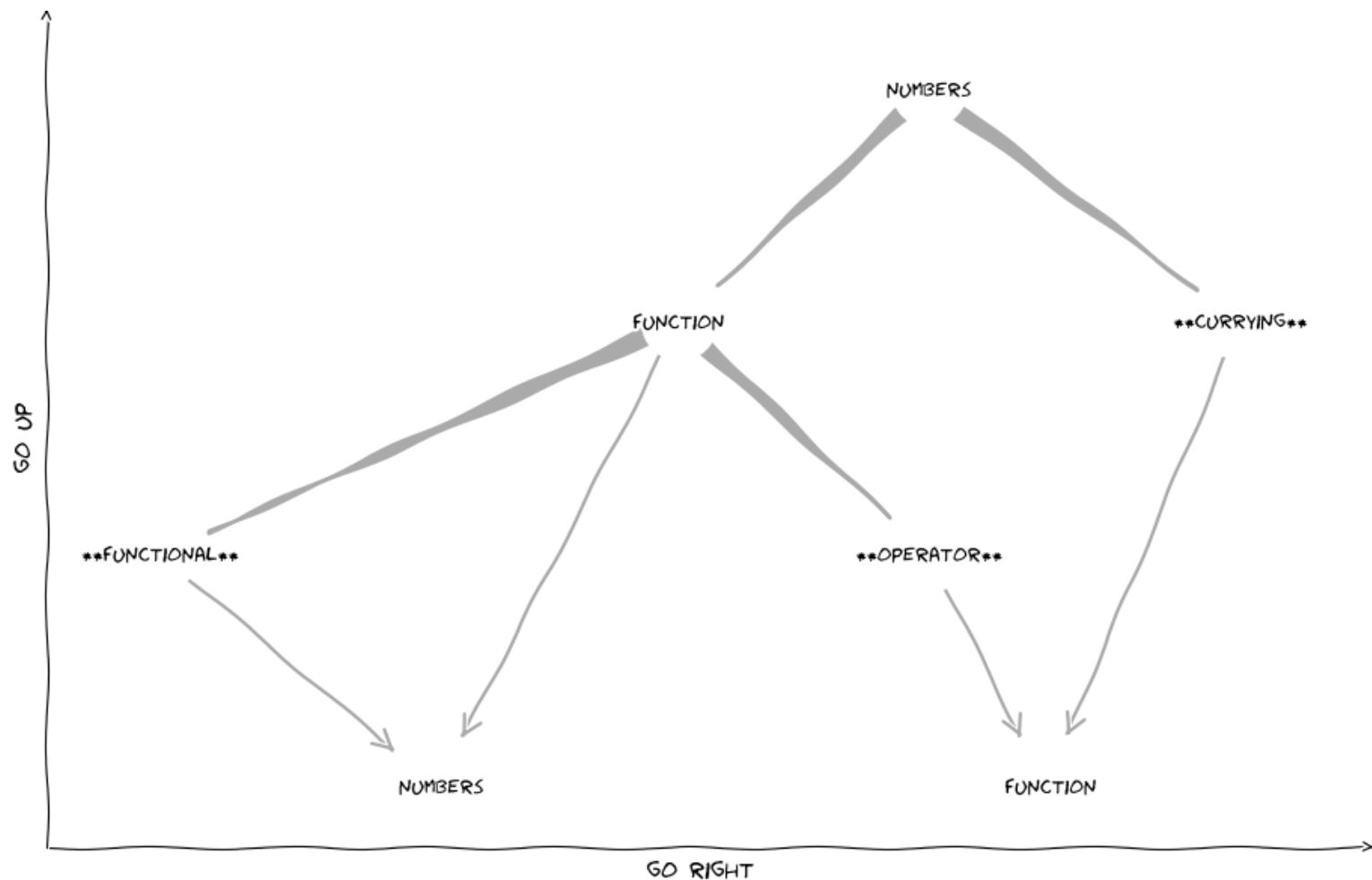
- How did this happen? (you probably want to try this on neural-ODE)



DATA AUGMENTATION

- Form a group $\left\{ A_{ug}, U_{ga}, G_{au} \dots \dots \right\}$, sometimes not commutative/Abelian
- \subset unary operator : $Signal \implies Signal$ (making it a group action)
- \subset **higher-order function**

DATA AUGMENTATION



DATA AUGMENTATION \implies G-CONVNET

Lemma: If the augmentation group $\{A_{ug}\}$ satisfies:

- **Transitivity:** for any pair of points x, y and any function f , we can always find an augmentation that can transform value $f(x)$ to point y
- **Group Equivariance:** applying an augmentation A_{ug} on the input has the same effect as applying an augmentation U_{ga} from the same group on the output

Then a fully connected layer:

$$f_+(y) = \langle f(x), w(x, y) \rangle_x$$

collapses to a group convolution (**G-conv**) layer:

$$f_+(y) = \langle A_{ug} \circ f(x), w_0(x) \rangle_x$$

Looks familiar?

$$\text{conv}(f(-\Delta), w_0(\Delta)) = \text{corr}(f(\Delta), w_0(\Delta)) = \langle f(\Delta + x), w_0(x) \rangle_x$$

DATA AUGMENTATION \implies G-CONVNET

$$f_+(y) = \langle A_{ug} \circ f(x), w_0(x) \rangle_x$$

In short:

A CONVNET LAYER IS JUST AN AUGMENTED LINEAR/FULLY-CONNECTED LAYER!

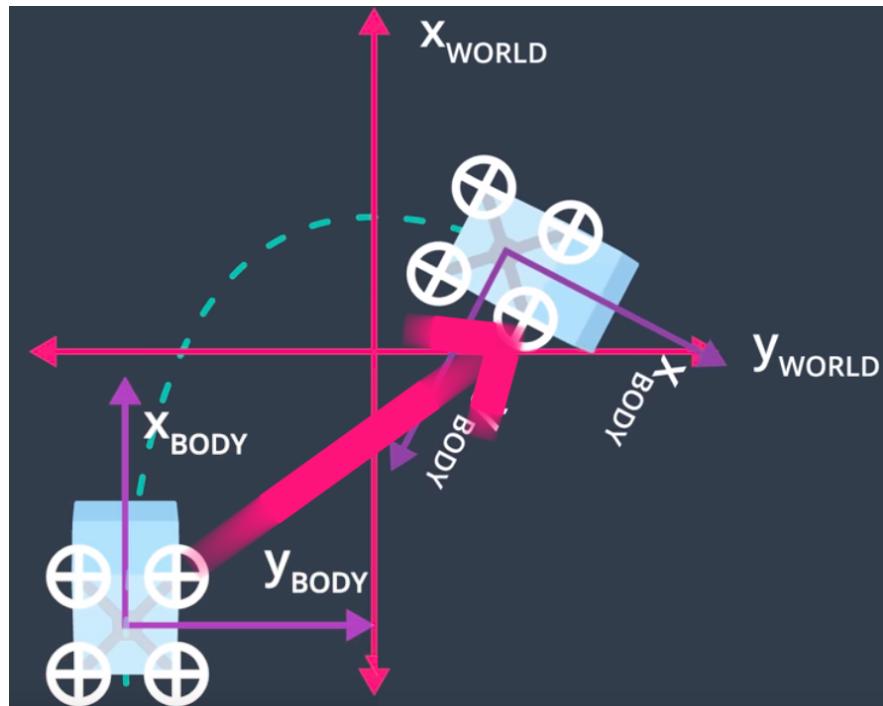
DATA AUGMENTATION - TRANSITIVITY

Transitivity: for any pair of points x, y and any function f , we can always find an augmentation that can transform value $f(x)$ to point y

$$\forall x : f(x) = (A_{ug} \circ f)(x_0)$$

DATA AUGMENTATION - TRANSITIVITY

- Effectively means the augmentation group is the 'carriage' to move reference frame around the observer



