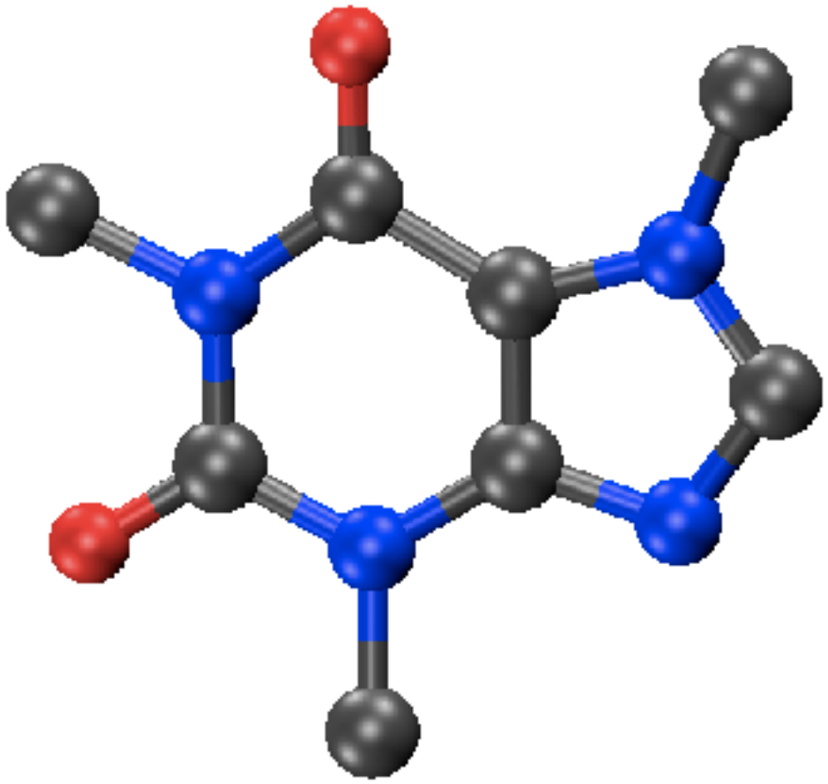
Covariant neural networks for learning in physics and chemistry

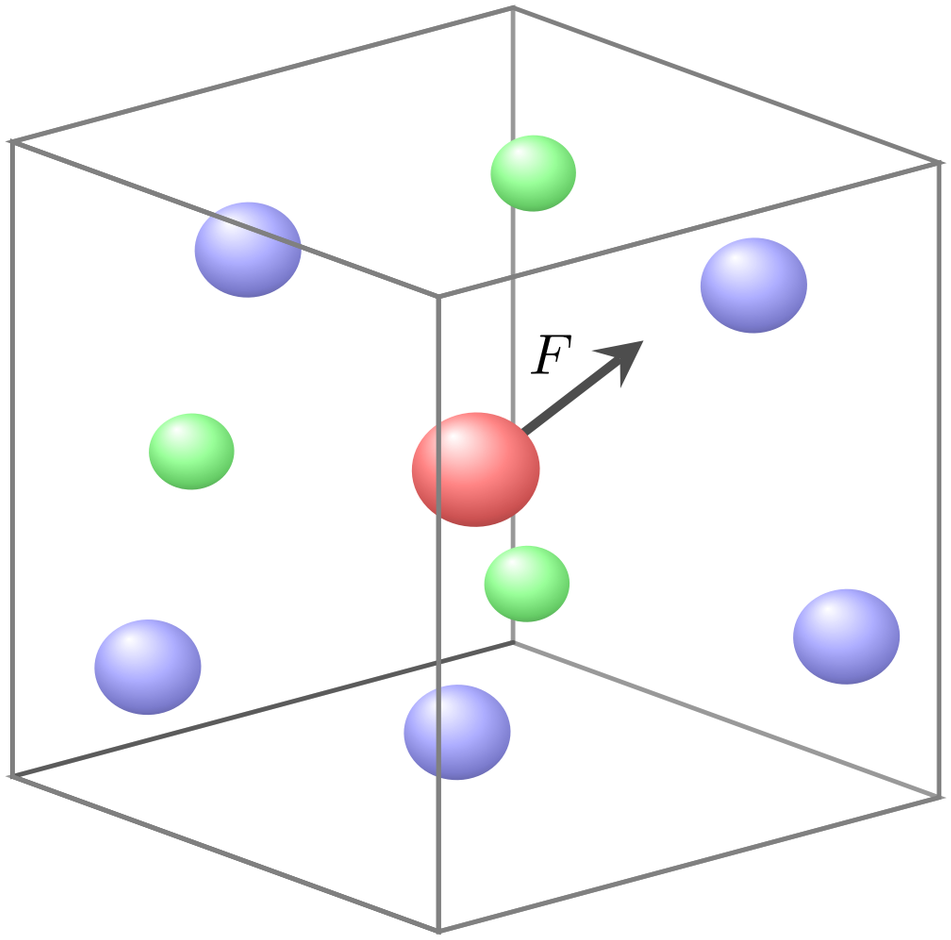
Risi Kondor

FLATIRON INSTITUTE

The University of Chicago

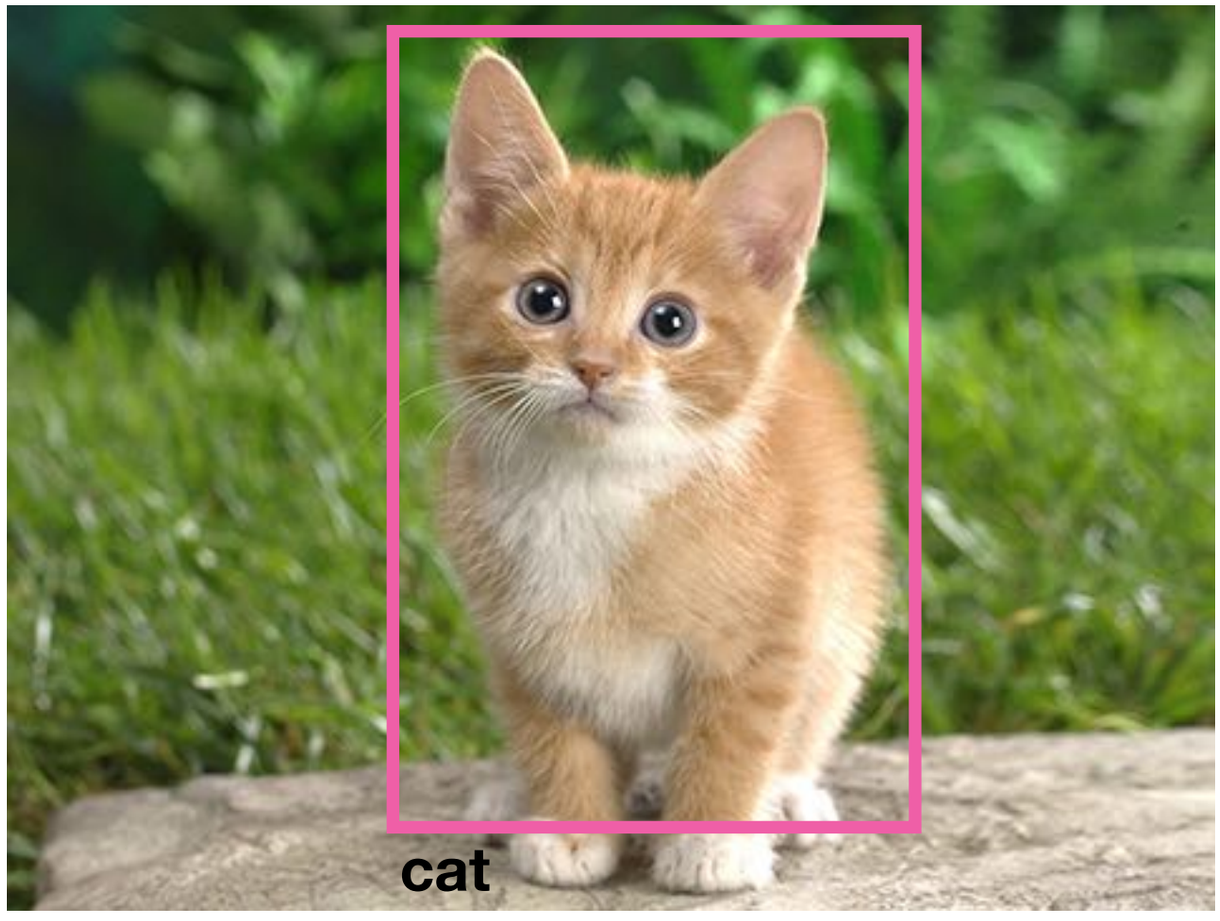


$\phi(G)$



$F(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_m)$

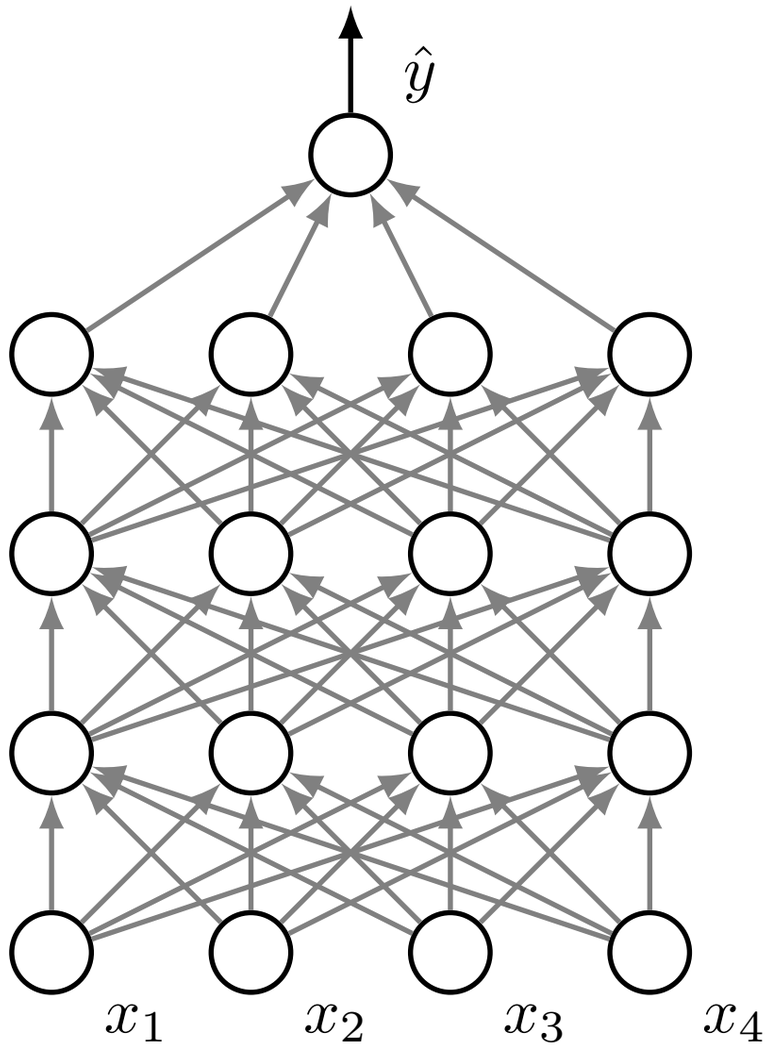
Current capability:



source: NVIDIA

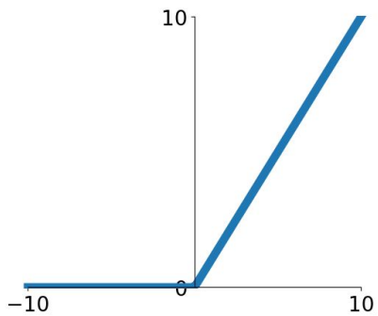
Feed-forward Neural Networks

$$f_{\text{out}} = \sigma\left(\sum_i w_i f_{\text{in}}^{(i)} + b\right)$$

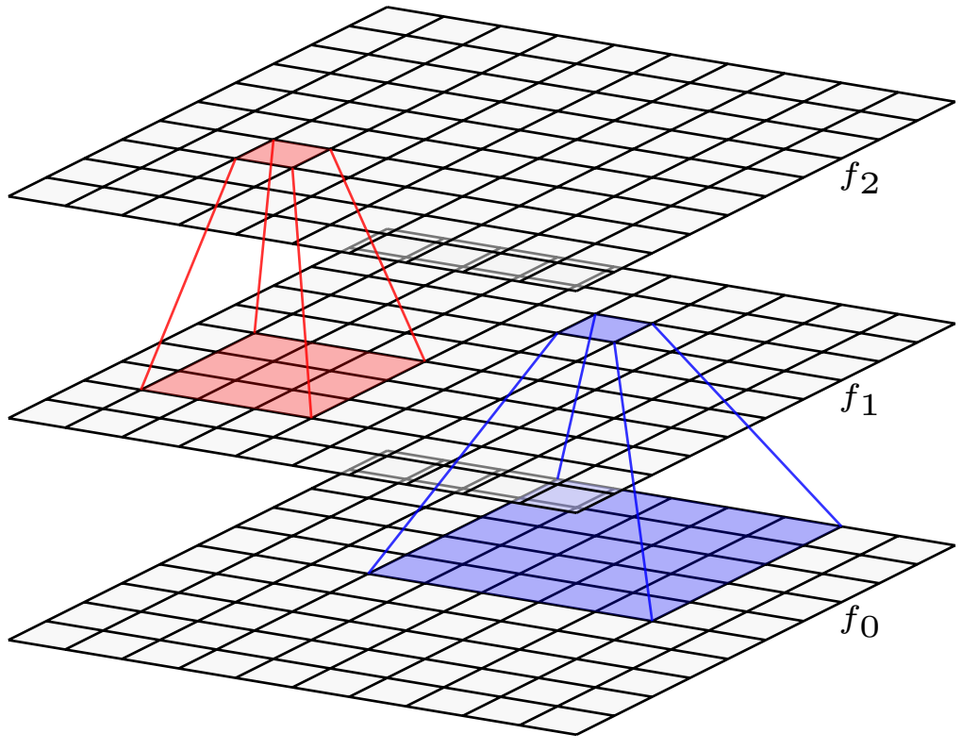


Common choice of nonlinearity:

$$\sigma_{\text{ReLU}}(z) = \max(0, x)$$



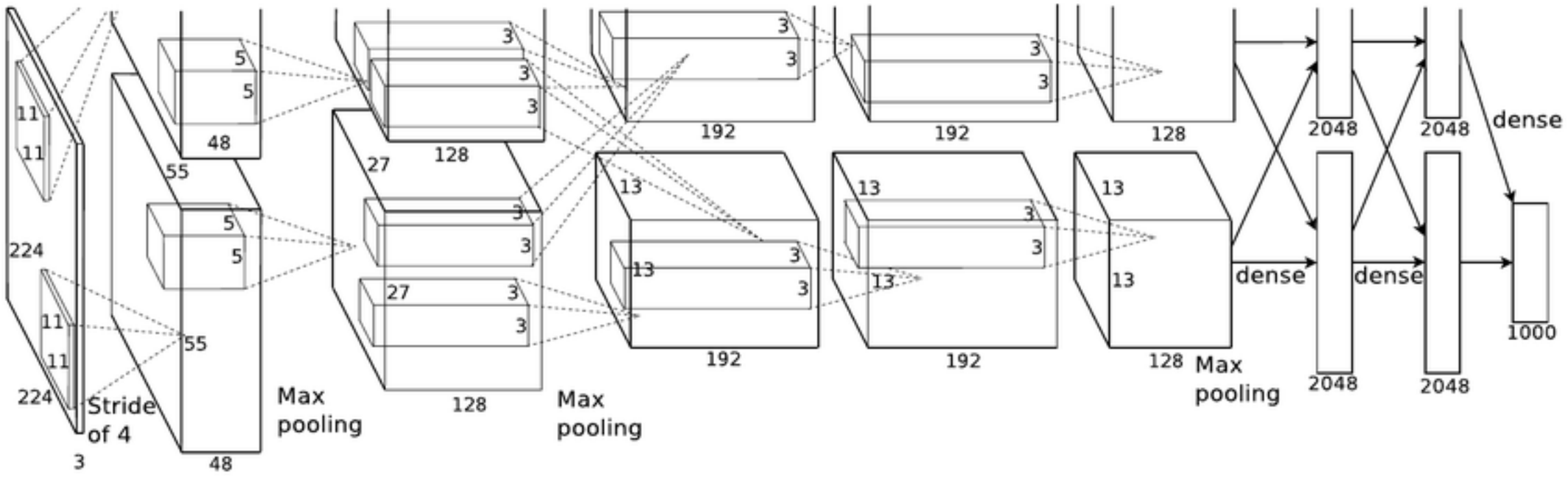
Convolutional Neural Networks



$$\phi_\ell(f_{\ell-1}) = (f_{\ell-1} * g_\ell)(x) = \sum_{y \in \mathbb{Z}^2} f_{\ell-1}(x-y)\, g_\ell(y)$$