[*] Image courtesy: udacity.com

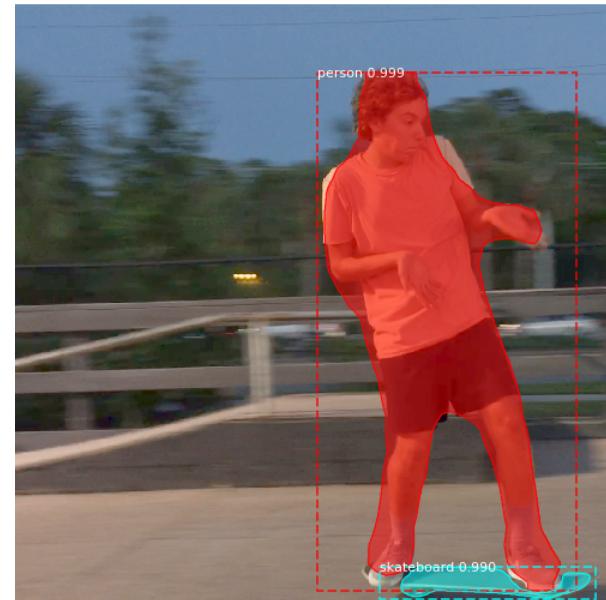
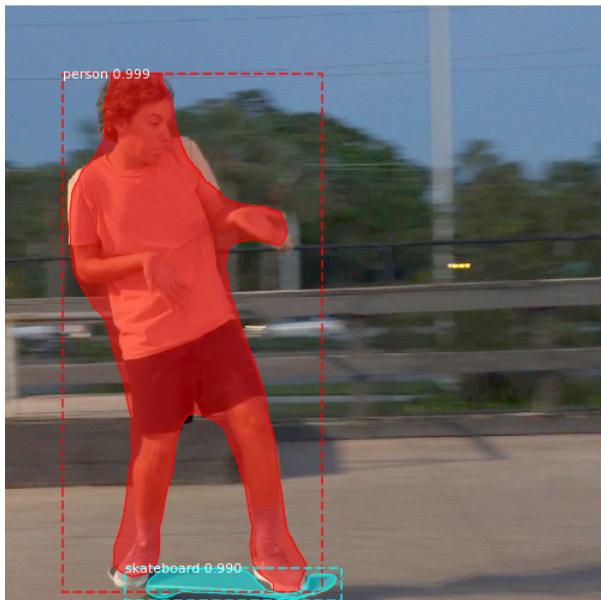
DATA AUGMENTATION - EQUIVARIANCE

Plain old **Equivariance**: applying an augmentation A_{ug} on the input has the same effect as applying A_{ug} on the output

$$A_{ug} \circ f_+(y) = < A_{ug} \circ f(x), w(x, y) >_x$$

DATA AUGMENTATION - EQUIVARIANCE

- example: SQL predicate pushdown
- example: first input & final output of Masked-CNN & Autoencoder (& maybe Style Transfer)



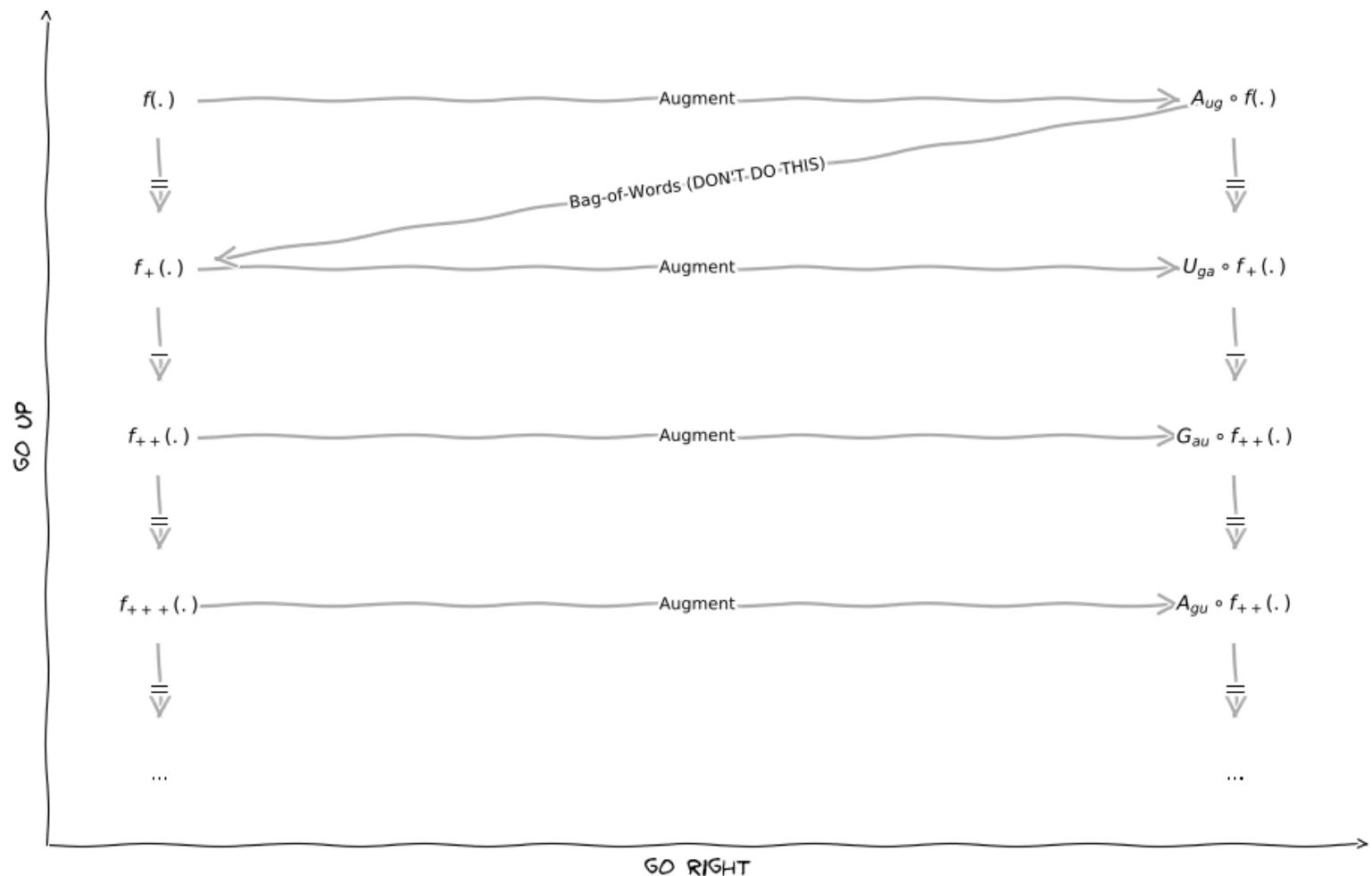
DATA AUGMENTATION - GROUP EQUIVARIANCE

Group Equivariance: applying an augmentation A_{ug} on the input has the same effect as applying an augmentation U_{ga} **from the same group** on the output

$$U_{ga} \circ f_+(y) = < A_{ug} \circ f(x), w(x, y) >_x$$

- Relaxed a bit comparing to Plain old equivariance
- Effectively means that the architecture of the first layer can be carried over to the subsequent layers with little changes, applied on high-level features

DATA AUGMENTATION - GROUP EQUIVARIANCE



DATA AUGMENTATION \implies **G-CONVNET - PROOF**

- **Transitivity:**

$$\forall x : f(x) = (\bar{A}_{ug} \circ f)(x_0)$$

- **Group equivariance**

$$U_{ga} \circ f_+(y) = \langle A_{ug} \circ f(x), w(x, y) \rangle_x$$

Combining all together:

$$f_+(y) = (\bar{U}_{ga} \circ f_+)(y_0) = \langle \bar{A}_{ug} \circ f(x), w(x, y_0) \rangle_x = \langle \bar{A}_{ug} \circ f(x), w_0(x) \rangle_x$$

[*] More rigorous proof: R. Kondor and S. Trivedi, "On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups" 2018

G-CONVNET

$$f_+(y) = \langle A_{ug} \circ f(x), w_0(x) \rangle_x$$

- this implies bijection/isomorphism $y \longleftrightarrow A_{ug}$
- ... and high-level features usually have more dimensions $\{x\} \subset \{y\}$

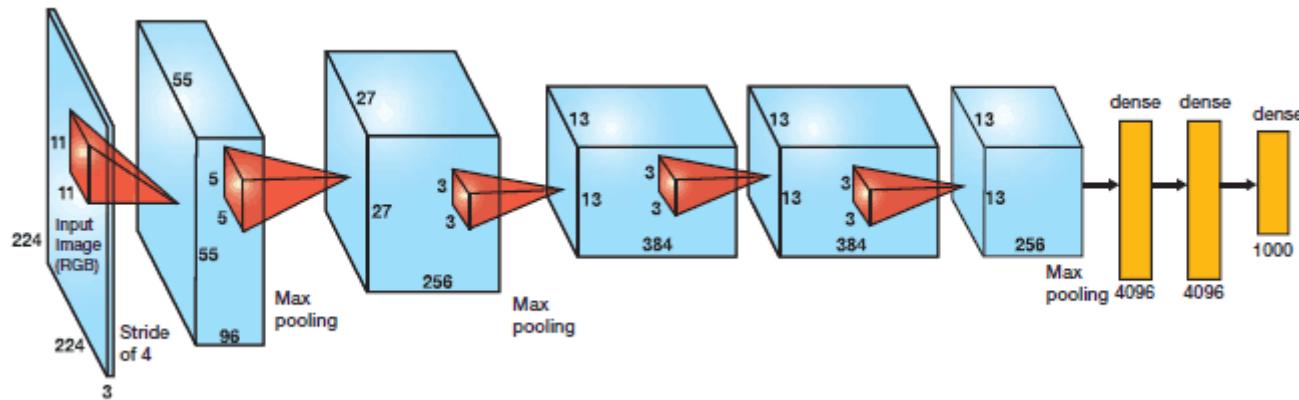
All of the followings are concrete subclasses:

Augmentation types	Answer
2d translation	ConvNet
others	G-ConvNet
- 2d translation + 90° rotation	Group Equivariant CNNs
- 2d translation + rotation	Harmonic Net
- 3d rotation	Spherical CNNs
- 3d translation + rotation	Tensor Field Net