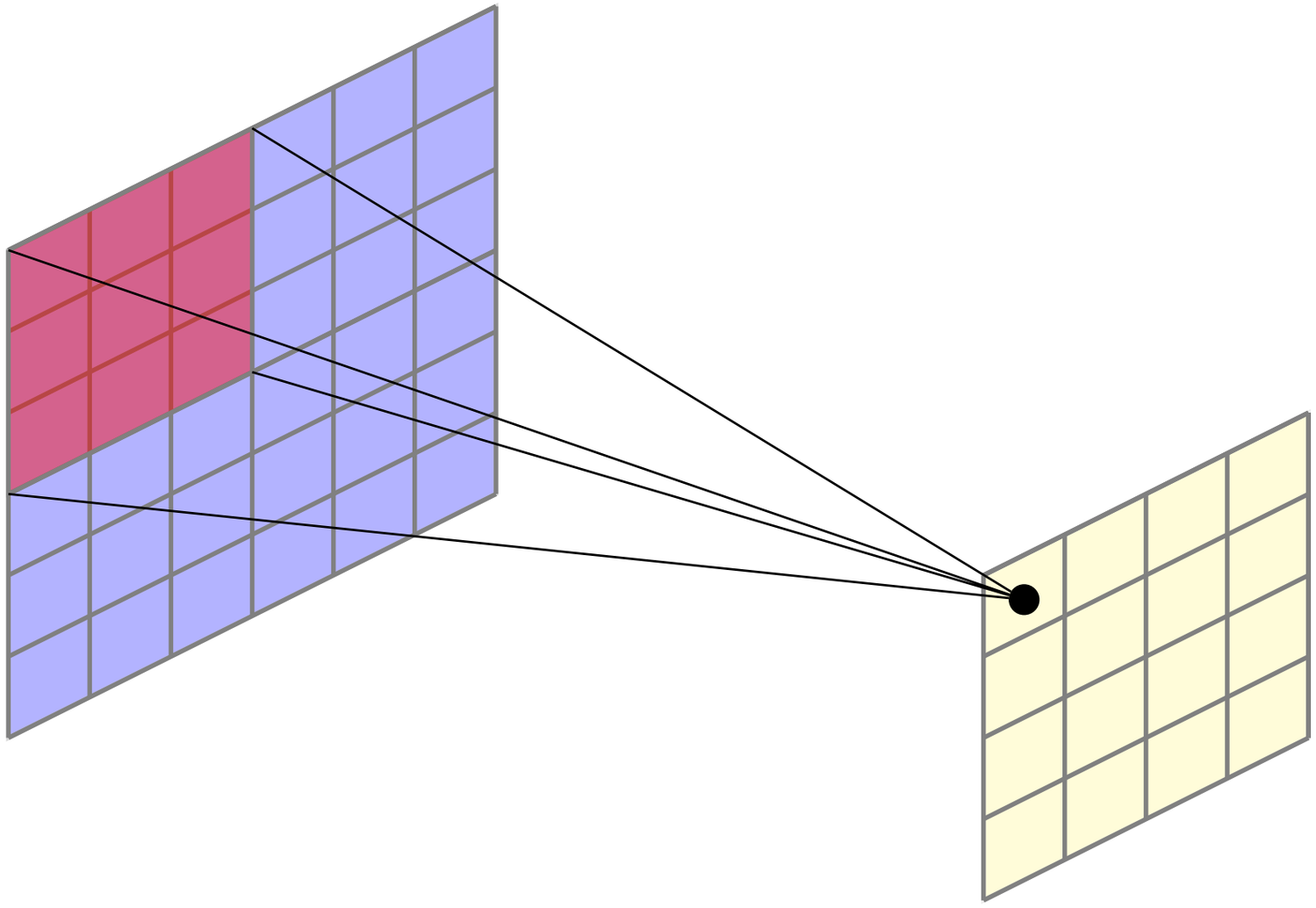
Filter at layer ℓ

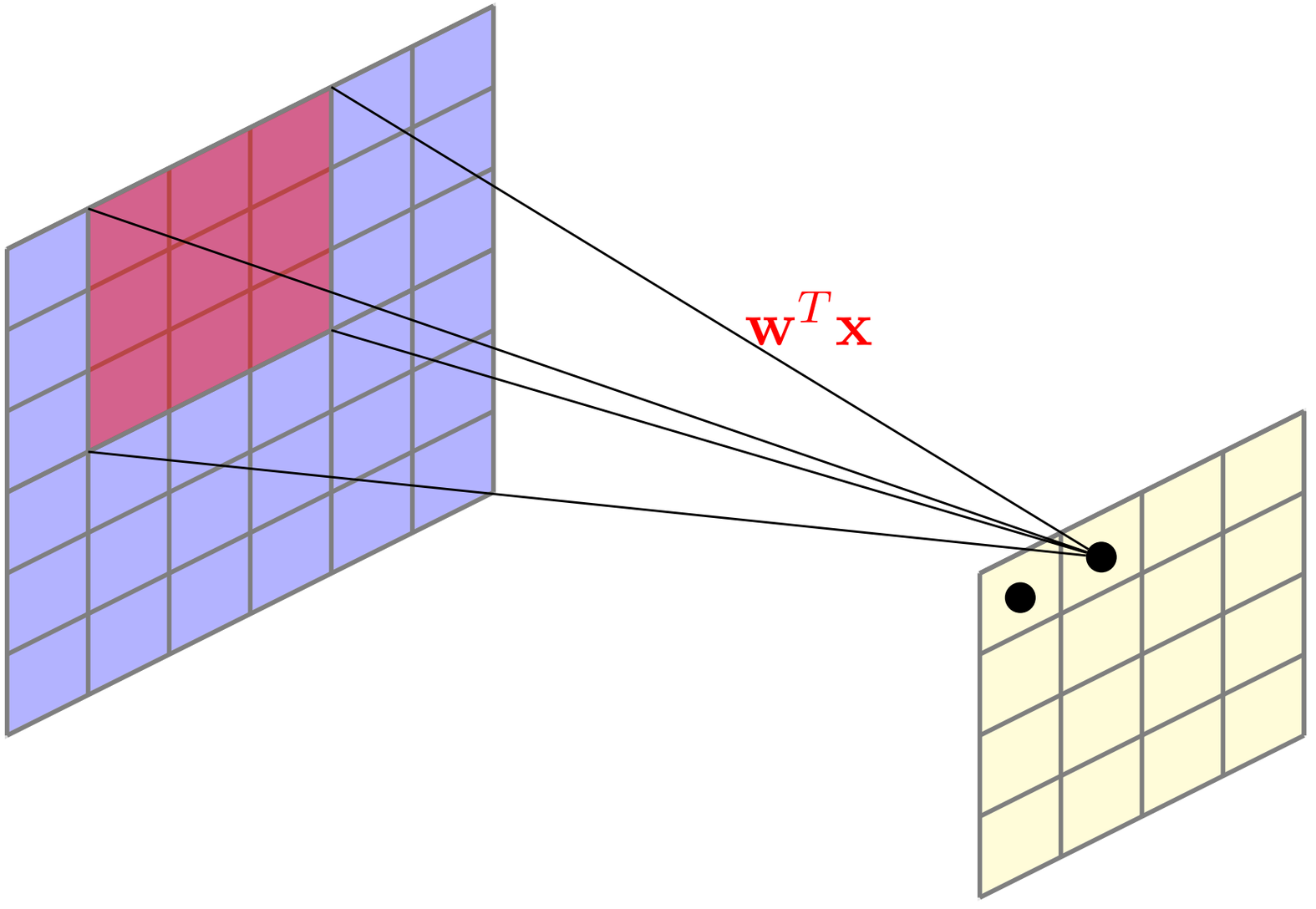
Convolutional Neural Networks



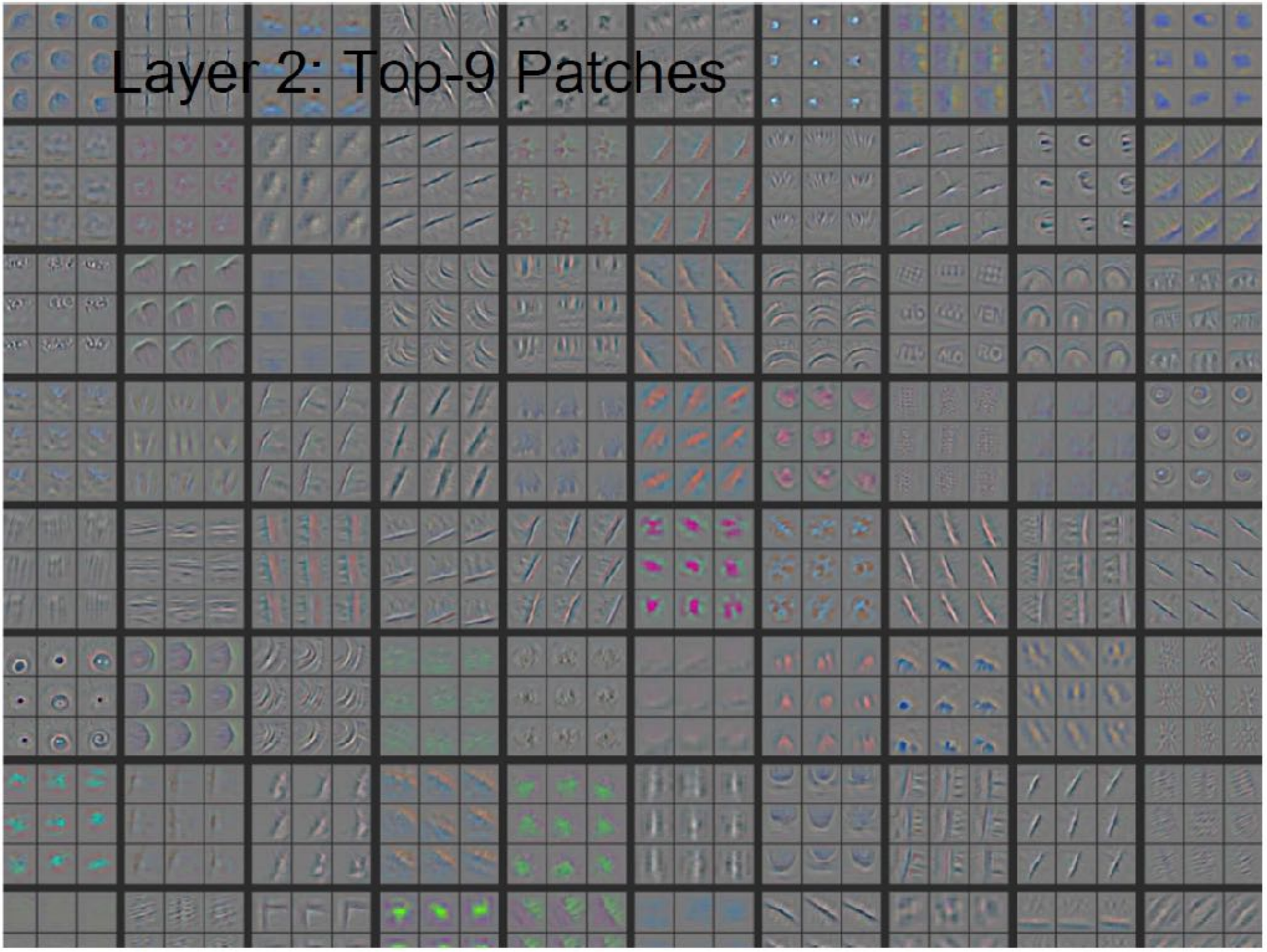
$$f_\ell(x) = \sigma\left(\sum_{\mathbf{y}} f_{\ell-1}(\mathbf{x}-\mathbf{y})\,\chi(\mathbf{y}) + b\right)$$

[LeCun et al, 1989; Krizhevsky, Sutskever & Hinton, 2012]