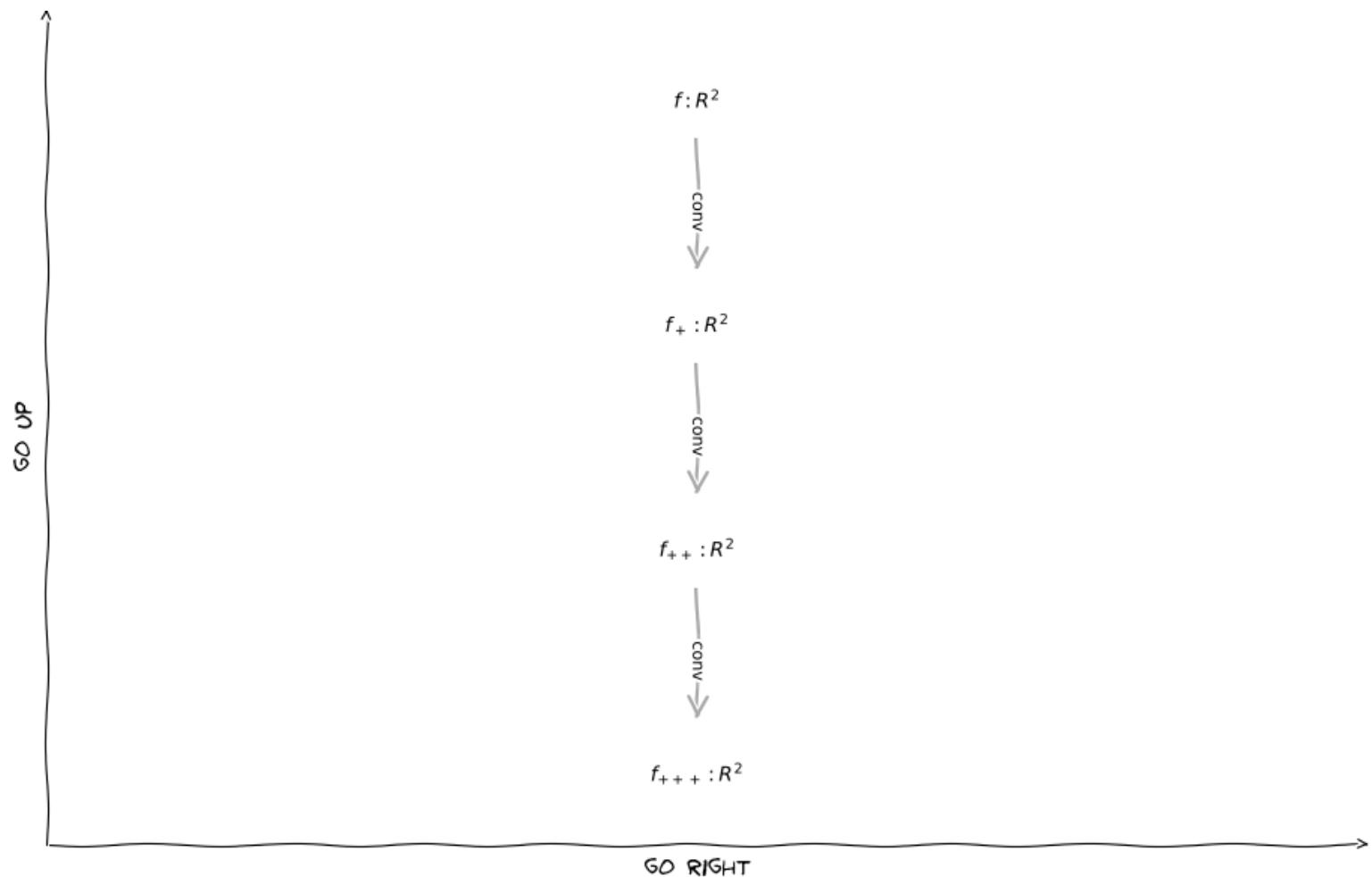
CONVNET

-	Input $f(x)$	High-level $f_+(y), f_{++}(z), \dots$	Augmentation A_{ug}, U_{ga}, \dots
domain	\mathbb{R}^2	\mathbb{R}^2	\mathbb{R}^2 (translation only)

- First of its kind but not the last
- A rare case when high-level feature domain $\{y\} = \{x\}$, in all other cases $\{y\} \supset \{x\}$



CONVNET



GROUP EQUIVARIANT CNNS (ICML 2016*)

-	Input $f(x)$	High-level $f_+(y), f_{++}(z), \dots$	Augmentation A_{ug}, U_{ga}, \dots
domain	R^2	$R^2 \times p4$	$R^2 \times p4$ (translation, rotation $\pm 90^\circ$)

- change looks trivial

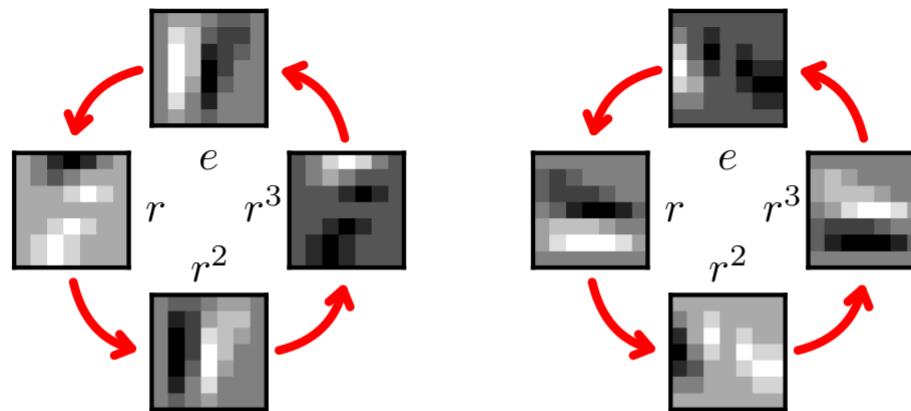
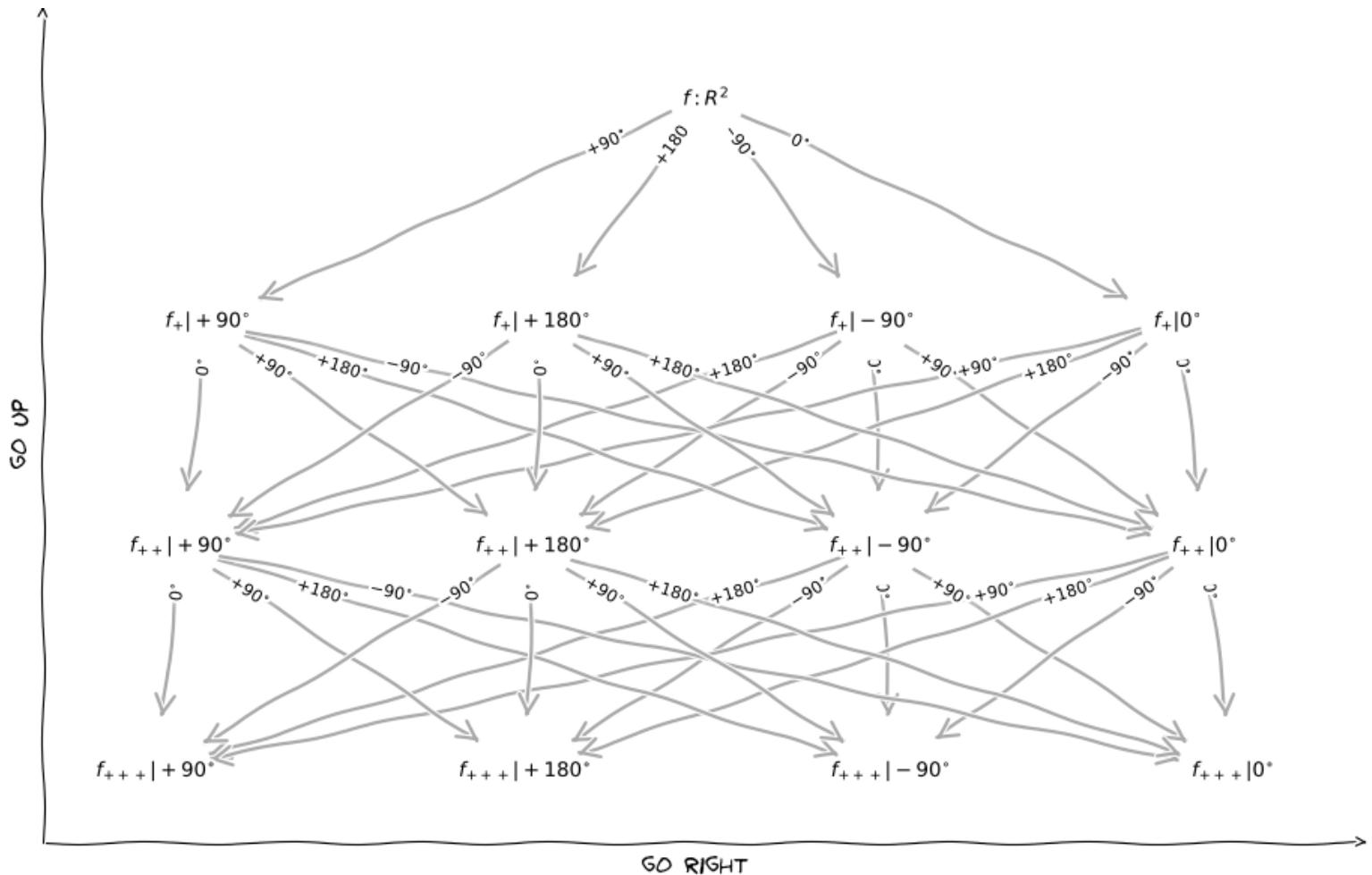


Figure 1. A $p4$ feature map and its rotation by r .