$\mathbf{w}^T\mathbf{x}$

Layer 2: Top-9 Patches



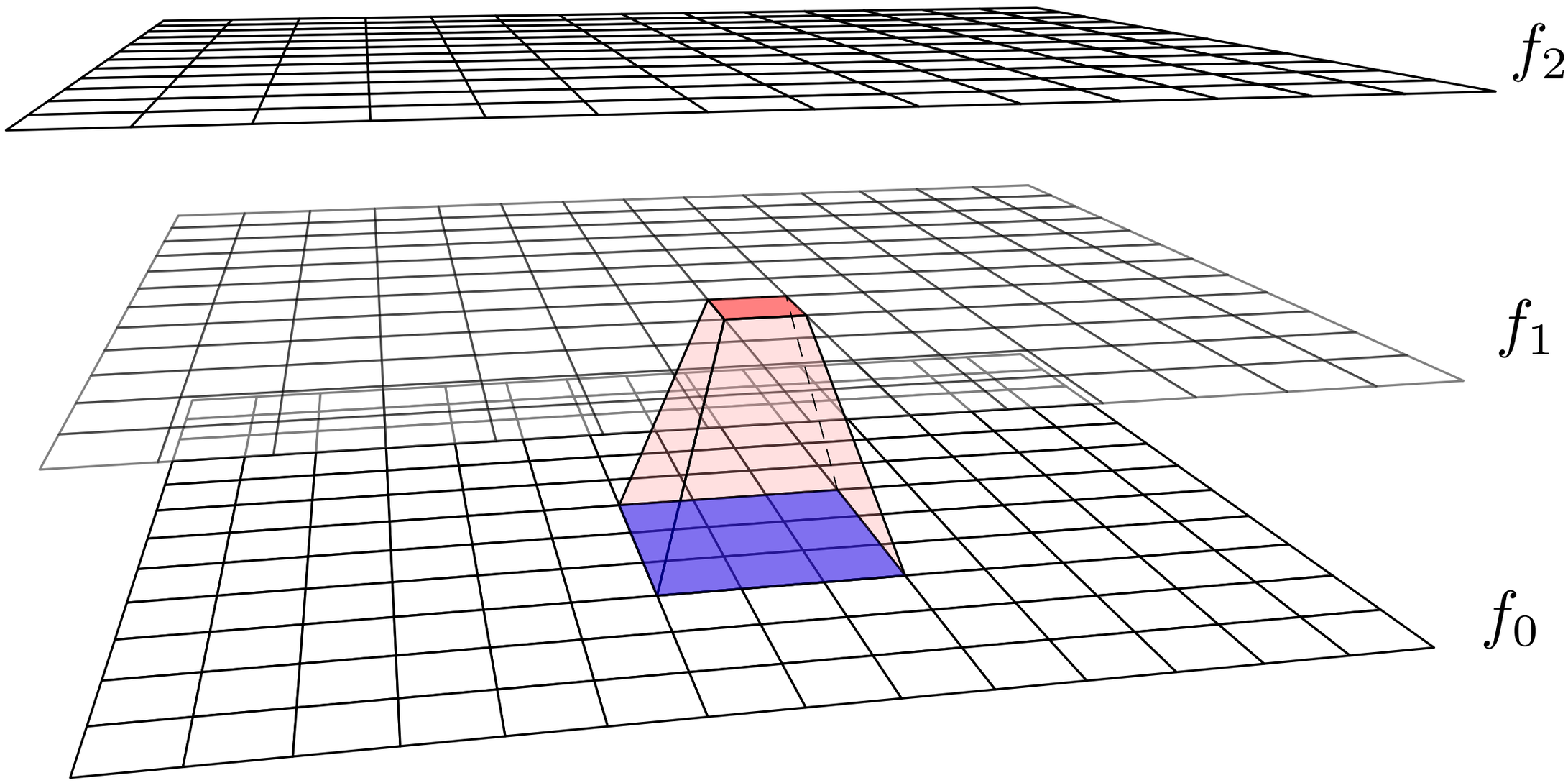
Layer 3: Top-9 Patches

Layer 4: Top-9 Patches

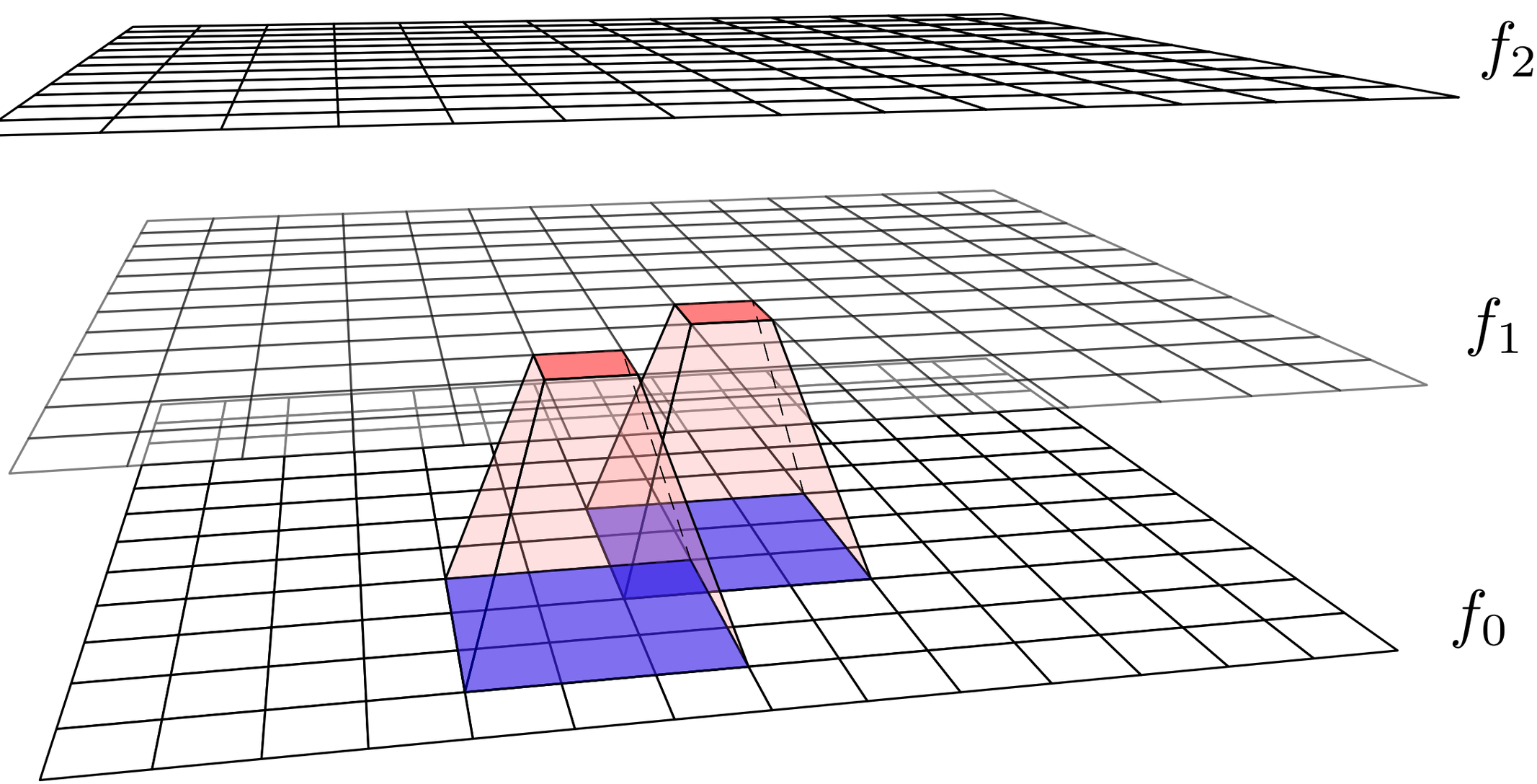
Neural networks

1. Multiscale structure
2. Equivariance

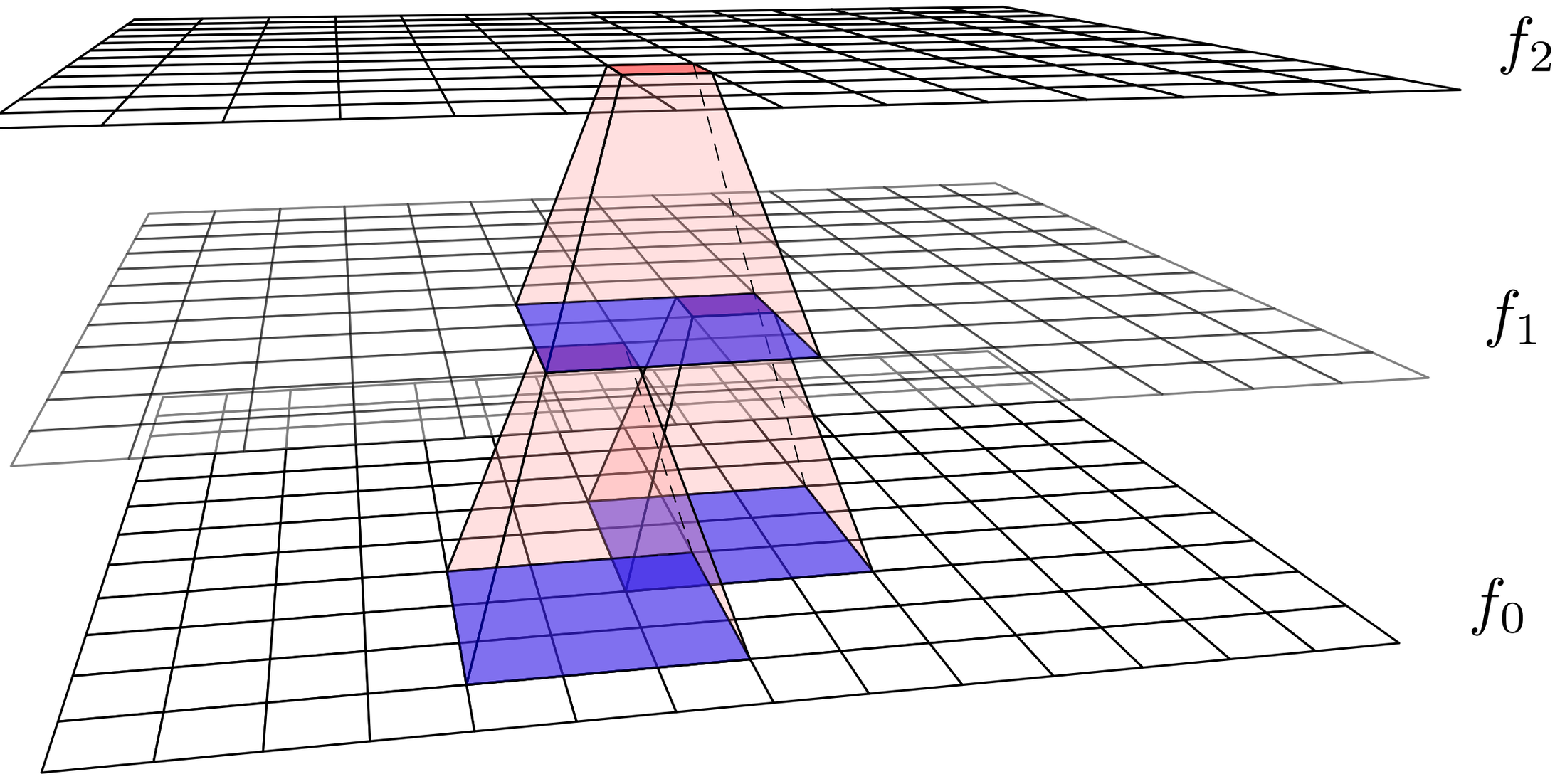
Multiscale structure



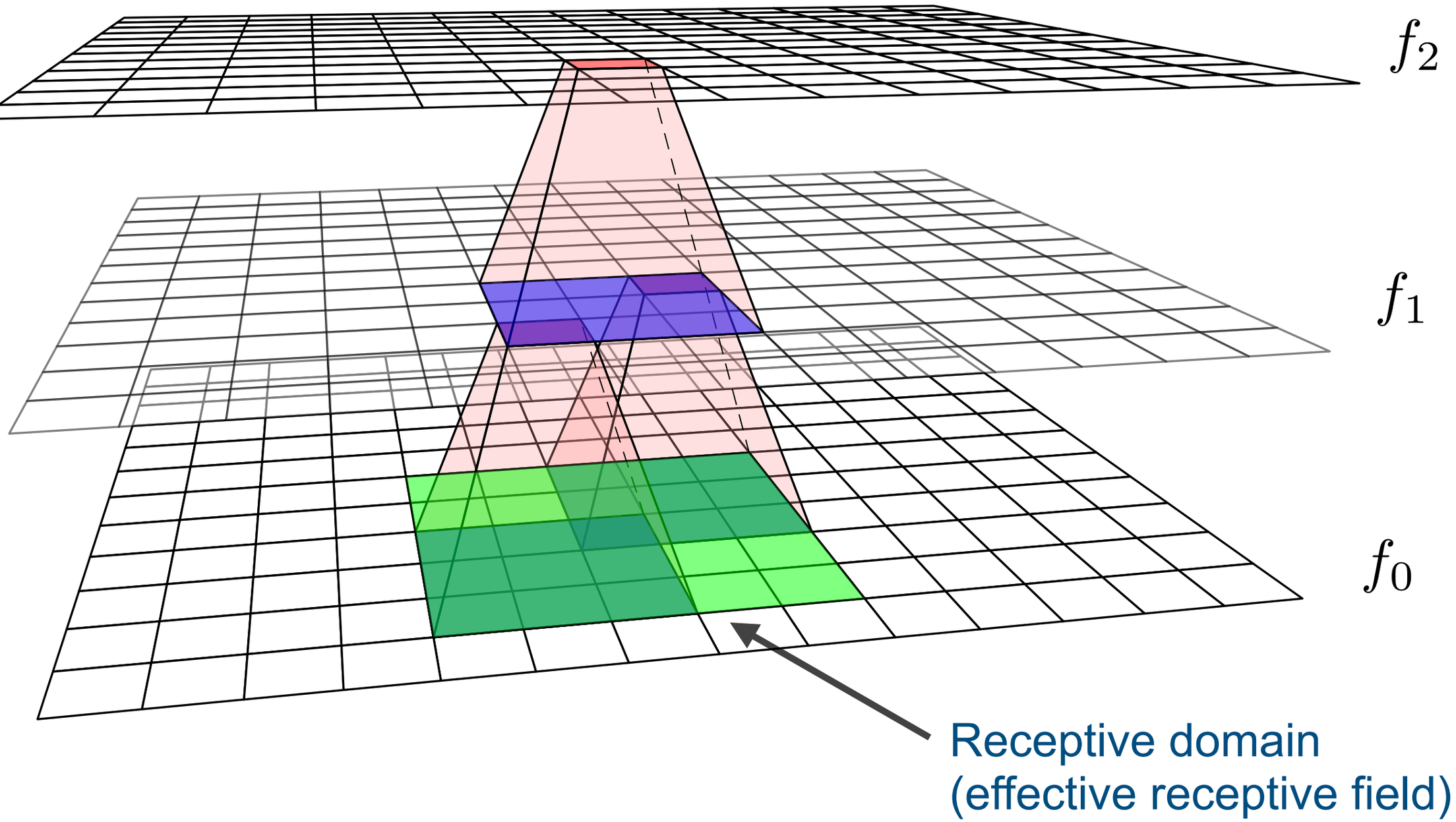
Multiscale structure



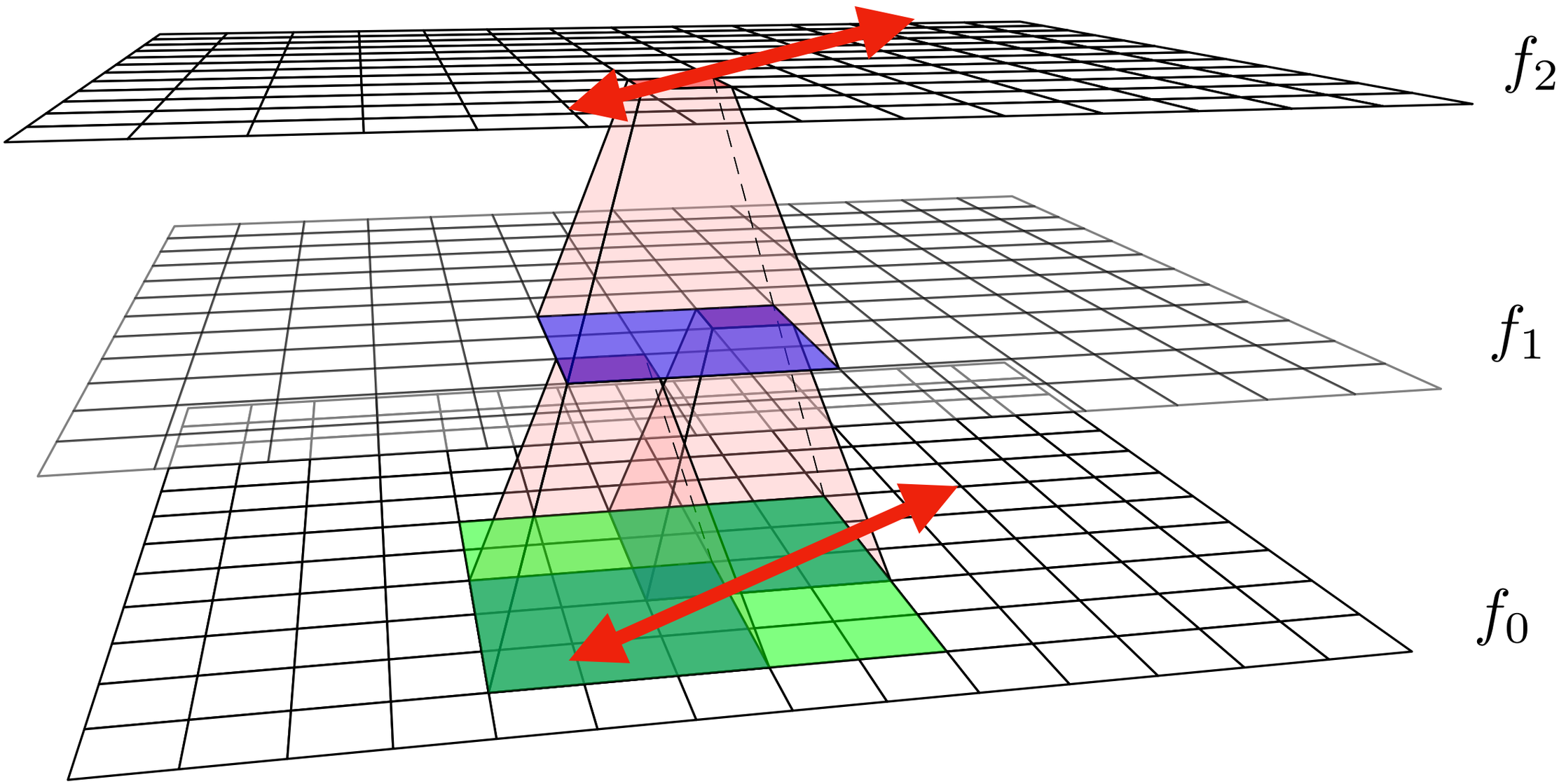
Multiscale structure



Multiscale structure

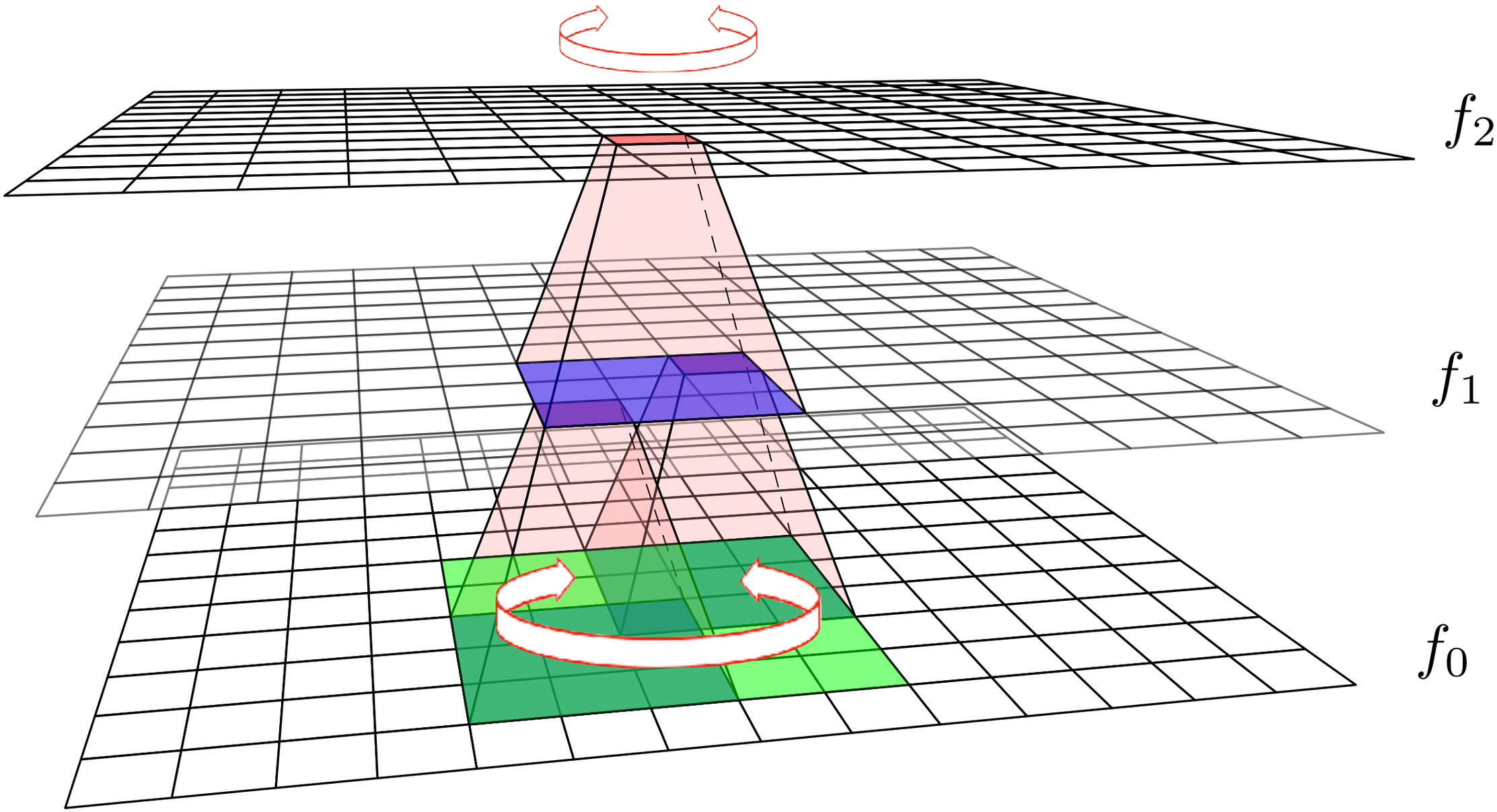


Equivariance (covariance)



[Cohen & Welling: Group equivariant CNNs (ICML 2016)]

Equivariance (covariance)



[Cohen & Welling: Steerable CNNs (ICLR 2017)]

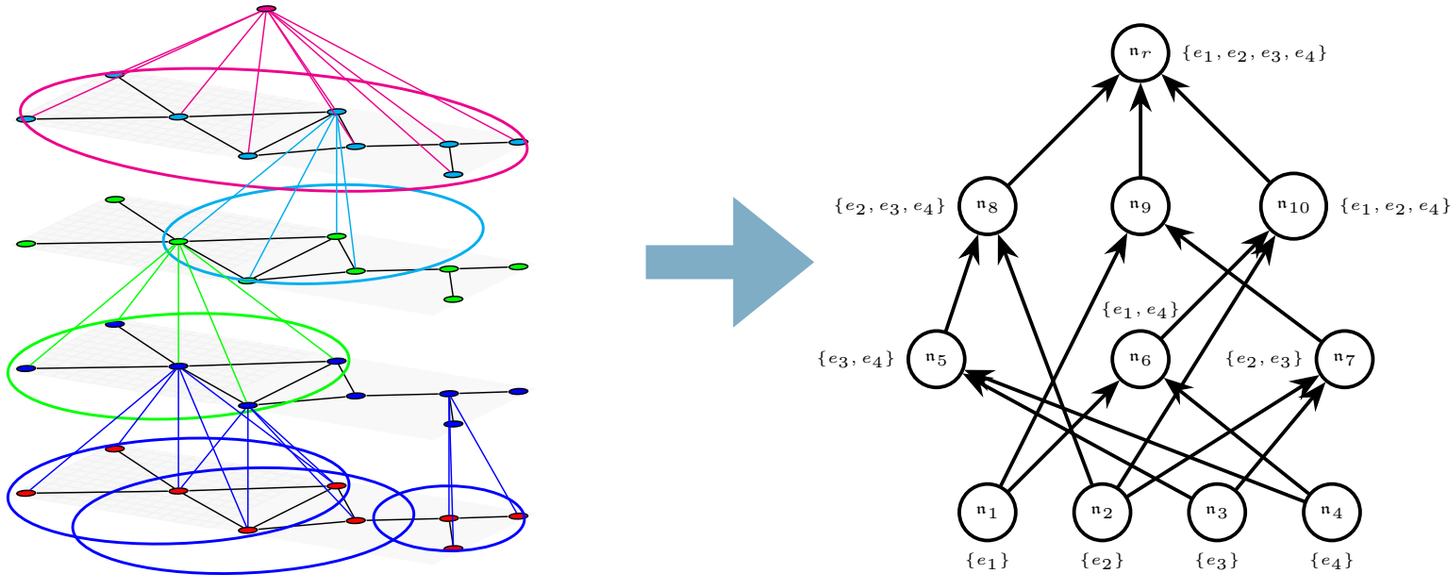
$$
\begin{array}{ccc}
L(\mathcal{X}_1) & \xrightarrow{\mathbb{T}_g^{(1)}} & L(\mathcal{X}_1) \\
\downarrow{\scriptstyle \phi} & & \downarrow{\scriptstyle \phi} \\
L(\mathcal{X}_2) & \xrightarrow{\mathbb{T}_g^{(2)}} & L(\mathcal{X}_2)
\end{array}
$$

1. What is the analog of convolution for molecular graphs?

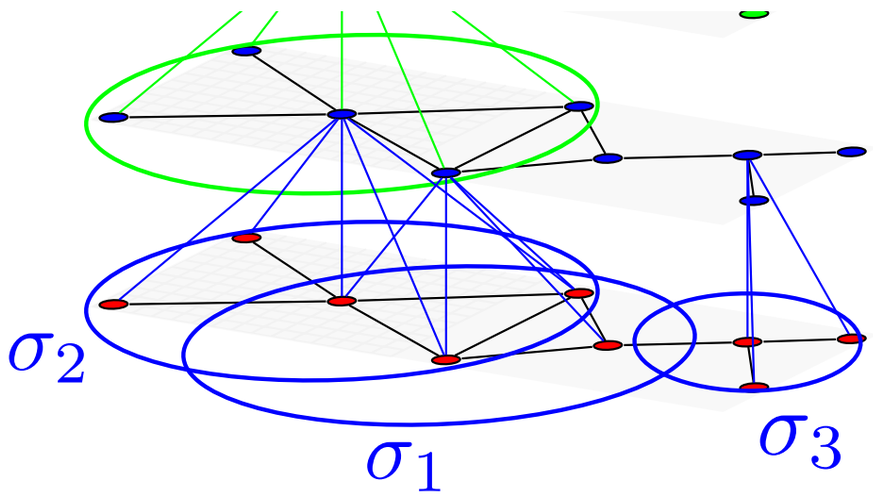
Covariant Compositional Networks
[Hy, Trivedi, Pan, Anderson & K, JCP 18]

Covariant Compositional Networks (CCNs)

1. Multiscale structure



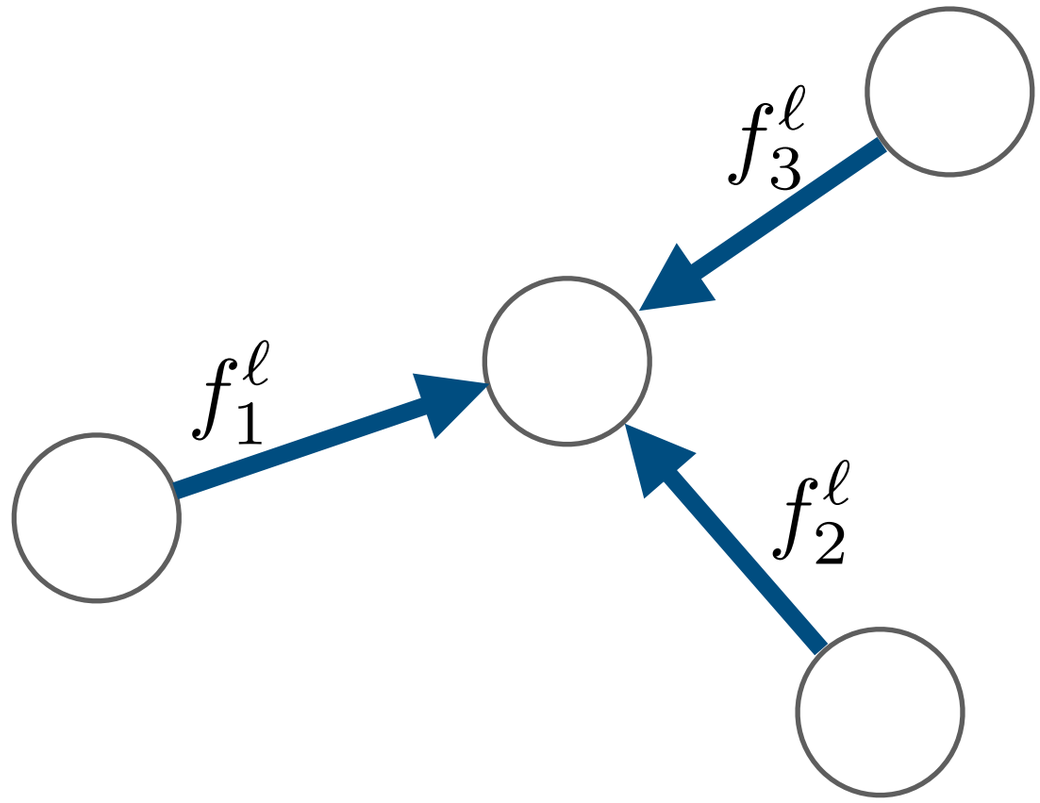
2. Covariance to permutations



$$F \mapsto \rho(\sigma) F$$

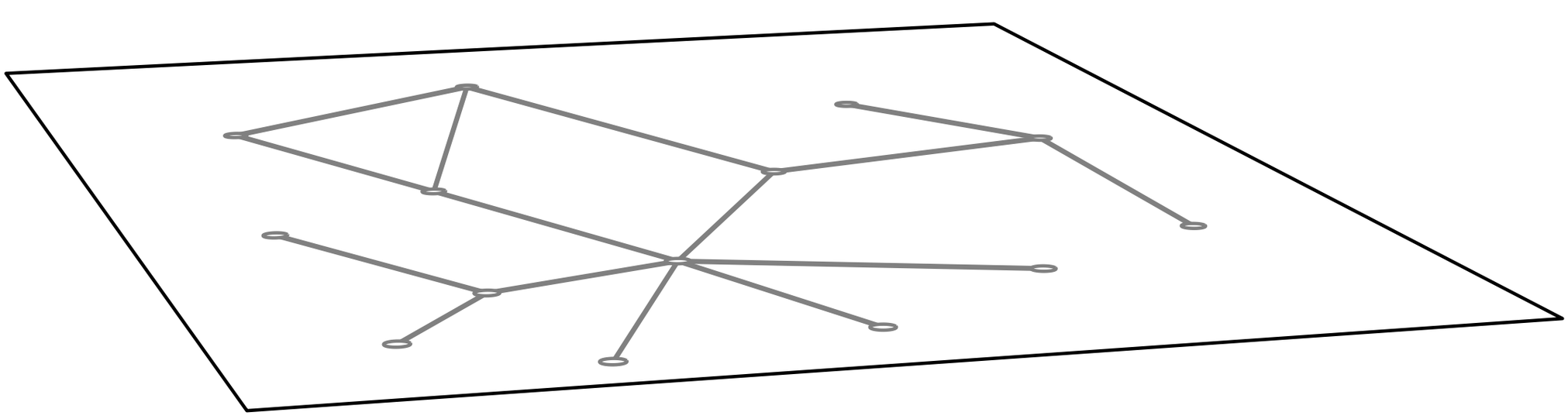
Message passing neural networks (MPNNs)

$$f_i^{\ell+1} = \xi\left(W \sum_{j \in \mathcal{N}(i)} f_j^{\ell} + b\right)$$

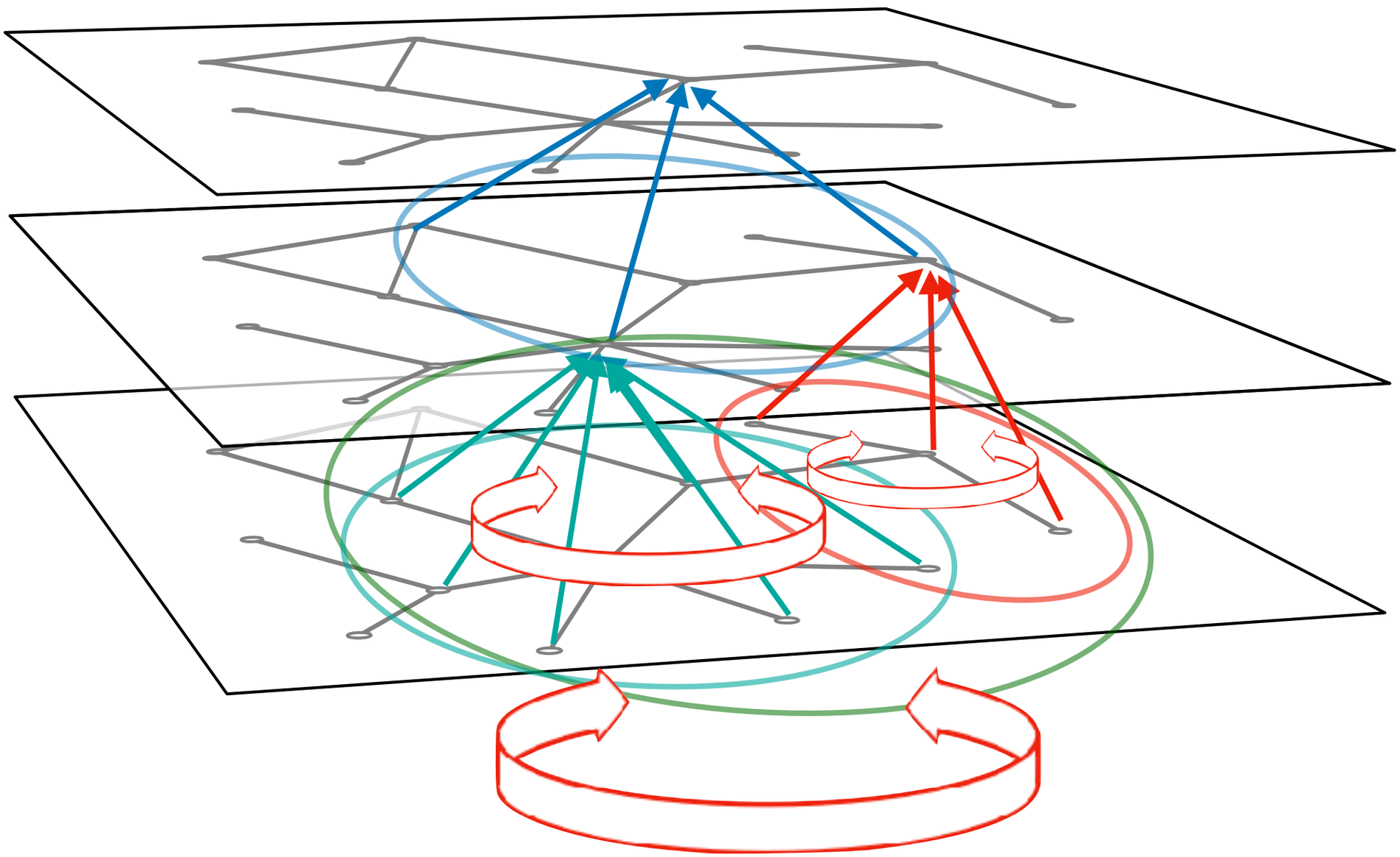


[Gilmer et al, ’17] [Kriege, ’16] [Niepert, ’16] [Duvenaud et al., ’15] [Dai, Dai & Song, ’16]

Multiscale structure in graphs



Multiscale structure and covariance in graphs



[Son, Trivedi, Pan, Anderson & K: Predicting molecular properties with covariant compositional networks (JCP 2018)]

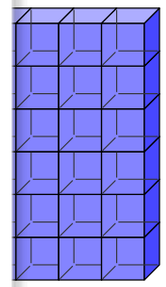
Permutation covariant activations

How does a given activation transform when its receptive field is subjected to a permutation σ?

Scalar activation
(0th

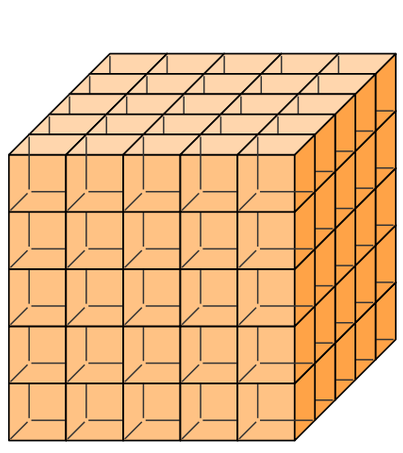
Vector activation
(1st

Matrix activation
(2nd r covariant)



$P_\sigma F_i P_\sigma^\top$

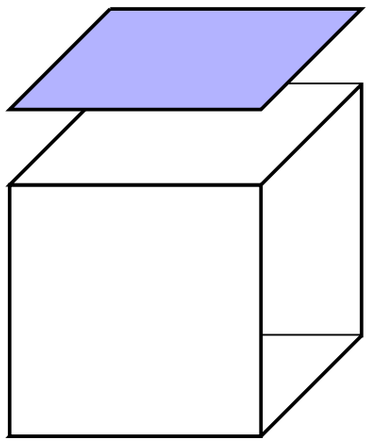
General tensor activation
(kth order covariant)



$$F_{i_1,i_2,\ldots,i_k} \overset{\sigma}{\longmapsto} [P_\sigma]_{i_1}^{j_1} [P_\sigma]_{i_2}^{j_2} \cdots [P_\sigma]_{i_k}^{j_k} F_{j_1,j_2,\ldots,j_k}$$

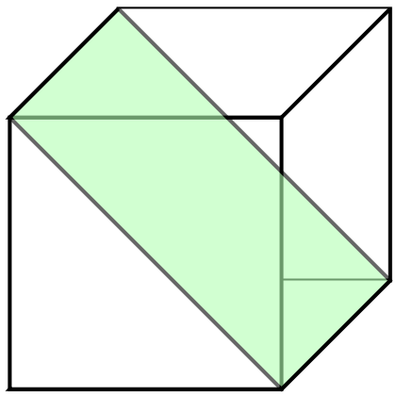
Permutation covariant operations:

1. Projections



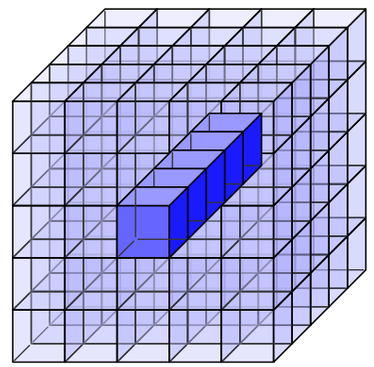
$$C_{i,j} = \sum_a A_{a,i,j}$$

2. Diagonals



$$C_{i,j} = \sum_i A_{i,i,j}$$

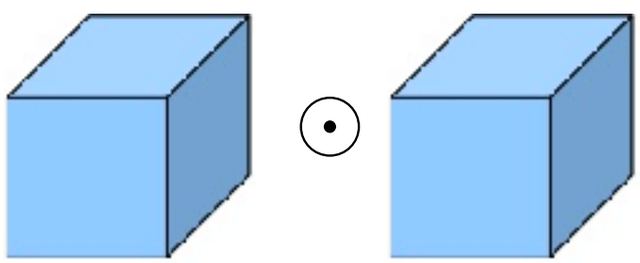
3. Contractions



$$C_k = \sum_{i,j} A_{i,j,k}$$

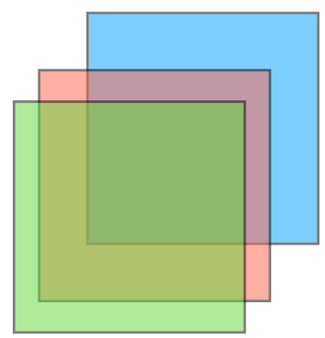
Permutation covariant operations:

4. Hadamard products



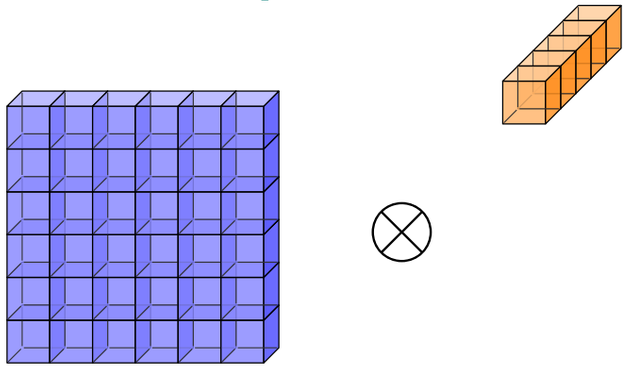
$$C_{i,j,k} = A_{i,j,k} B_{i,j,k}$$

5. Stacking



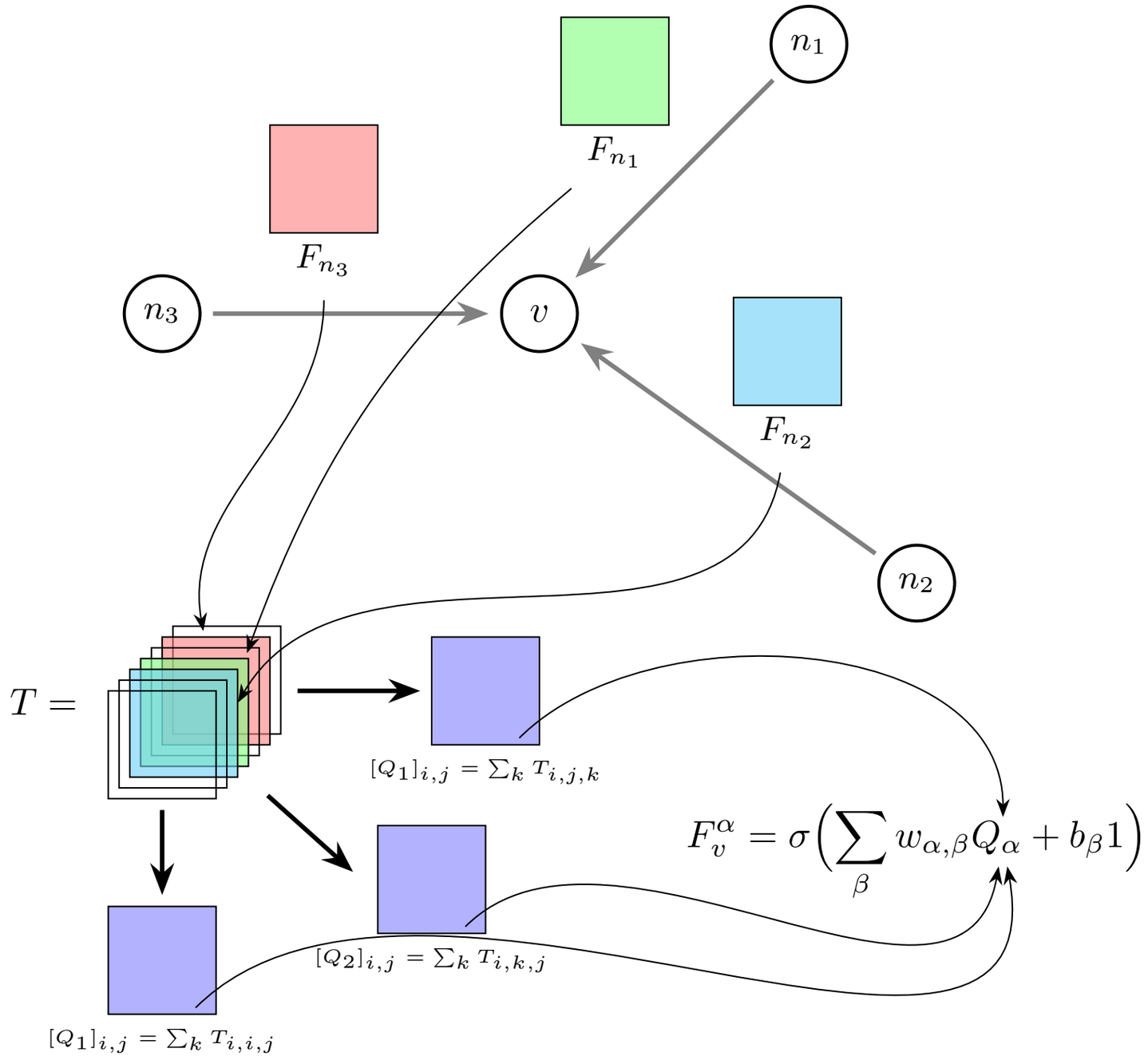
$$C_{i,j,k} = A^{(k)}_{i,j}$$

6. Tensor products



$$C_{i,j,k} = A_{i,j} B_k$$

Second order vertex aggregation



Harvard Clean Energy Project results

Method	Train MAE	Train RMSE	Test MAE	Test RMSE
Lasso	0.863	1.190	0.867	1.437
Ridge Regression	0.849	1.164	0.854	1.376
Random Forest	0.999	1.331	1.004	1.799
Gradient Boosted Tree	0.676	0.939	0.704	1.005
Weisfeiler-Lehman Graph Kernel	0.805	1.111	0.805	1.096
Neural Graph Fingerprint	0.848	1.187	0.851	1.177
Learning Convolution Neural Network	0.704	0.972	0.718	0.973
CCN 2D	0.562	0.773	0.570	0.773

[Duvenaud et al., 2015] [Kriege, 2016] [Niepert, 2016]
[Hachmann et al., 2011]

QM9 results

	WLGK	NGF	PSCN	CCN 2D
α (Bohr^3)	3.75	3.51	1.63	**1.30**
C_v (cal/(mol K))	2.39	1.91	1.09	**0.93**
G (eV)	4.84	4.36	3.13	**2.75**
GAP (eV)	0.92	0.86	0.77	**0.69**
H (eV)	5.45	4.92	3.56	**3.14**
HOMO (eV)	0.38	0.34	0.30	**0.23**
LUMO (eV)	0.89	0.82	0.75	**0.67**
μ (Debye)	1.03	0.94	0.81	**0.72**
ω_1 (cm^{-1})	192.16	168.14	152.13	**120.10**
R_2 (Bohr^2)	154.25	137.43	61.70	**53.28**
U (eV)	5.41	4.89	3.54	**3.02**
U_0 (eV)	5.36	4.85	3.50	**2.99**
ZPVE (eV)	0.51	0.45	0.38	**0.35**

[Hy et al, JCP 18]

2. General theory of equivariant neural networks

Theorem:

A feed-forward neural network is equivariant to the action of a compact group G if and only if the linear operation in each layer is of the form

$$\phi_\ell(f_{\ell-1}) = f_{\ell-1} * g_\ell.$$

where $*$ denotes the generalization of convolution to compact groups, defined

$$(f * g)(u) = \int_G f(uv^{-1})\, g(v)\, d\mu(v)$$

[K and Trivedi: On the generalization of convolution… (ICML 2018)]

Convolution on groups

$$\widehat{f}(k) = \int f(x)\, e^{-ikx}\, dx$$

$$\widehat{(f * \chi)}(k) = \widehat{f}(k) \cdot \widehat{\chi}(k)$$

$$\widehat{f}(\rho) = \int f(x)\, \rho(x)\, d\mu(x)$$

$$(f * g)(u) = \int_G f(uv^{-1})\, g(v)\, d\mu(v)$$

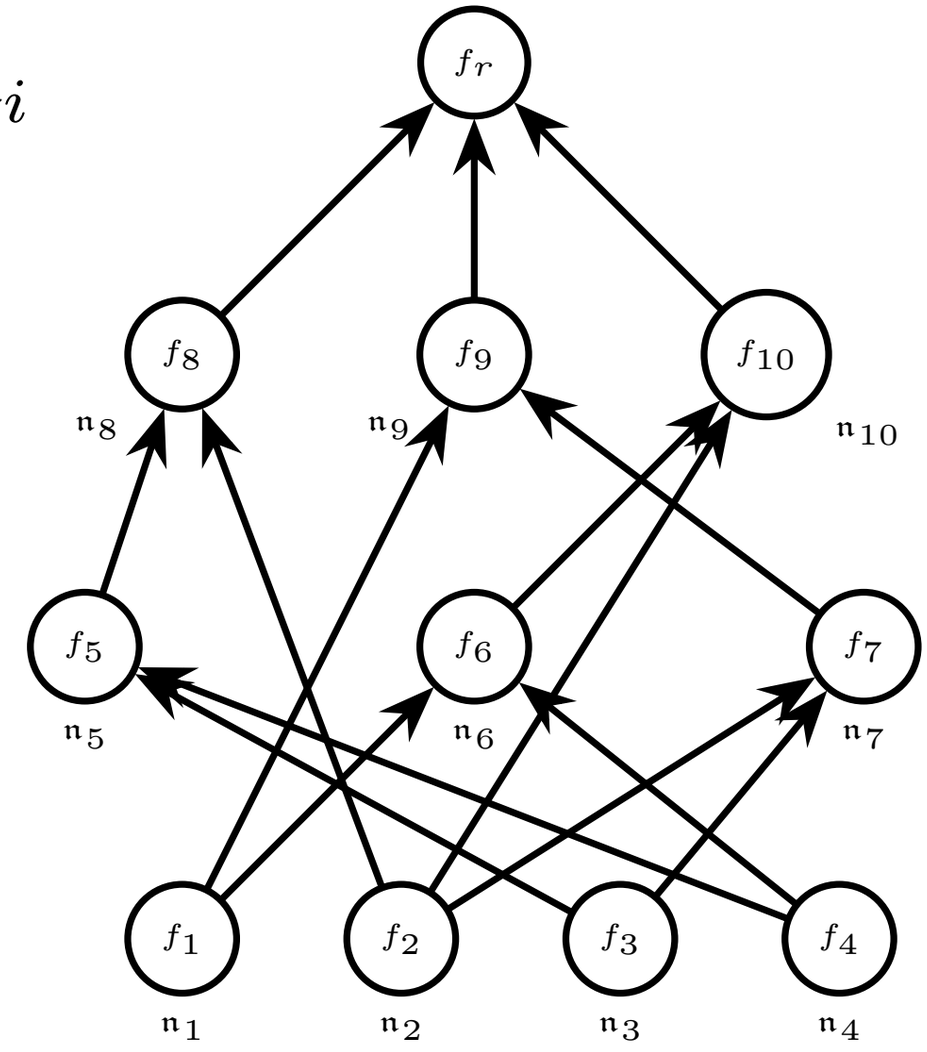
$$\widehat{(f * \chi)}(\rho) = \widehat{f}(\rho) \cdot \widehat{\chi}(\rho)$$