[*] T. S. Cohen and M. Welling, "Group Equivariant Convolutional Networks," ICML 2016.

GROUP EQUIVARIANT CNNS (ICML 2016*)



GROUP EQUIVARIANT CNNS (ICML 2016*) - ALTERNATIVELY

-	Input $f(x)$	High-level $f_+(y), f_{++}(z), \dots$	Augmentation A_{ug}, U_{ga}, \dots
domain	R^2	$R^2 \times p4m$	$R^2 \times p4m$ (translation, rotation $\pm 90^\circ$, flipping)

- Size of filter bank start to become annoying, but still acceptable.

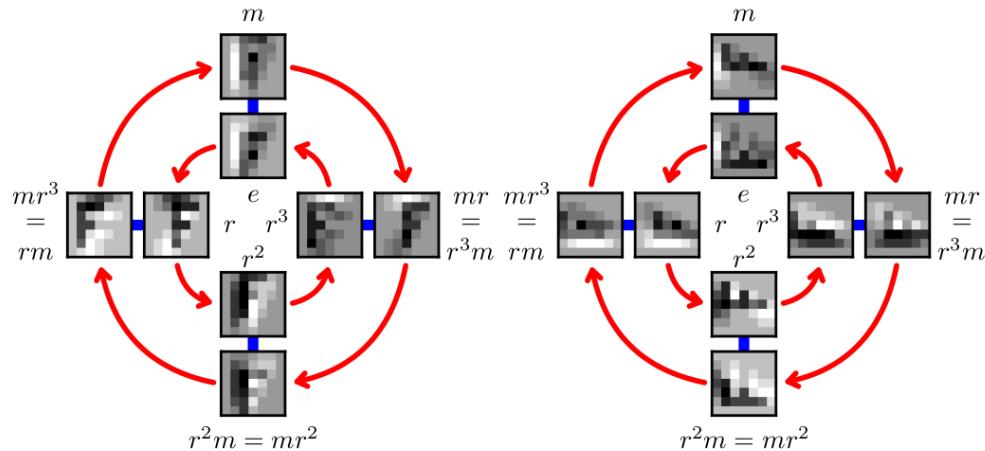


Figure 2. A p4m feature map and its rotation by r .

[*] T. S. Cohen and M. Welling, "Group Equivariant Convolutional Networks," ICML 2016.

HARMONIC NET (CVPR 2017*)

-	Input $f(x)$	High-level $f_+(y), f_{++}(z), \dots$	Augmentation A_{ug}, U_{ga}, \dots
domain	R^2	$O(2)$	$O(2) \cong R^2 \times SO(2)$ (translation, arbitrary rotation)

- Size of the filter bank become intolerably big, can't take it any more
- First algorithm to use **spectral decomposition as a weight compressor**



[*] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow, "Harmonic Networks: Deep Translation and Rotation Equivariance" CVPR 2017, vol. 2017-Jan, pp. 7168–7177.

GOING SPECTRAL

- function is like infinite-dimension vector
 - Spectral decomposition for functions is like eigen-decomposition for vectors
-

orthonormal basis: given a function domain $X \rightarrow C$, there may exist a (likely infinite) series of bases u_1, u_2, \dots that are both:

- **complete**: linear combination can approximate arbitrary function on the domain

$$f(\cdot) = \sum_{\forall m} \phi_m u_m(\cdot) = [\phi_1, \phi_2, \dots] \begin{bmatrix} u_1(\cdot) \\ u_2(\cdot) \\ \dots \end{bmatrix}$$

GOING SPECTRAL

- function is like infinite-dimension vector
 - Spectral decomposition for functions is like eigen-decomposition for vectors
-
- and **orthonormal**: have unit norms and orthogonal to each other
 $\langle u_m(\cdot), u_n(\cdot) \rangle = I_{mn}$ (**Kronecker delta**)
 - ... which implies:
$$\phi_m = \widehat{f}(m) = \langle f(\cdot), u_m(\cdot) \rangle \quad (\text{GFT})$$

GOING SPECTRAL - WHAT'S THE POINT?

It makes a few things easier:

- Convolution theorem still works in most cases! a.k.a. **G-conv theorem**

$$\widehat{f}_+(m) = \langle \widehat{A}_{ug}(m) \circ \widehat{f}(m), \widehat{w}_0(m) \rangle_m$$

(This makes dot product and G-conv much faster)

- Most G-ConvNet features are smooth on all dimensions

(This means low-frequency coefficients can compress high-dimension filter banks)

G-CONV THEOREM - PROOF

$$\begin{aligned} f_+(y) &= \langle A_{ug} \circ f(x), w_0(x) \rangle_x \\ &= \sum_m \sum_n \widehat{A_{ug} \circ f(m)} \widehat{w_0}(j) \langle u_m(\cdot), u_n(\cdot) \rangle \\ (\text{orthonormal}) \quad &= \sum_m \widehat{A_{ug} \circ f(m)} \widehat{w_0}(m) \\ (\text{GFT}) \quad &= \sum_m \langle A_{ug} \circ f(x), u_m(x) \rangle_x \cdot \langle w_0(x), u_m(x) \rangle_x \end{aligned}$$

[*] More rigorous proof for $SO(3)$ case: T. S. Cohen, M. Geiger, J. Koehler, and M. Welling, “Spherical CNNs,” no. 3, pp. 1–15, 2018.

G-CONV THEOREM - PROOF

If luckily A_{ug} is linear:

(bijectory)

$$\widehat{f}_+(n) = \sum_m \left\langle \left\langle A_{ug}(y) \circ f(x), u_m(x) \right\rangle_x \Big| u_n(y) \right\rangle_y \cdot \langle w_0(x), u_m(x) \rangle_x$$

(linear)

$$= \sum_m \langle u_n(y), A_{ug}(y) \rangle_y \circ \langle f(x), u_m(x) \rangle_x \cdot \langle w_0(x), u_m(x) \rangle_x$$

$$= \langle \widehat{A}_{ug}(n) \circ \widehat{f}(m), \widehat{w_0}(m) \rangle_m$$

(IFF A_{ug} is distance-preserving)

$$= \langle \widehat{f}(m), \widehat{A}_{ug}^{-1}(n) \circ \widehat{w_0}(m) \rangle_m$$

HARMONIC NET (CVPR 2017*)

-	Input $f(x)$	High-level $f_+(y), f_{++}(z), \dots$	Augmentation A_{ug}, U_{ga}, \dots
domain	R^2	$O(2)$	$O(2) \cong R^2 \times SO(2)$ (translation, arbitrary rotation)

- $u_m(x) \leftarrow e^{mx}$ (2D Fourier series, x is a complex number)
- GFT \leftarrow FFT (with Gaussian resampling)
- number of coefficients $\leftarrow 2$: $m \in 0, 1$

[*] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow, "Harmonic Networks: Deep Translation and Rotation Equivariance" CVPR 2017, vol. 2017-Jan, pp. 7168–7177.