matrix multiplication

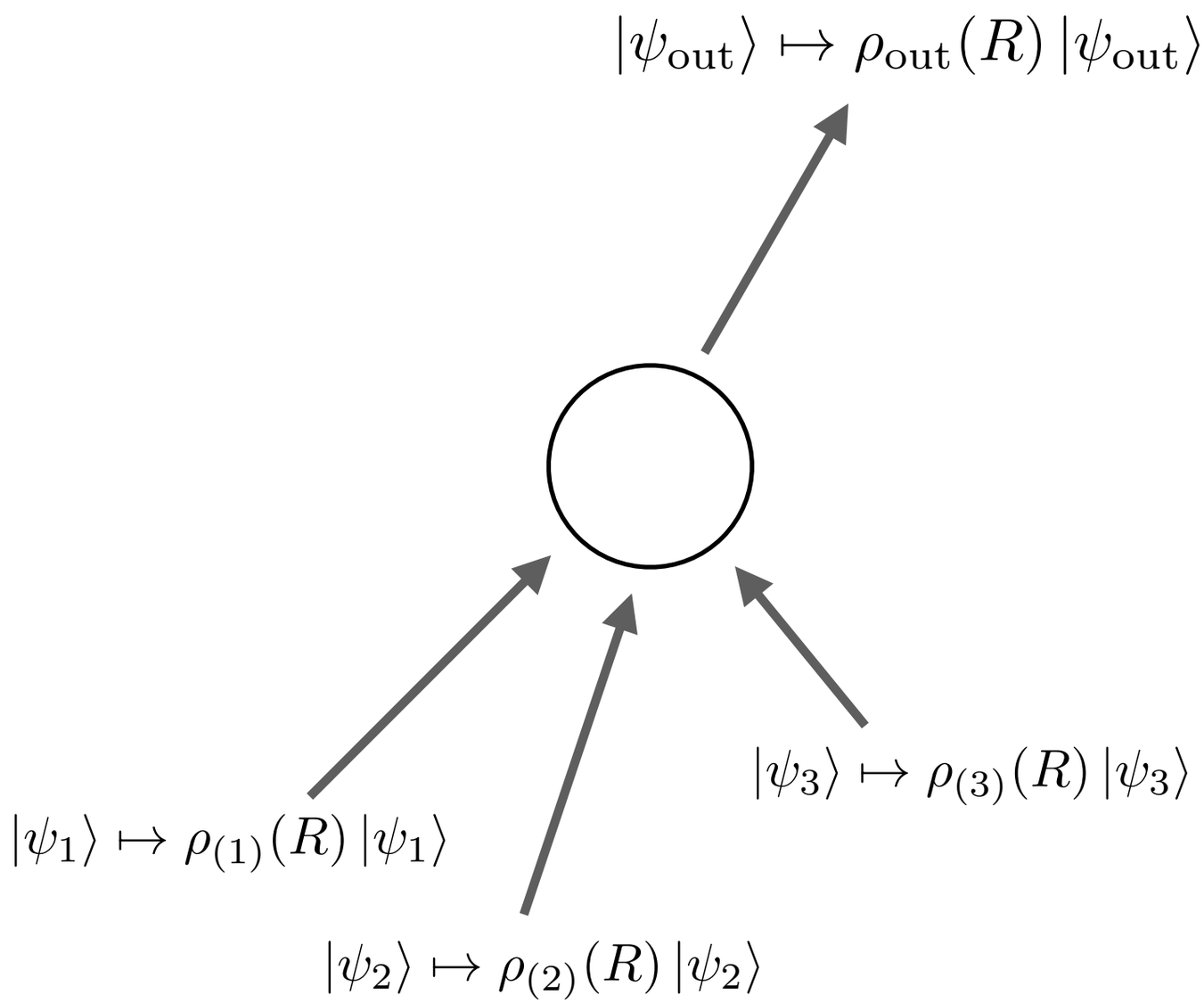
Fourier space neural networks

1. Each node in the graph is a neuron $\mathfrak{n}_i$ and its activation f_i is covariant to the action of G.

2. Each activation f_i is stored in Fourier space.



[Reisert & Burkhardt, 2019] [Masci et al, 2015] [Cohen & Welling, 2016] [Thomas et al, 2018] [Weiler et al, 2018] [Esteves, Allen-Blanchette, Makadia, Daniilidis, 2017] [Cohen, Weiler, Kicanaoglu & Welling, 2019] [Cohen, Geiger, Weiler, 2018]


$|\psi_{\text{out}}\rangle \mapsto \rho_{\text{out}}(R)\,|\psi_{\text{out}}\rangle$
$|\psi_1\rangle \mapsto \rho_{(1)}(R)\,|\psi_1\rangle$
$|\psi_2\rangle \mapsto \rho_{(2)}(R)\,|\psi_2\rangle$
$|\psi_3\rangle \mapsto \rho_{(3)}(R)\,|\psi_3\rangle$

Decomposability

$|\psi\rangle$

$\ell = 0$ ρ_0

ψ_1^0 ψ_2^0 ψ_3^0

$\ell = 1$ ρ_1

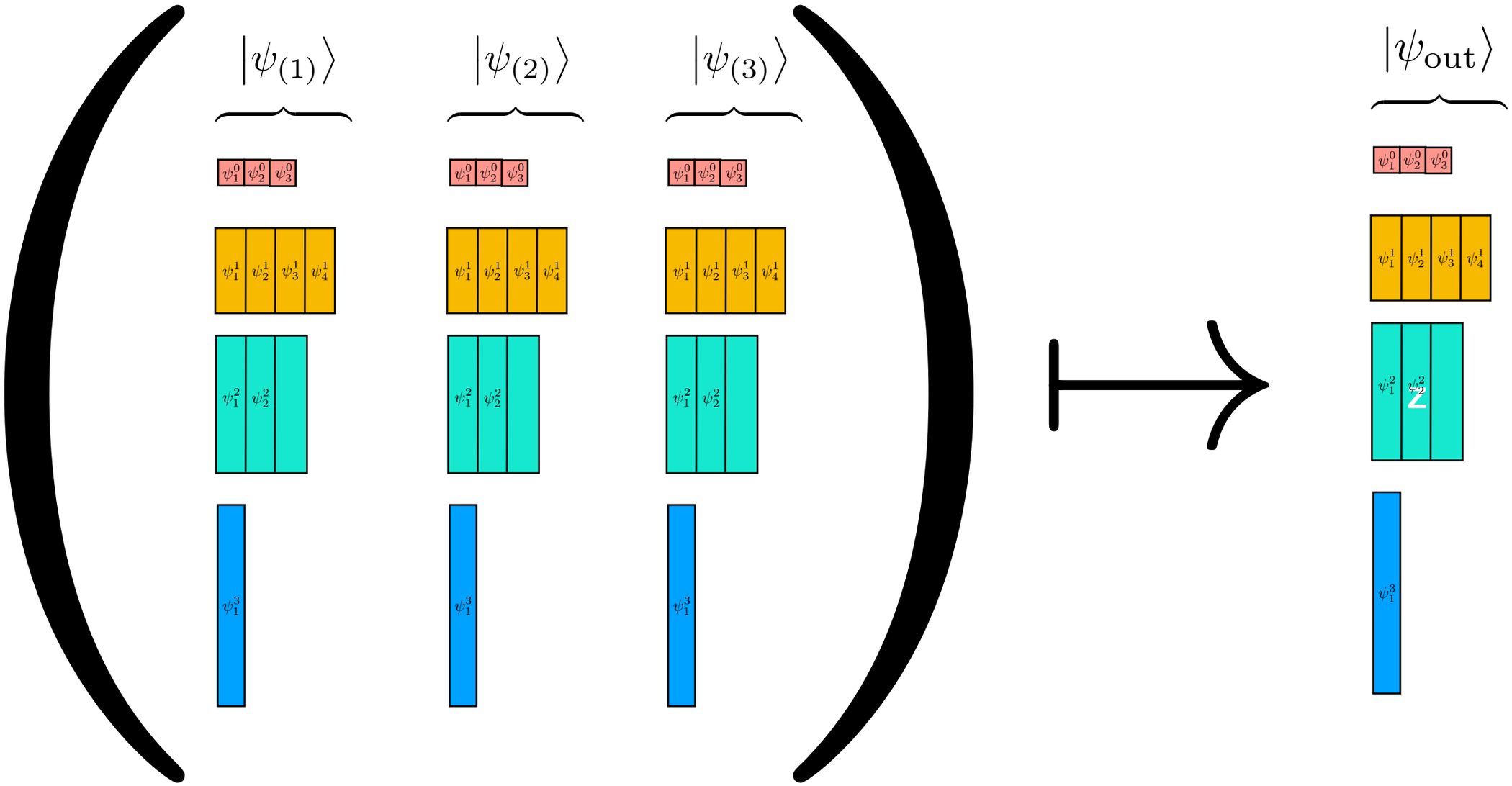
ψ_1^1 ψ_2^1 ψ_3^1 ψ_4^1

$\ell = 2$ ρ_2

ψ_1^2 ψ_2^2

$\ell = 3$ ρ_3

ψ_1^3


$|\psi_{(1)}\rangle$
$|\psi_{(2)}\rangle$
$|\psi_{(3)}\rangle$
$|\psi_{\text{out}}\rangle$
ψ_1^0 ψ_2^0 ψ_3^0
ψ_1^1 ψ_2^1 ψ_3^1 ψ_4^1
ψ_1^2 ψ_2^2
ψ_1^3
$\mathbb{Z}$

Fourier space nonlinearities

Pointwise nonlinearities:

$$h(u) = \mathrm{ReLU}(f(u)) \qquad\qquad h(u-t) = \mathrm{ReLU}(f(u-t))$$

In Fourier space:

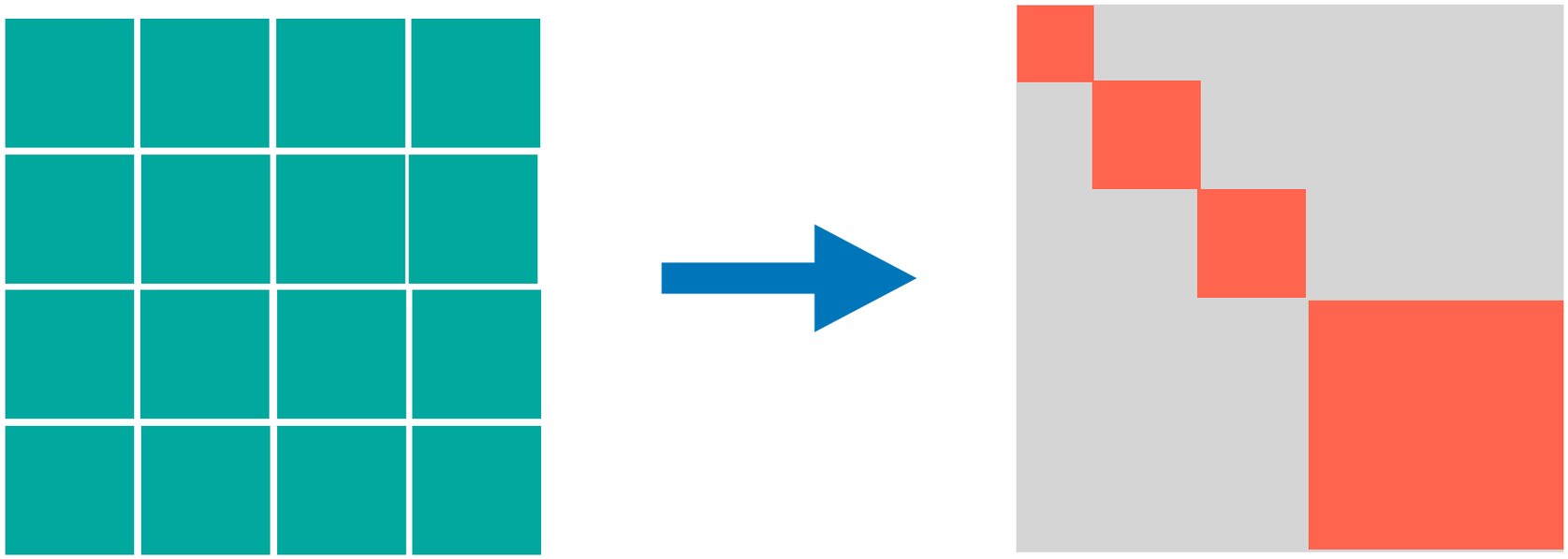
$$h(u) = (f(u))^2 \qquad\qquad \widehat{h}(k) = \int_{k'} \widehat{f}(k-k')\,\widehat{f}(k')\,dk'$$

The Clebsch-Gordan product

The tensor product of two irreducible representations of a compact group G decomposes into irreducibles in the form

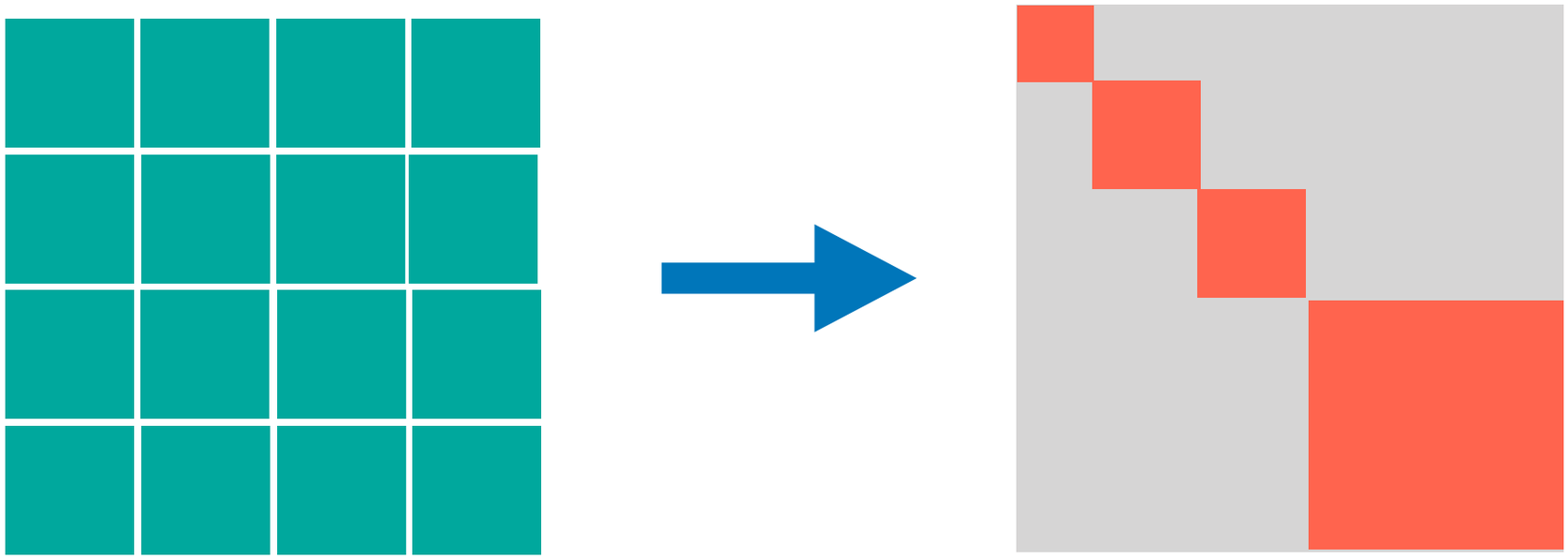
$$\rho_1(g) \otimes \rho_2(g) = \bigoplus_{\ell} \bigoplus_{m=1}^{\kappa(\ell)} C_m^\ell \cdot \rho_\ell(g) \cdot (C_m^\ell)^\dagger$$

Clebsch-Gordan nonlinearities



$$\rho_1(g) \otimes \rho_2(g) = C_{\rho_1,\rho_2} \left[\bigoplus_{\rho} \bigoplus_{1}^{\kappa(\rho)} \rho(g) \right] C^{\dagger}_{\rho_1,\rho_2}$$

Clebsch-Gordan nonlinearities



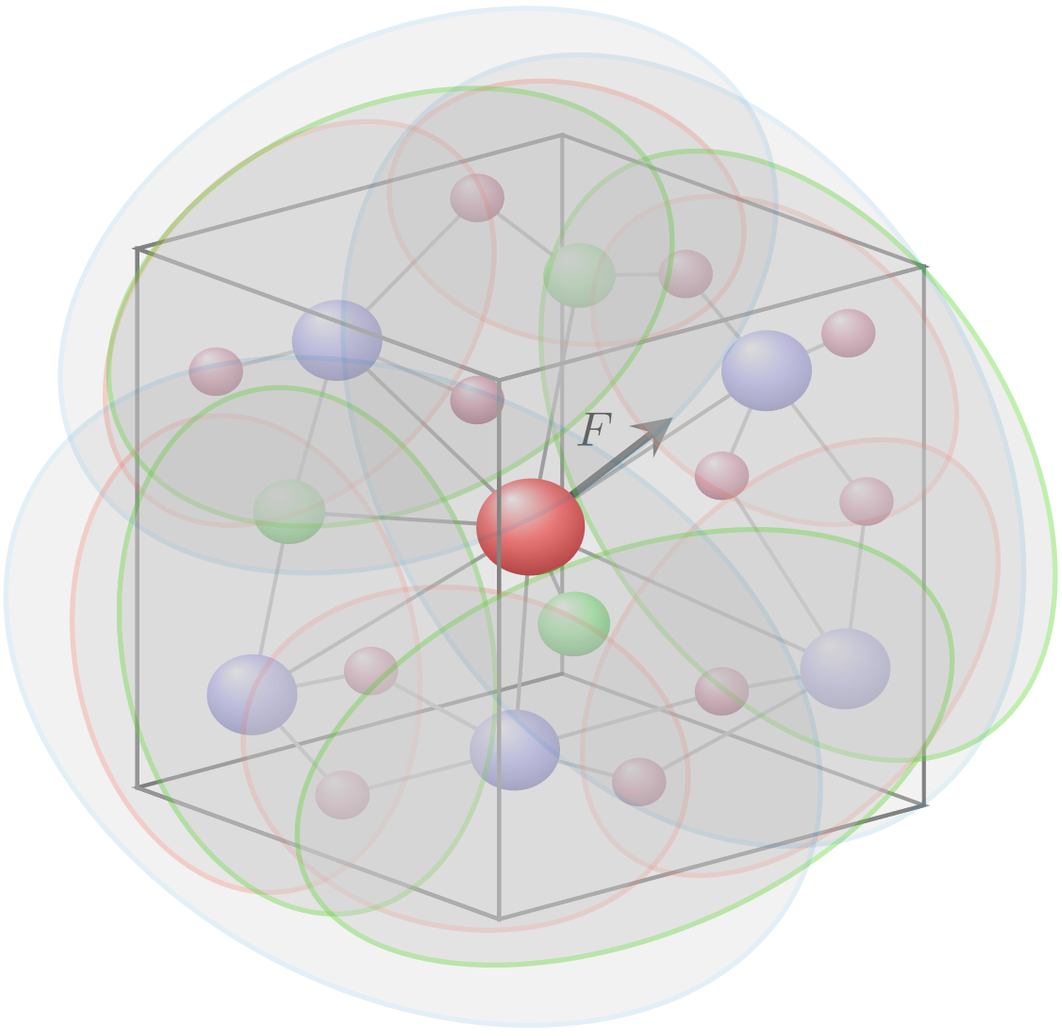
$$\widehat{f}_1(\rho_1) \otimes \widehat{f}_2(\rho_2) = C_{\rho_1,\rho_2} \left[\bigoplus_{\rho} \bigoplus_{1}^{\kappa(\rho)} \widehat{h}(\rho) \right] C^{\dagger}_{\rho_1,\rho_2}$$