HARMONIC NET (CVPR 2017*)

Orthonormal bases (without radial):

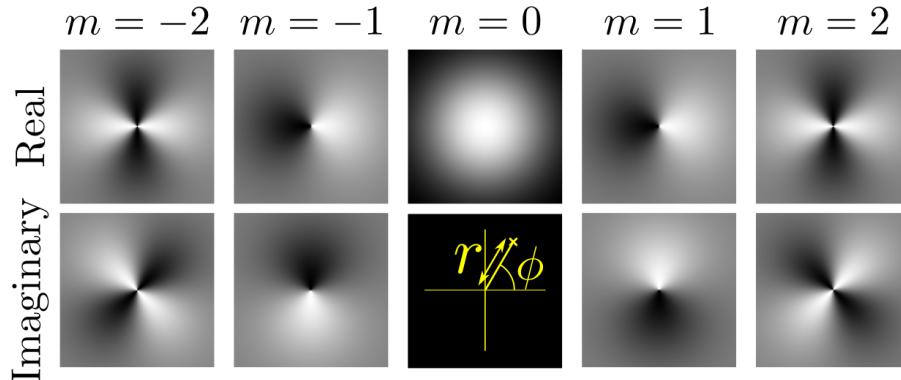
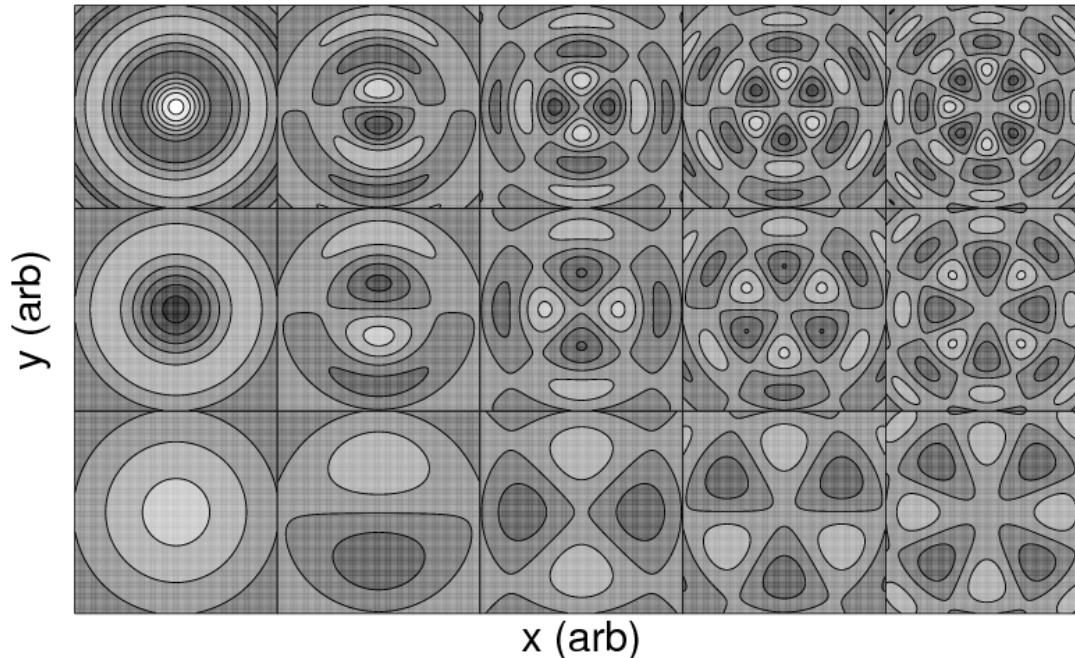


Figure 2. Real and imaginary parts of the complex Gaussian filter $\mathbf{W}_m(r,\phi';e^{-r^2},0) = e^{-r^2} e^{im\phi}$, for some rotation orders. As a simple example, we have set $R(r) = e^{-r^2}$ and $\beta = 0$, but in general we learn these quantities. Cross-correlation, of a feature map of rotation order n with one of these filters of rotation order m , results in a feature map of rotation order $m+n$. Note the negative rotation order filters have flipped imaginary parts compared to the positive orders.

[*] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow, "Harmonic Networks: Deep Translation and Rotation Equivariance" CVPR 2017, vol. 2017-Jan, pp. 7168–7177.

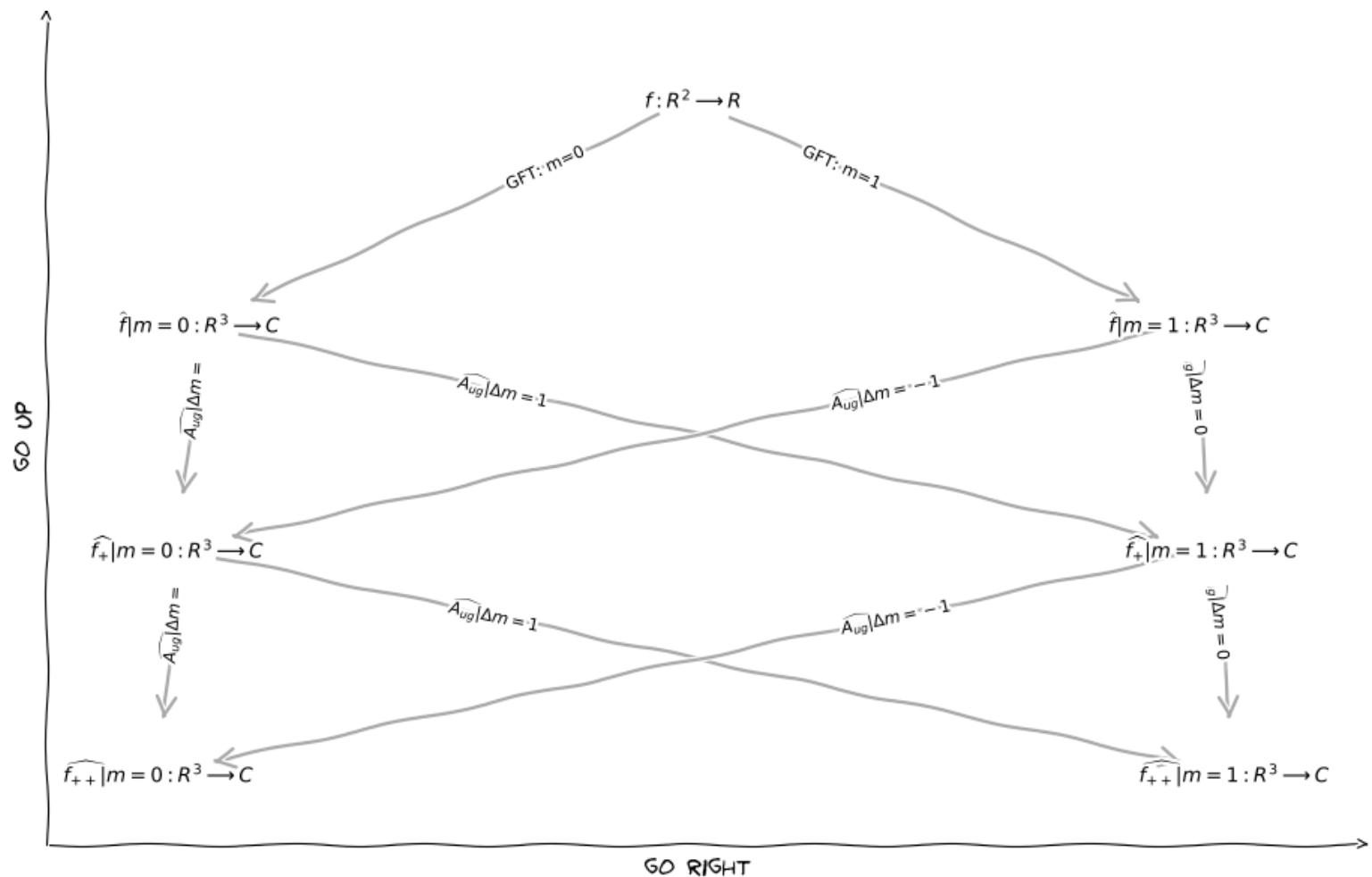
HARMONIC NET (CVPR 2017*)

Orthonormal bases (with radial):



[*] Image courtesy: C. E. Coleman-Smith, H. Petersen, and R. L. Wolpert,
"Classification of initial state granularity via 2d Fourier Expansion" Apr. 2012.

HARMONIC NET (CVPR 2017*)



SPHERICAL CNNS (ICLR 2018* BEST PAPER)

You can use many cameras for situation awareness



[*] Image Courtesy: https://www.tesla.com/en_CA/autopilot
[\(https://www.tesla.com/en_CA/autopilot\)](https://www.tesla.com/en_CA/autopilot)

SPHERICAL CNNS (ICLR 2018* BEST PAPER)

... Or you can use few fisheye camera(s)



[*] Image courtesy: DJI-X <https://www.halfchrome.com/dji-360-drone/> (<https://www.halfchrome.com/dji-360-drone/>)

SPHERICAL CNNS (ICLR 2018* BEST PAPER)

Instead ...

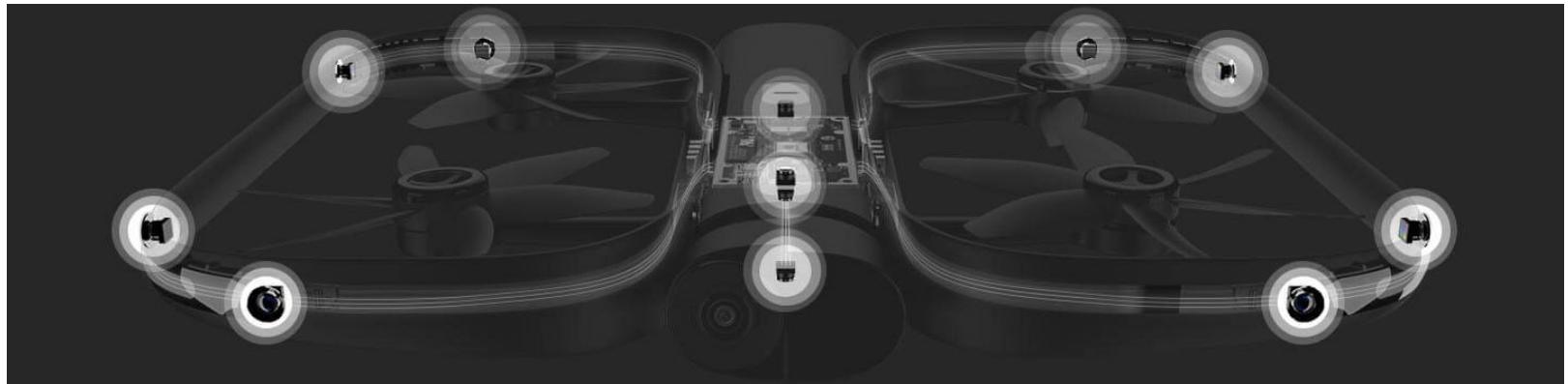


paradoxhorizon

[*] Image Courtesy: Skydio R1 <https://www.skydio.com/technology/> (<https://www.skydio.com/technology/>)

SPHERICAL CNNS (ICLR 2018* BEST PAPER)

Instead ...



[*] Image Courtesy: Skydio R1 <https://www.skydio.com/technology/> (<https://www.skydio.com/technology/>)

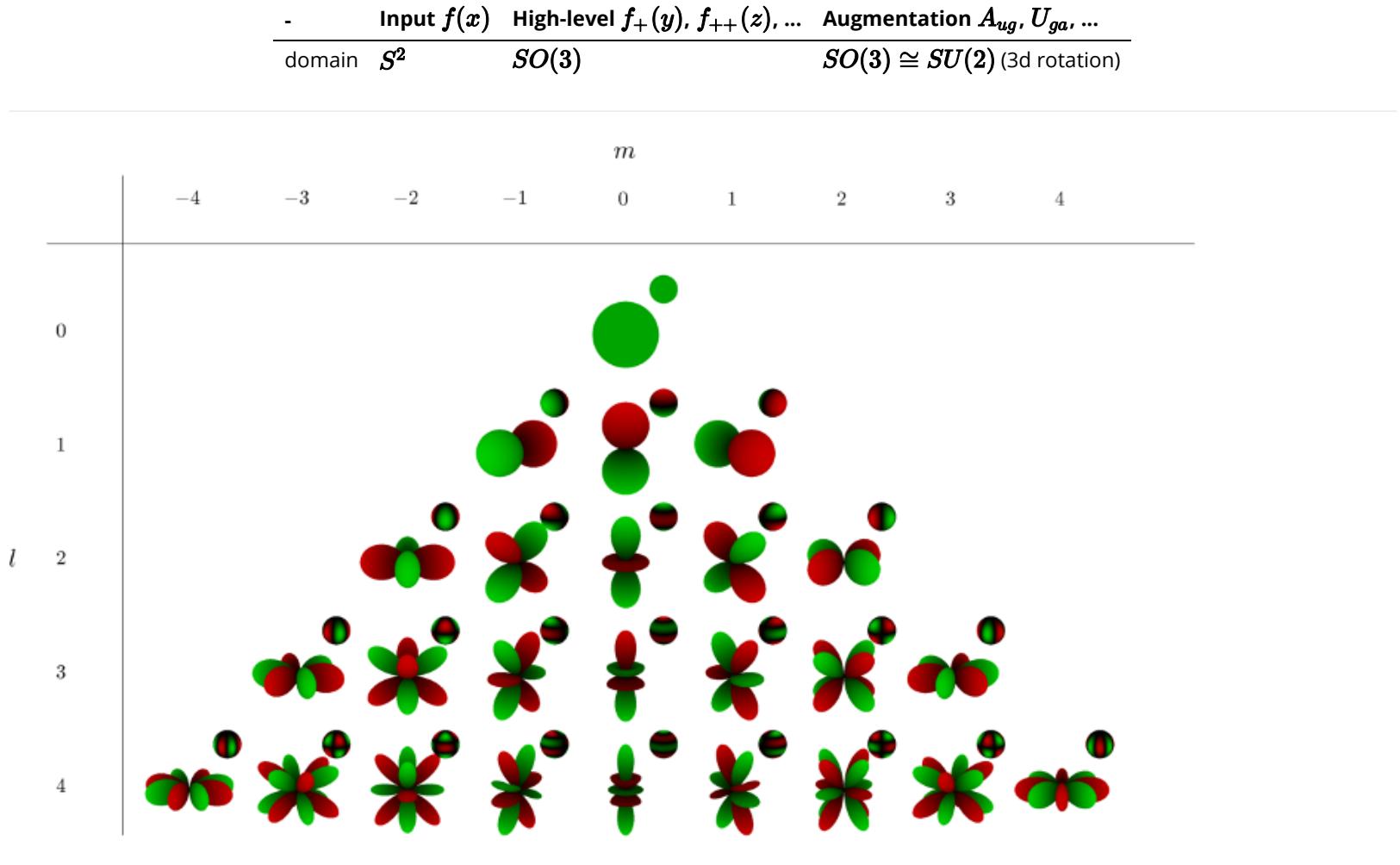
SPHERICAL CNNS (ICLR 2018* BEST PAPER)