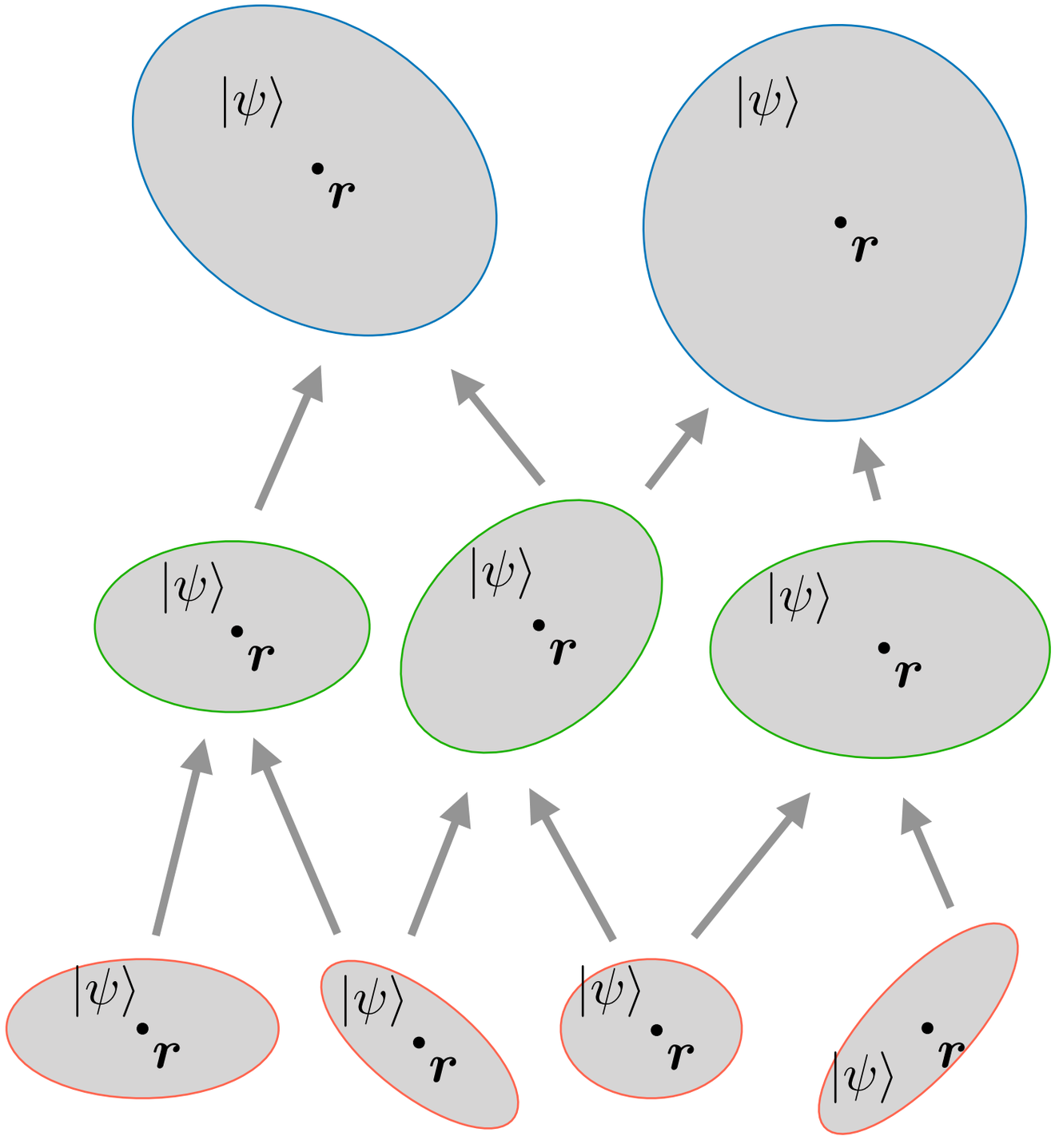
3. Cormorant networks for learning force fields

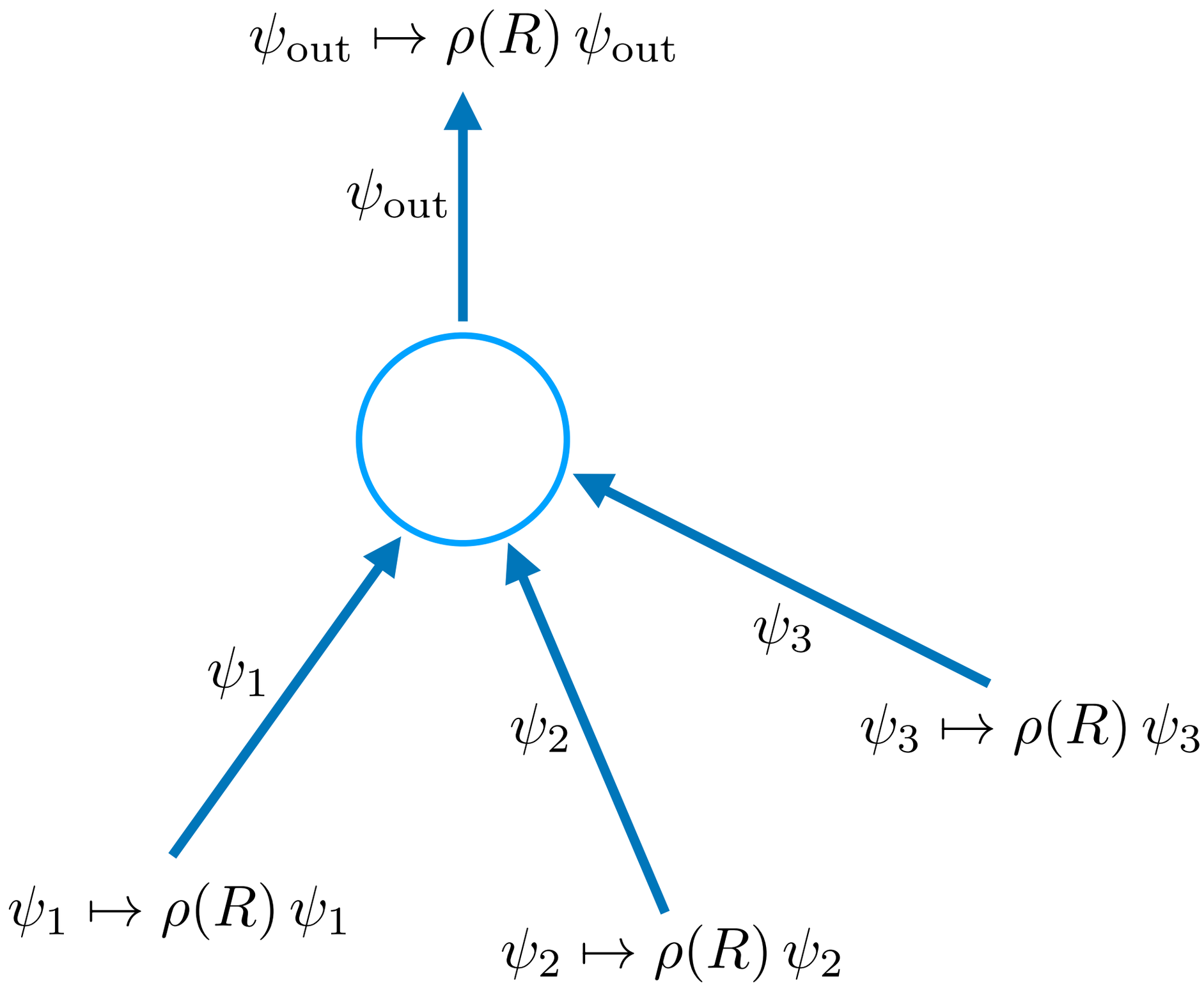
Brandon Anderson

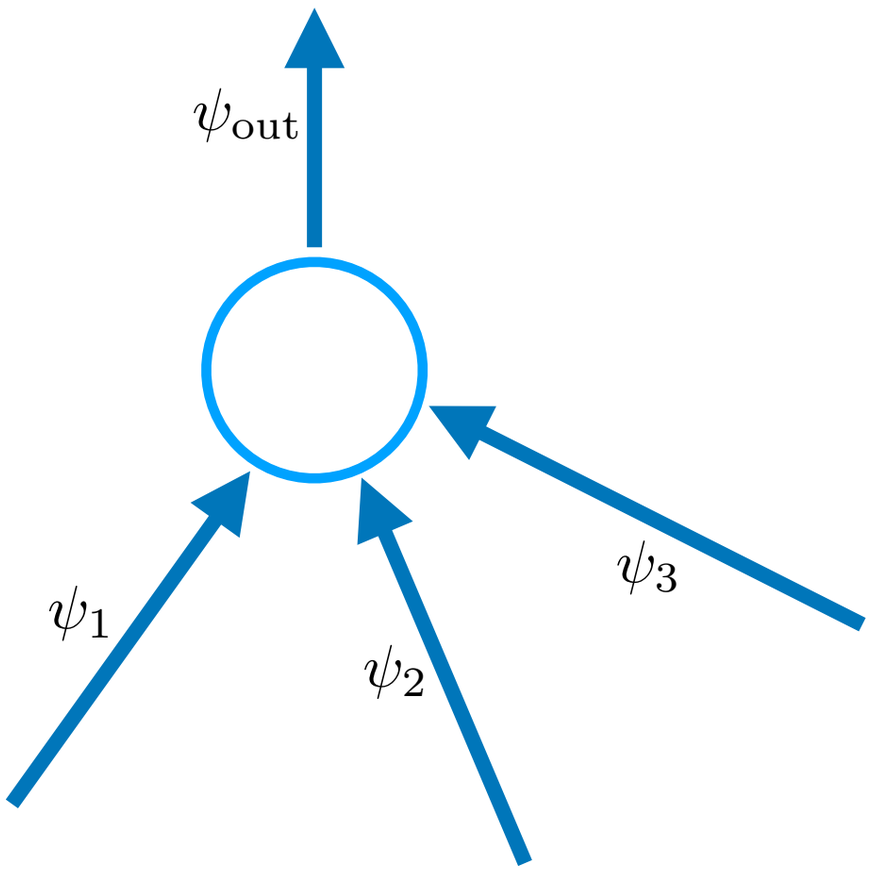
Hy Trong Son

Compositional structure






$\psi_{\text{out}} \mapsto \rho(R)\,\psi_{\text{out}}$
ψ_{out}
ψ_1
$\psi_1 \mapsto \rho(R)\,\psi_1$
ψ_2
$\psi_2 \mapsto \rho(R)\,\psi_2$
ψ_3
$\psi_3 \mapsto \rho(R)\,\psi_3$



$$\psi_{\text{out}} = \sigma\left(\sum_i \psi_i * \chi\right)$$

$$\widehat{\psi}^{\ell'}_{\text{out}} = \sigma\left(\sum_i \psi^{\ell}_i * \chi^{\ell}\right)$$

But what form should the nonlinearity take?

The Clebsch-Gordan product

The tensor product of two irreducible representations of a compact group G decomposes into irreducibles in the form

$$\rho_1(g) \otimes \rho_2(g) = \bigoplus_{\ell} \bigoplus_{m=1}^{\kappa(\ell)} C_m^\ell \cdot \rho_\ell(g) \cdot (C_m^\ell)^\dagger$$

Clebsch-Gordan gates

General form of aggregation:

$$|\psi\rangle = \sigma\Big(W \sum_i (\boldsymbol{r}_i - \boldsymbol{r})^{\otimes k} \otimes |\psi_i\rangle^{\otimes p} \Big)$$

The nonlinearity is a low order Clebsch-Gordan product

$$|\psi\rangle \mapsto \rho(g)\,|\psi\rangle$$

["Tensor Field Networks" by Thomas et al., 2018] ["Clebsch-Gordan networks", K., Trivedi & Zhen , 2018] ["3D steerable CNNs", Weiler et al., 2018]

Physical interactions

Monopole:

$$V_C = -\frac{1}{4\pi\epsilon_0}\,\frac{q_A q_B}{|\boldsymbol{r}_{AB}|}$$

Dipole:

$$V_{d/d} = \frac{1}{4\pi\epsilon_0}\left[\frac{\boldsymbol{\mu}_A\cdot\boldsymbol{\mu}_B}{|\boldsymbol{r}_{AB}|^3} - 3\,\frac{(\boldsymbol{\mu}_A\cdot\boldsymbol{r}_{AB})(\boldsymbol{\mu}_B\cdot\boldsymbol{r}_{AB})}{|\boldsymbol{r}_{AB}|^5}\right].$$

Quadropole:

$$V_{q/q} = \frac{3}{4}\frac{\vartheta_A\vartheta_B}{4\pi\epsilon_0\,|\boldsymbol{r}_{AB}|^5}\big[1 - 5\cos^{\theta}_A - 5\cos^2\theta_B - 15\cos^2\theta_A\cos^2\theta_B +$$
$$2(4\cos\theta_A\theta_B - \sin\theta_A\sin\theta_B\cos(\phi_A-\phi_B))^2\big]$$

[Thomas et al: Tensor Field Networks (2018)]

The Cormorant architecture

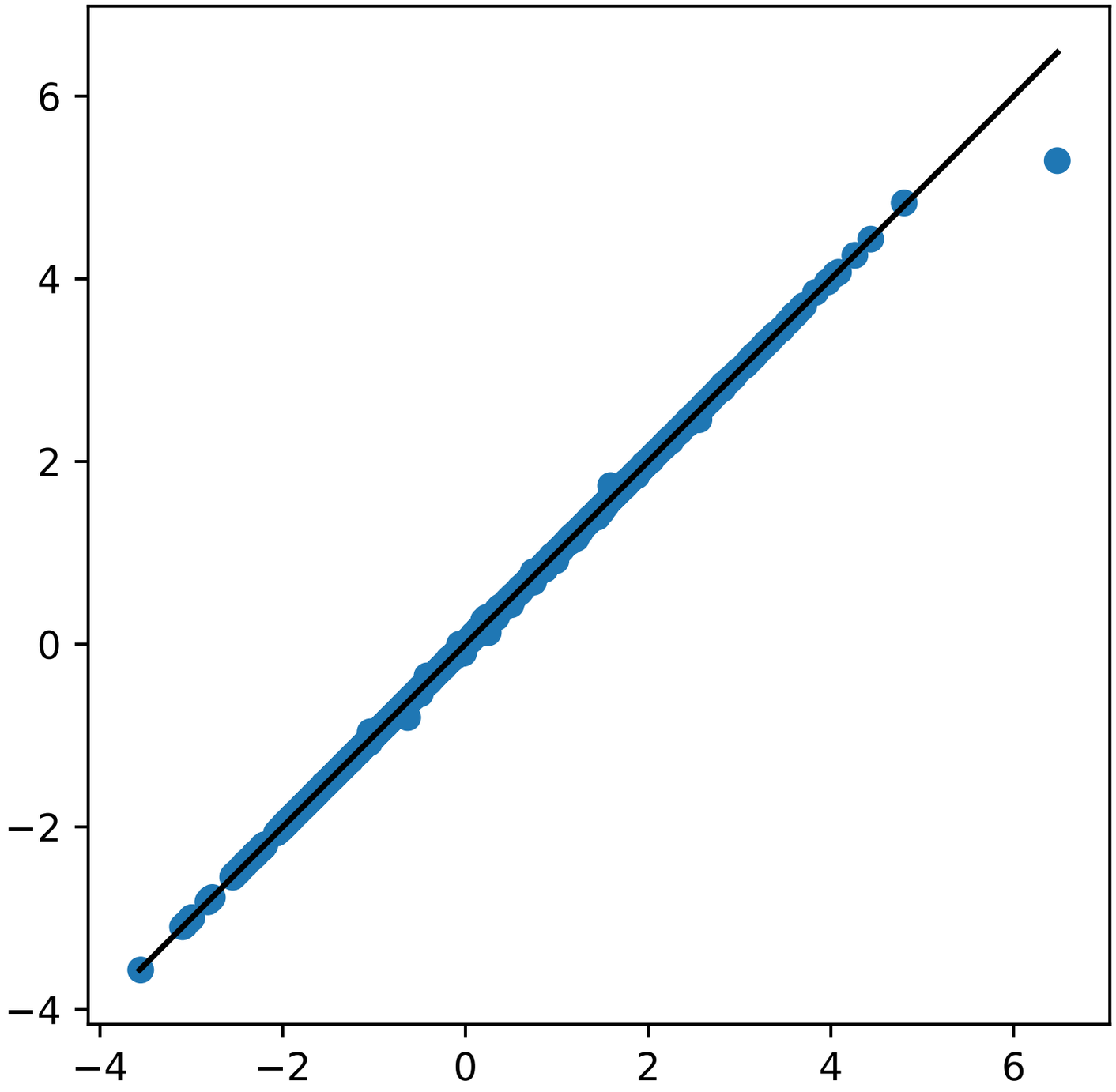
$$\mathrm{CGLayer}(\{F_i, \mathbf{r}_{ij}\}) = \left[F_i \oplus \left(F_i \otimes_{\mathrm{cg}} F_i\right) \oplus \left(\sum_j \left(\Upsilon^{(1)}_{ij} \oplus \left(F_i \cdot F_j\right)\Upsilon^{(3)}_{ij}\right) \otimes_{\mathrm{cg}} F_j\right)\right] \cdot W'$$

$$\Upsilon^{(n)}(\mathbf{r}_{jj'}) = \bigoplus_{\ell=0}^{\ell_{\max}} \mathcal{F}^{\ell}(r_{ij}) Y^{\ell}(\hat{\mathbf{r}}_{jj'}),$$

[Anderson, Son & K: Cormorant (2019)]

MD-17 results (kcal/mol)

	NBody50k	Deep95k	DTNN50k	SchE50k	SchEF50k	GDML1k	SchE1k	SchEF1k
Aspirin	* 0.103	0.201	--	0.250	0.120	0.270	4.200	0.370
Benzene	* 0.035	0.065	0.040	0.080	0.070	0.070	1.190	0.080
Ethanol	* 0.029	0.055	--	0.070	0.050	0.150	0.930	0.080
Malonaldehyde	* 0.056	0.092	0.190	0.130	0.080	0.160	2.030	0.130
Naphthalene	* 0.043	0.095	--	0.200	0.110	0.120	3.580	0.160
Salicylic_acid	* 0.072	0.106	0.410	0.250	0.100	0.120	3.270	0.200
Toluene	* 0.042	0.085	0.180	0.160	0.090	0.120	2.950	0.120
Uracil	* 0.045	0.085	--	0.130	0.100	0.110	2.260	0.140

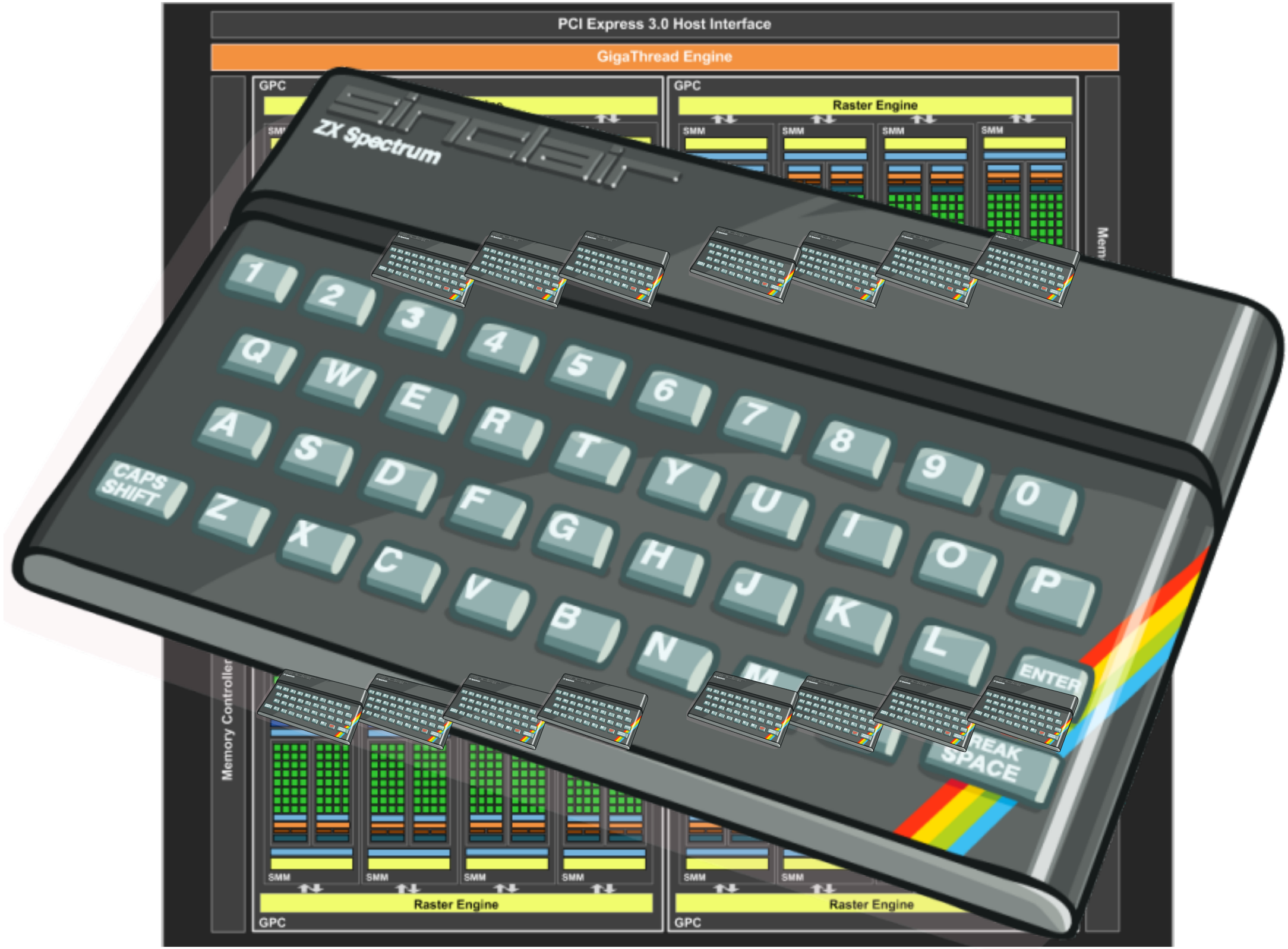

6
4
2
0
−2
−4
−4
−2
0
2
4
6

QM9 results

	us	SchNett	google1	google2	googleML
alpha (bohr3)	* 0.100	0.235	0.161	0.232	0.175
Cv (cal/molK)	* 0.031	0.033	0.084	0.097	0.044
gap (eV)	* 0.061	0.063	0.088	0.087	0.107
homo (eV)	* 0.036	0.041	0.057	0.055	0.066
lumo (eV)	* 0.032	0.034	0.063	0.062	0.084
mu (D)	0.046	* 0.033	0.247	0.101	0.334
omega1 (cm-1)	3.229	--	6.220	4.760	* 2.710
U0 (eV)	0.028	* 0.014	0.042	0.150	0.025
zpve (meV)	0.004	1.700	0.004	0.010	* 0.002