-	Input $f(x)$	High-level $f_+(y), f_{++}(z), \dots$	Augmentation A_{ug}, U_{ga}, \dots
domain	S^2	$SO(3)$	$SO(3) \cong SU(2)$ (3d rotation)

- $u_m(\mathbf{x}) \leftarrow D_m(\mathbf{x})$ (Wigner-D function, \mathbf{x} is an Euler-angle tuple or quaternion)
 - collapses to spherical harmonics on the first layer
- GFT $\leftarrow SO(3)$ FFT
- number of coefficients ≤ 25 : $l \in [0, 4]$ (increasing beyond that contributes little to accuracy)

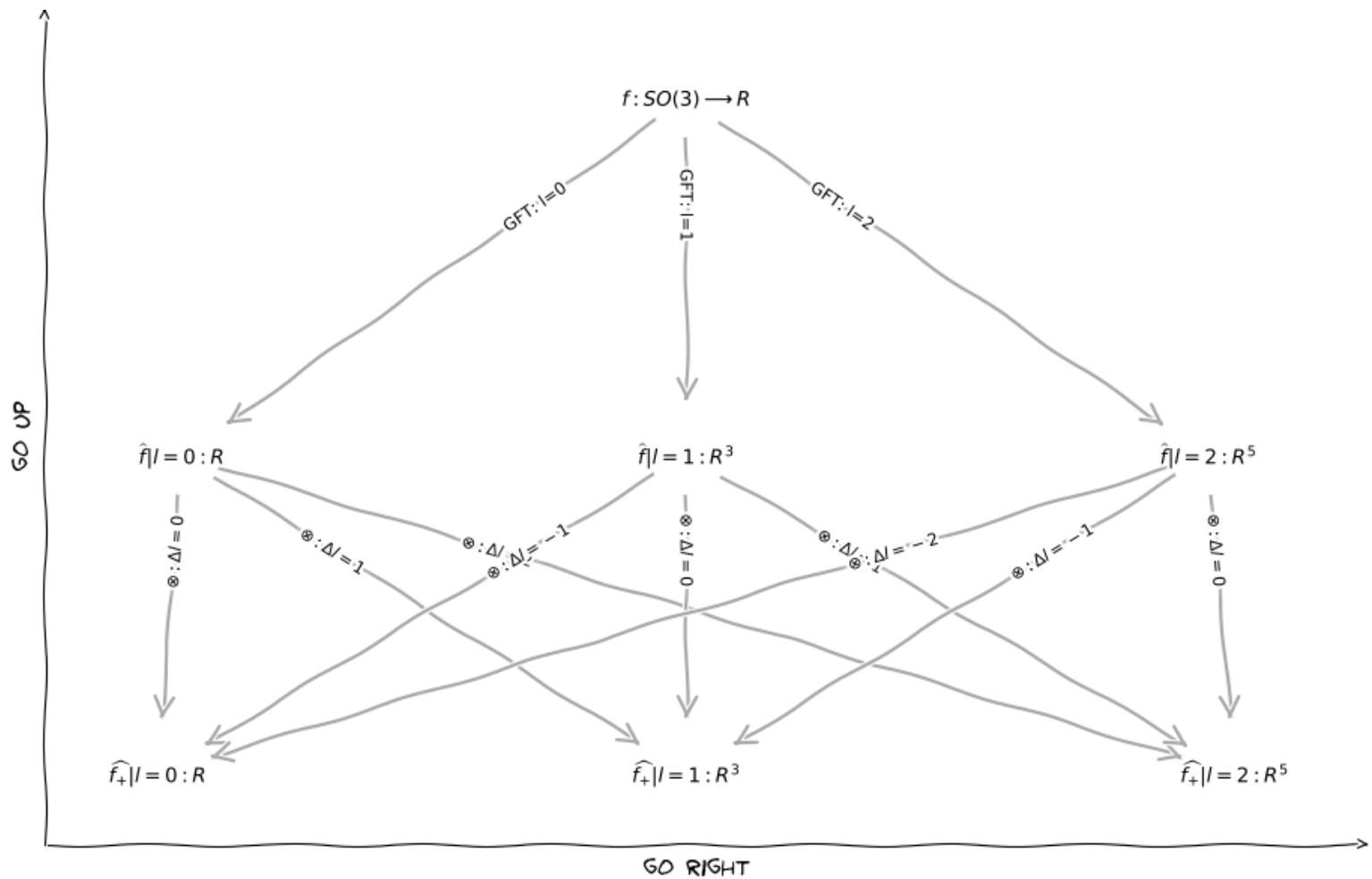
[*] T. S. Cohen, M. Geiger, J. Koehler, and M. Welling, "Spherical CNNs," no. 3, pp. 1–15, 2018.

SPHERICAL CNNS (ICLR 2018* BEST PAPER)



[*] T. S. Cohen, M. Geiger, J. Koehler, and M. Welling, "Spherical CNNs," no. 3, pp. 1–15, 2018.

SPHERICAL CNNS (ICLR 2018* BEST PAPER)



TENSOR FIELD NETWORK (NOT PEER REVIEWED!)

-	Input $f(x)$	High-level $f_+(y), f_{++}(z), \dots$	Augmentation A_{ug}, U_{ga}, \dots
domain	R^3	$O(3)$	$O(3) \cong R^3 \times SO(3)$ (3d translation & rotation)

- $u_m(\mathbf{x}) \leftarrow D_m(\mathbf{x})$ (Wigner-D function, \mathbf{x} is an Euler-angle tuple or quaternion)
 - collapses to spherical harmonics on the first layer
- GFT \leftarrow SO(3) FFT
- number of coefficients $\leftarrow 4: l \in 0, 1$
- C-G transformation is used to accelerate G-conv in frequency space
 - which was not needed in Spherical CNNs due to lack of radial component

TENSOR FIELD NETWORK (NOT PEER REVIEWED!)