4. Implementation


PCI Express 3.0 Host Interface
GigaThread Engine
Memory Controller
GPC
Raster Engine
SMM
L2 Cache
Memory Controller
GPC
Raster Engine
SMM
Memory Controller
Memory Controller
GPC
Raster Engine
SMM
GPC
Raster Engine
SMM

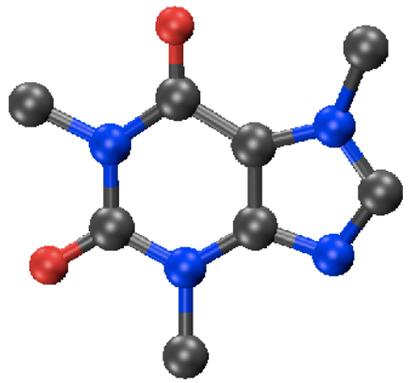

PCI Express 3.0 Host Interface
GigaThread Engine
GPC
Raster Engine
SMM
Memory Controller
ZX Spectrum
CAPS SHIFT
ENTER
SPACE

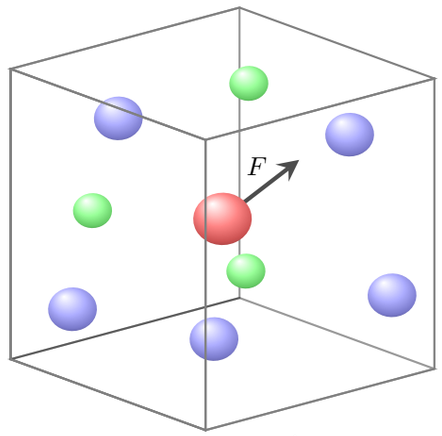
GraphFlow

Custom C++ based deep learning library

PyTorch N-body

General purpose CG operations, extensive functionality





PyTorch CCN

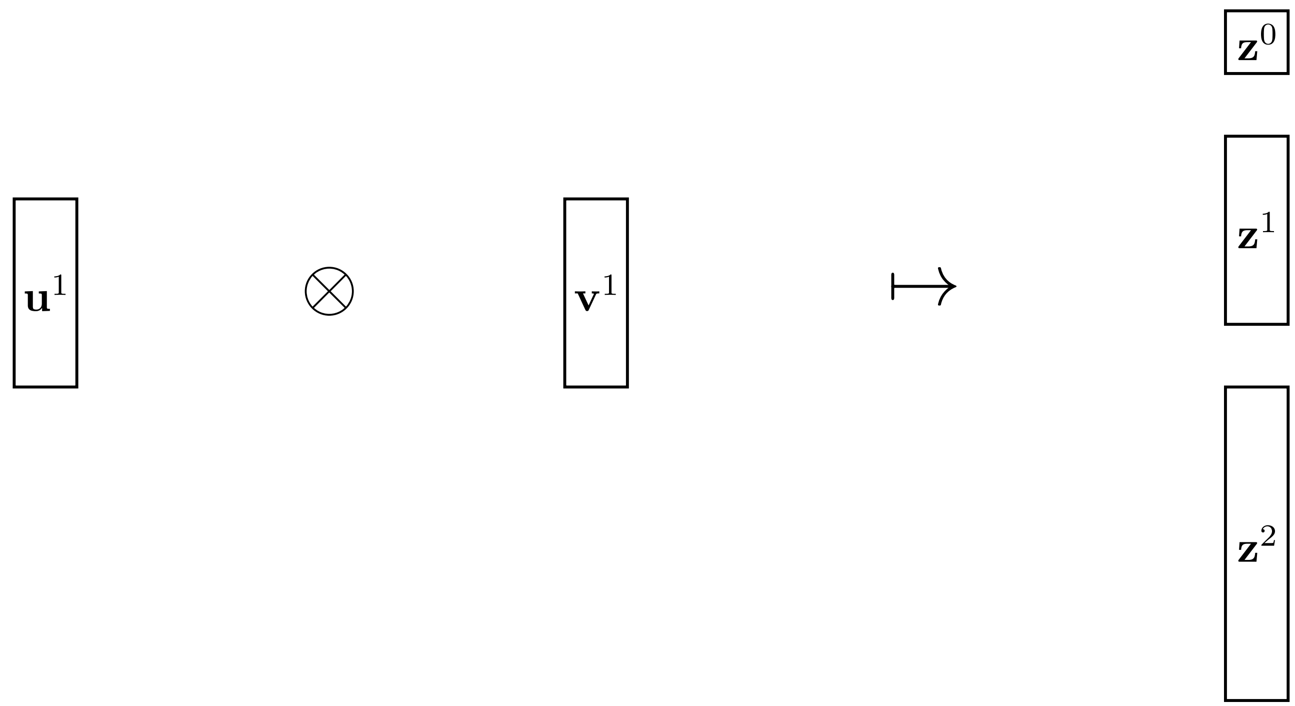
Simple Pythonic implementation

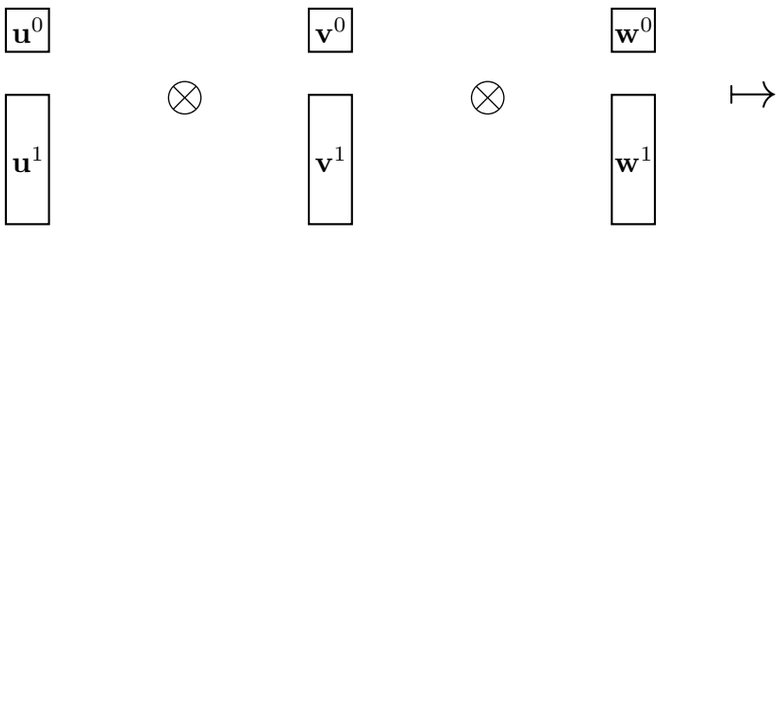
GCCN

Production level code with GUI and GPU packs

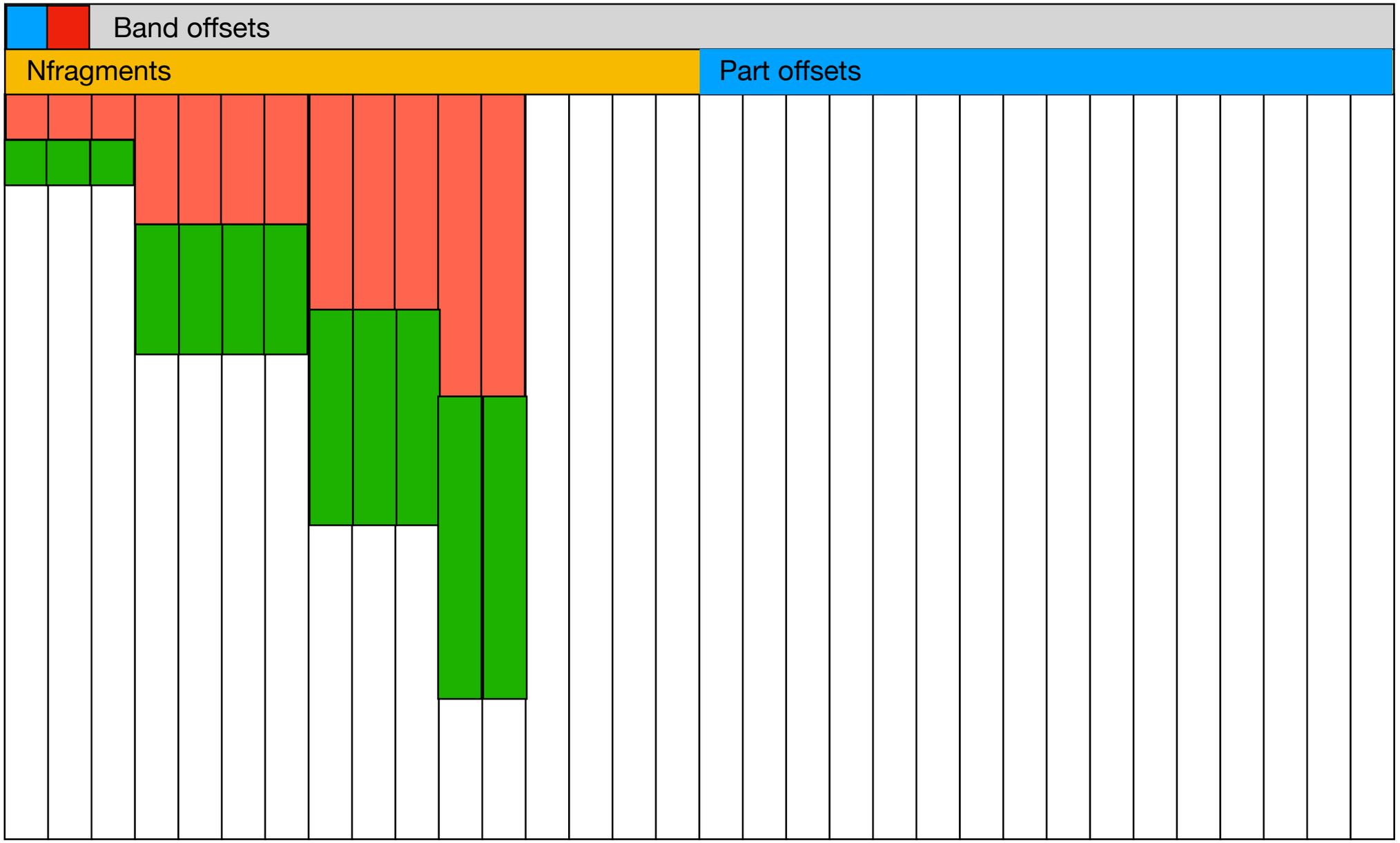
FastCG

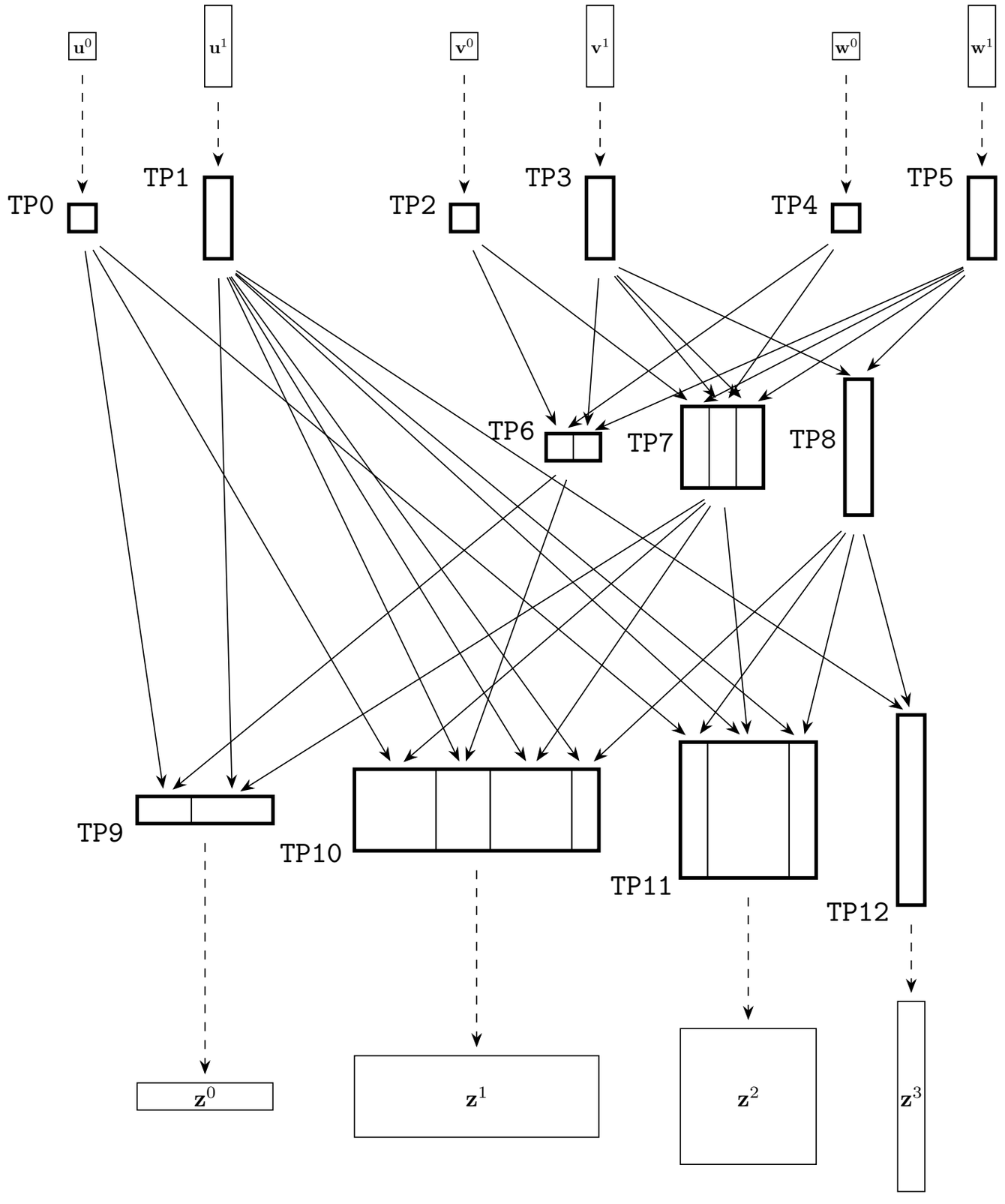
Highly optimized GPU-bound CG library


$\mathbf{u}^1$
$\otimes$
$\mathbf{v}^1$
$\mapsto$
$\mathbf{z}^0$
$\mathbf{z}^1$
$\mathbf{z}^2$

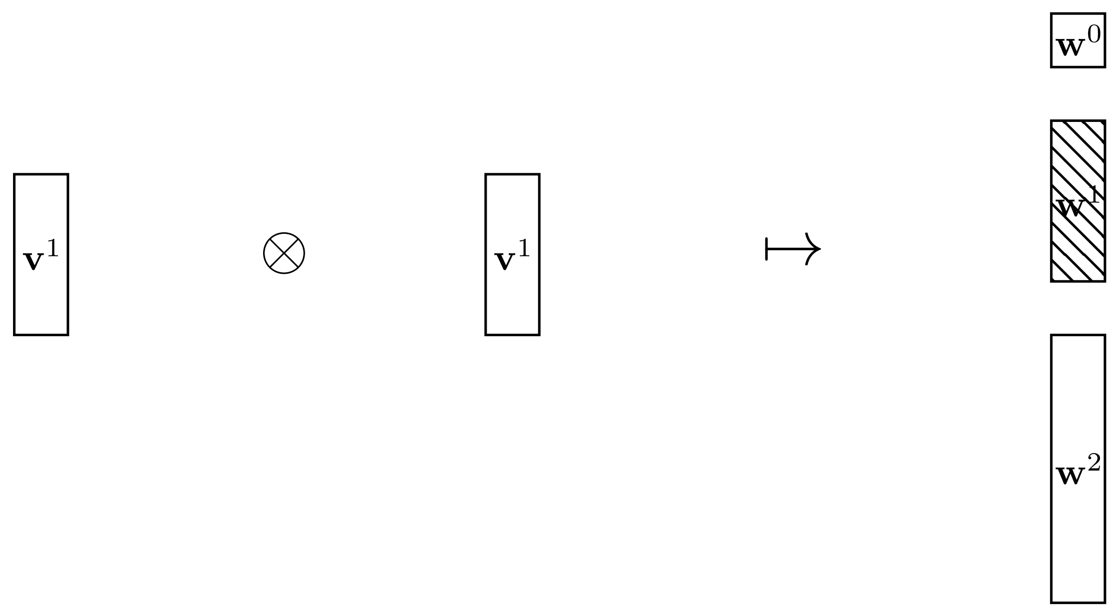

$\mathbf{u}^0$
$\mathbf{u}^1$
$\otimes$
$\mathbf{v}^0$
$\mathbf{v}^1$
$\otimes$
$\mathbf{w}^0$
$\mathbf{w}^1$
$\mapsto$
$\mathbf{u}^0 \otimes \mathbf{v}^0 \otimes \mathbf{w}^0$
$\mathbf{u}^0 \otimes \mathbf{v}^1 \otimes \mathbf{w}^1$
$\mathbf{u}^1 \otimes \mathbf{v}^0 \otimes \mathbf{w}^1$
$\mathbf{u}^1 \otimes \mathbf{v}^1 \otimes \mathbf{w}^0$
$\mathbf{u}^1 \otimes \mathbf{v}^1 \otimes \mathbf{w}^1$
$\mathbf{u}^0 \otimes \mathbf{v}^0 \otimes \mathbf{w}^1$
$\mathbf{u}^0 \otimes \mathbf{v}^1 \otimes \mathbf{w}^0$
$\mathbf{u}^1 \otimes \mathbf{v}^0 \otimes \mathbf{w}^0$
$\mathbf{u}^0 \otimes \mathbf{v}^1 \otimes \mathbf{w}^1$
$\mathbf{u}^1 \otimes \mathbf{v}^0 \otimes \mathbf{w}^1$
$\mathbf{u}^1 \otimes \mathbf{v}^1 \otimes \mathbf{w}^0$
$\mathbf{u}^1 \otimes \mathbf{v}^1 \otimes \mathbf{w}^1$
$\mathbf{u}^1 \otimes \mathbf{v}^1 \otimes \mathbf{w}^1$

GPU friendly storage

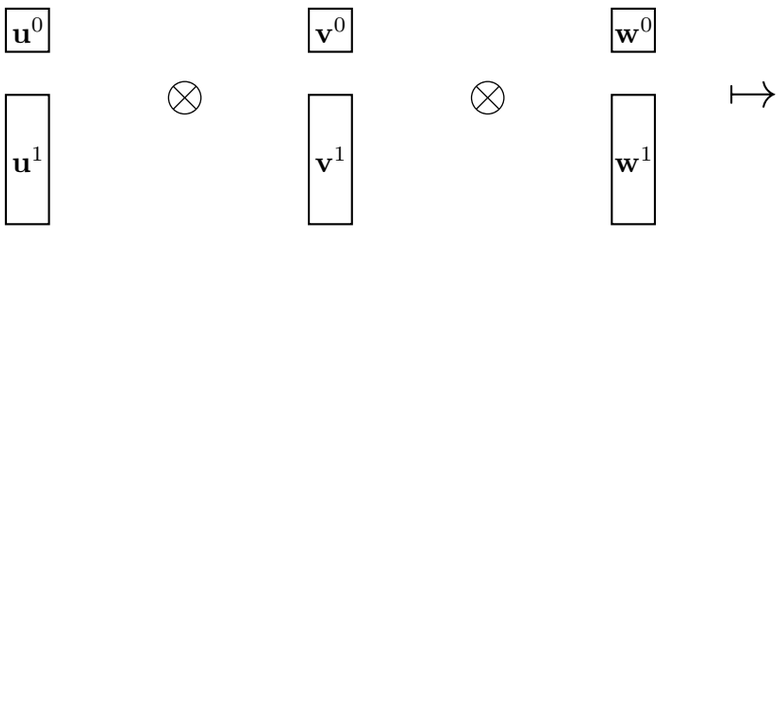


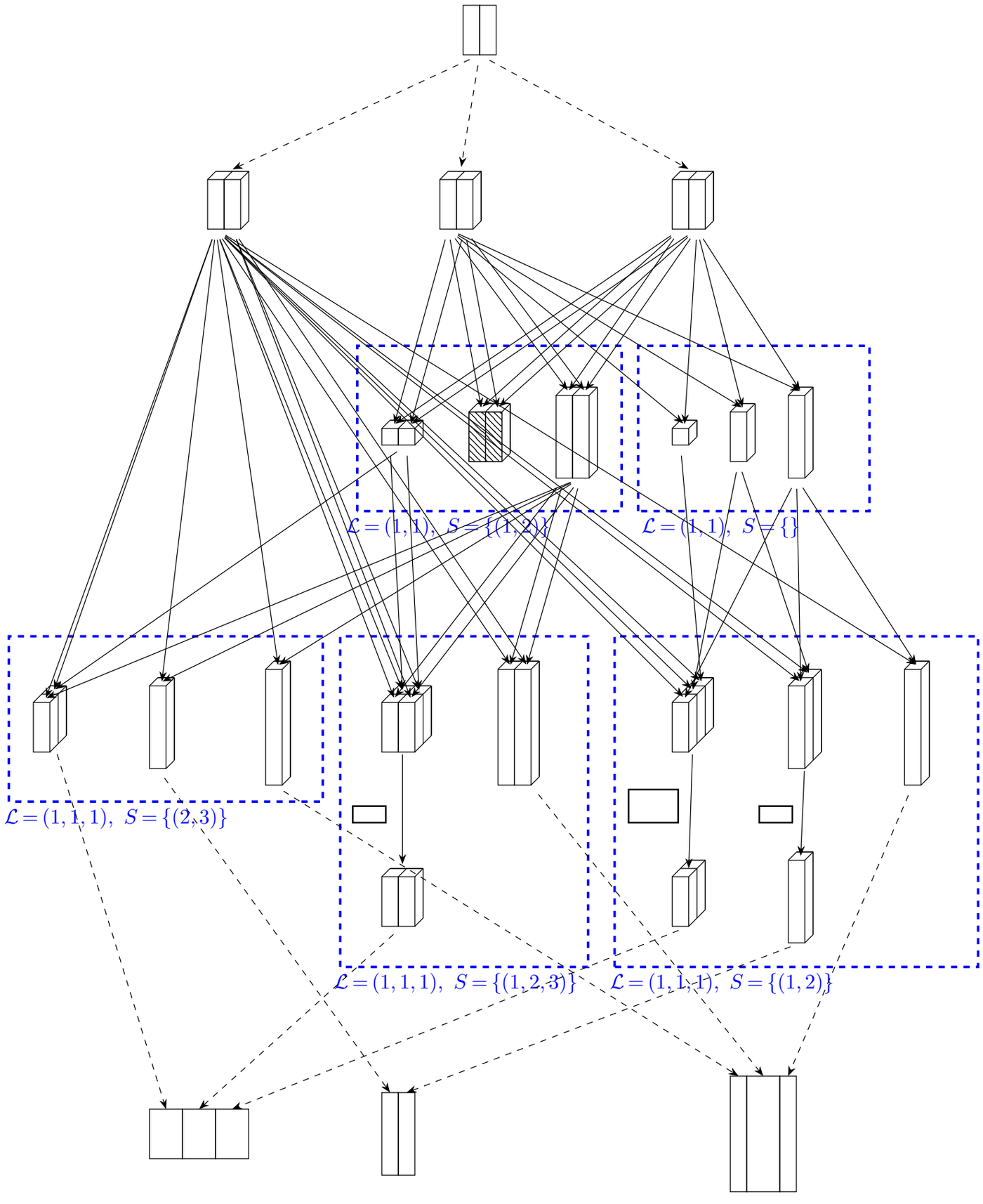

$\mathbf{u}^0$
$\mathbf{u}^1$
$\mathbf{v}^0$
$\mathbf{v}^1$
$\mathbf{w}^0$
$\mathbf{w}^1$
TP0
TP1
TP2
TP3
TP4
TP5
TP6
TP7
TP8
TP9
TP10
TP11
TP12
$\mathbf{z}^0$
$\mathbf{z}^1$
$\mathbf{z}^2$
$\mathbf{z}^3$

```
1   TPprogram  CGproduct(){
2     TPpart0 (l=0)[0m (n=1){
3       input(0,0);
4     }
5     TPpart1 (l=1) (n=1){
6       input(0,1);
7     }
8     TPpart2 (l=0) (n=1){
9       input(1,0);
10    }
11    TPpart3 (l=1) (n=1){
12      input(1,1);
13    }
14    TPpart4 (l=0) (n=1){
15      input(2,0);
16    }
17    TPpart5 (l=1) (n=1){
18      input(2,1);
19    }
20    TPpart6 (l=0) (n=2){
21      CG(2,4)[0];
22      CG(3,5)[1];
23    }
24    TPpart7 (l=1) (n=3){
25      CG(2,5)[0];
26      CG(3,4)[1];
27      CG(3,5)[2];
28    }
29    TPpart8 (l=2) (n=1){
30      CG(3,5)[0];
31    }
32    TPpart9 (l=0) (n=5){
33      output(0);
34      CG(0,6)[0];
35      CG(1,7)[2];
36    }
37    TPpart10 (l=1) (n=9){
38      output(1);
39      CG(0,7)[0];
40      CG(1,6)[3];
41      CG(1,7)[5];
```



$\mathbf{v}^1$
$\otimes$
$\mathbf{v}^1$
$\mapsto$
$\mathbf{w}^0$
$\mathbf{w}^1$
$\mathbf{w}^2$

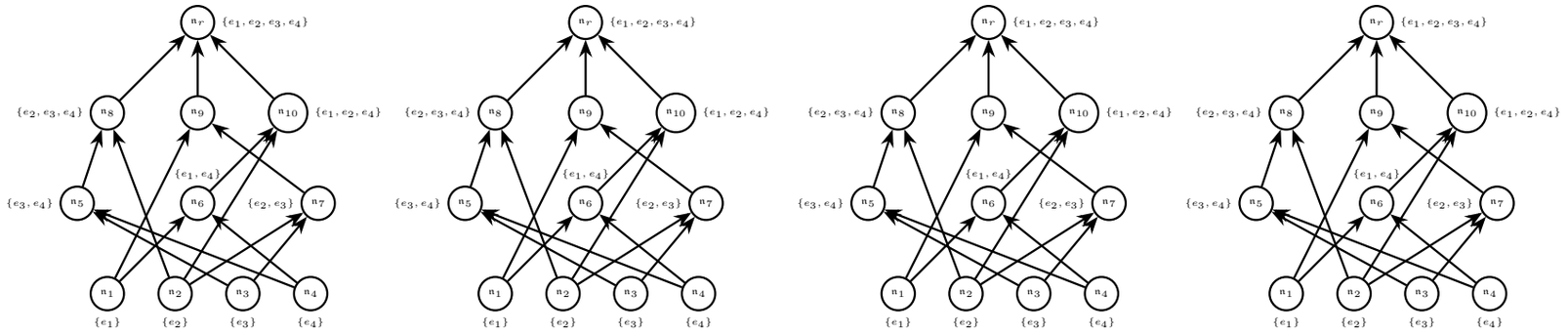

$\mathbf{u}^0$
$\mathbf{u}^1$
$\otimes$
$\mathbf{v}^0$
$\mathbf{v}^1$
$\otimes$
$\mathbf{w}^0$
$\mathbf{w}^1$
$\mapsto$
$\mathbf{u}^0 \otimes \mathbf{v}^0 \otimes \mathbf{w}^0$
$\mathbf{u}^0 \otimes \mathbf{v}^1 \otimes \mathbf{w}^1$
$\mathbf{u}^1 \otimes \mathbf{v}^0 \otimes \mathbf{w}^1$
$\mathbf{u}^1 \otimes \mathbf{v}^1 \otimes \mathbf{w}^0$
$\mathbf{u}^1 \otimes \mathbf{v}^1 \otimes \mathbf{w}^1$
$\mathbf{u}^0 \otimes \mathbf{v}^0 \otimes \mathbf{w}^1$
$\mathbf{u}^0 \otimes \mathbf{v}^1 \otimes \mathbf{w}^0$
$\mathbf{u}^1 \otimes \mathbf{v}^0 \otimes \mathbf{w}^0$
$\mathbf{u}^0 \otimes \mathbf{v}^1 \otimes \mathbf{w}^1$
$\mathbf{u}^1 \otimes \mathbf{v}^0 \otimes \mathbf{w}^1$
$\mathbf{u}^1 \otimes \mathbf{v}^1 \otimes \mathbf{w}^0$
$\mathbf{u}^1 \otimes \mathbf{v}^1 \otimes \mathbf{w}^1$
$\mathbf{u}^1 \otimes \mathbf{v}^1 \otimes \mathbf{w}^1$


$\mathcal{L} = (1,1),\ S = \{(1,2)\}$
$\mathcal{L} = (1,1),\ S = \{\}$
$\mathcal{L} = (1,1,1),\ S = \{(2,3)\}$
$\mathcal{L} = (1,1,1),\ S = \{(1,2,3)\}$
$\mathcal{L} = (1,1,1),\ S = \{(1,2)\}$

```
SPprogram  TensorPower3(){
  Snode0 () () (l=0) (0x1){
    input(0);
  }
  Snode1 () () (l=1) (2x1){
    input(1);
  }
  Snode2 (1,1) () (l=2) (1x1) weave(0){
    (1,1)[0];
  }
  Snode3 (1,1) () (l=1) (1x1) weave(0){
    (1,1)[0];
  }
  Snode4 (1,1) () (l=0) (1x1) weave(0){
    (1,1)[0];
  }
  Snode5 (1,1) ((0,1)) (l=2) (2x1) weave(1){
    (1,1)[0];
  }
  Snode6 (1,1) ((0,1)) (l=0) (2x1) weave(1){
    (1,1)[0];
  }
  Snode7 (1,1,1) ((1,2)) (l=3) (1x1) weave(2){
    (1,5)[0];
  }
  Snode8 (1,1,1) ((1,2)) (l=2) (1x1) weave(2){
    (1,5)[0];
  }
  Snode9 (1,1,1) ((1,2)) (l=1) (1x2) weave(2){
    (1,5)[0];
    (1,6)[1];
  }
```


PCI Express 3.0 Host Interface
GigaThread Engine
Memory Controller
GPC
Raster Engine
SMM
L2 Cache
Memory Controller
GPC
Raster Engine
SMM
Memory Controller
Memory Controller
GPC
Raster Engine
SMM
GPC
Raster Engine
SMM

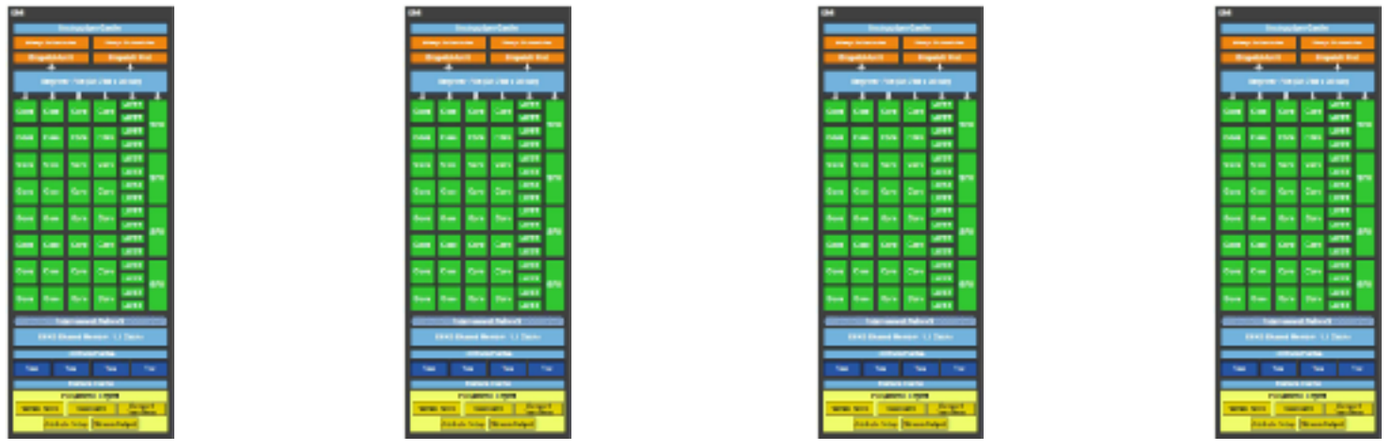
Computation graph



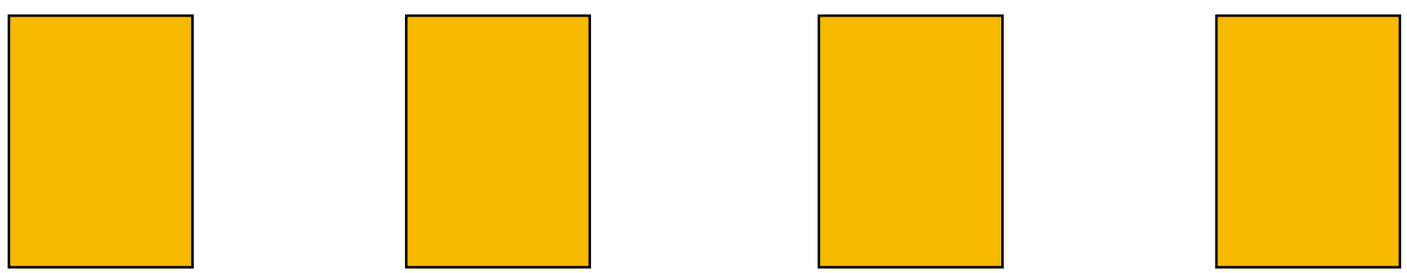
Bytecode

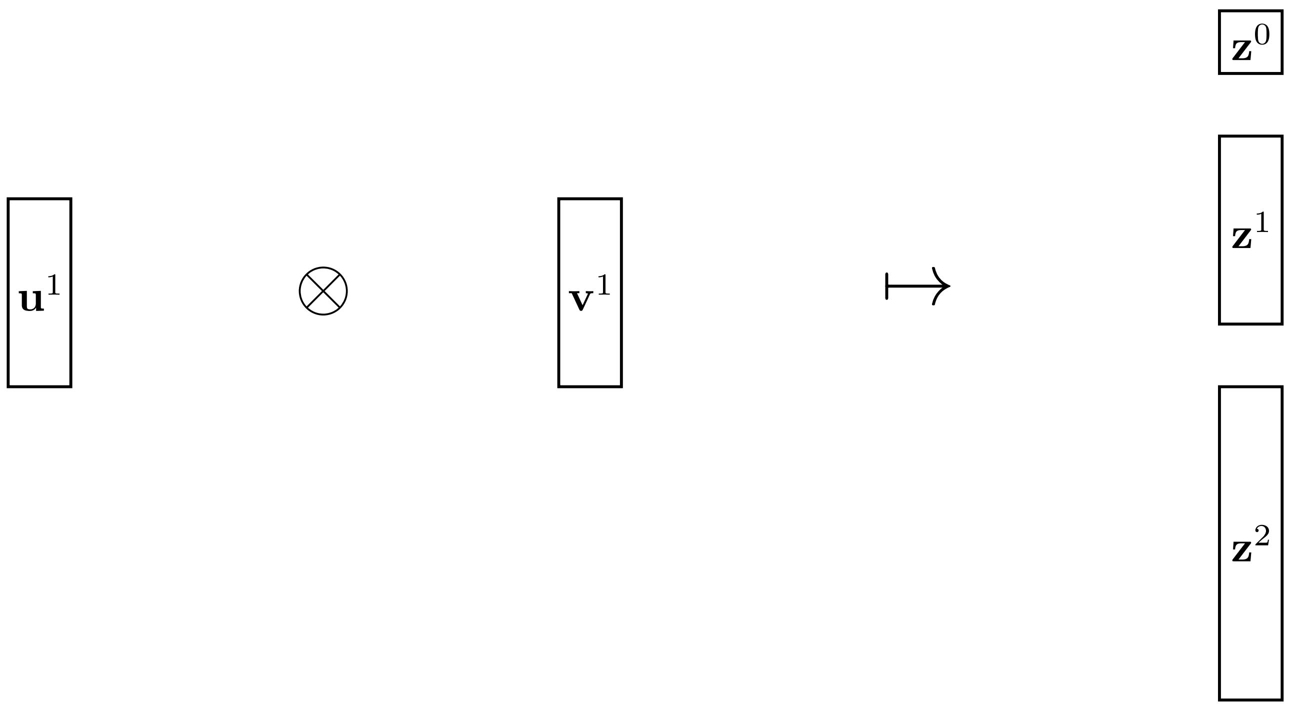
```
TPprogram  CGproduct(){
  TPpart0 (l=0)[0m (n=1){
    input(0,0);
  }
  TPpart1 (l=1) (n=1){
    input(0,1);
  }
  TPpart2 (l=0) (n=1){
    input(1,0);
  }
  TPpart3 (l=1) (n=1){
    input(1,1);
  }
  TPpart4 (l=0) (n=1){
    input(2,0);
  }
  TPpart5 (l=1) (n=1){
    input(2,1);
  }
  TPpart6 (l=0) (n=2){
    CG(2,4)[0];
    CG(3,5)[1];
  }
  TPpart7 (l=1) (n=3){
    CG(2,5)[0];
    CG(3,4)[1];
    CG(3,5)[2];
```

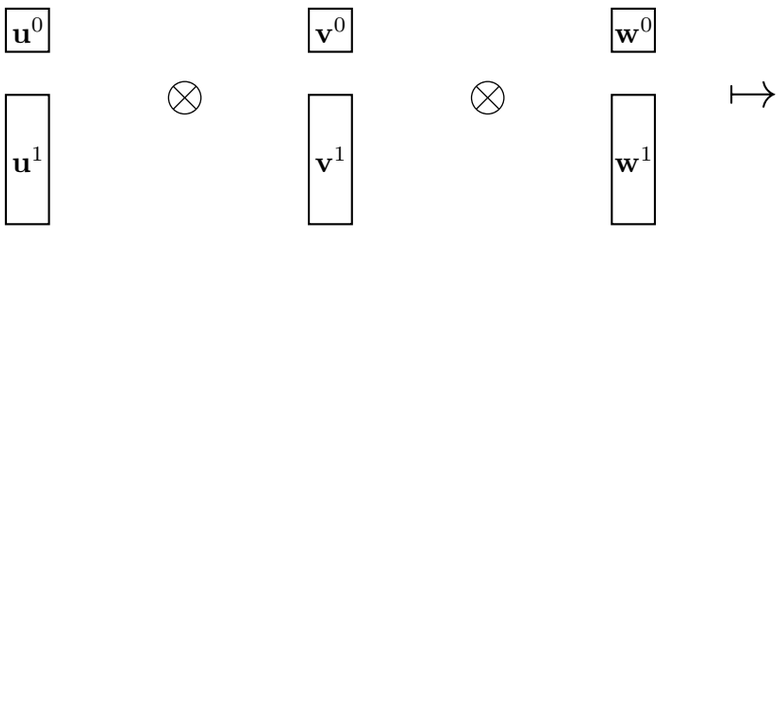
Same bytecode interpreter on each streaming mulitprocessor

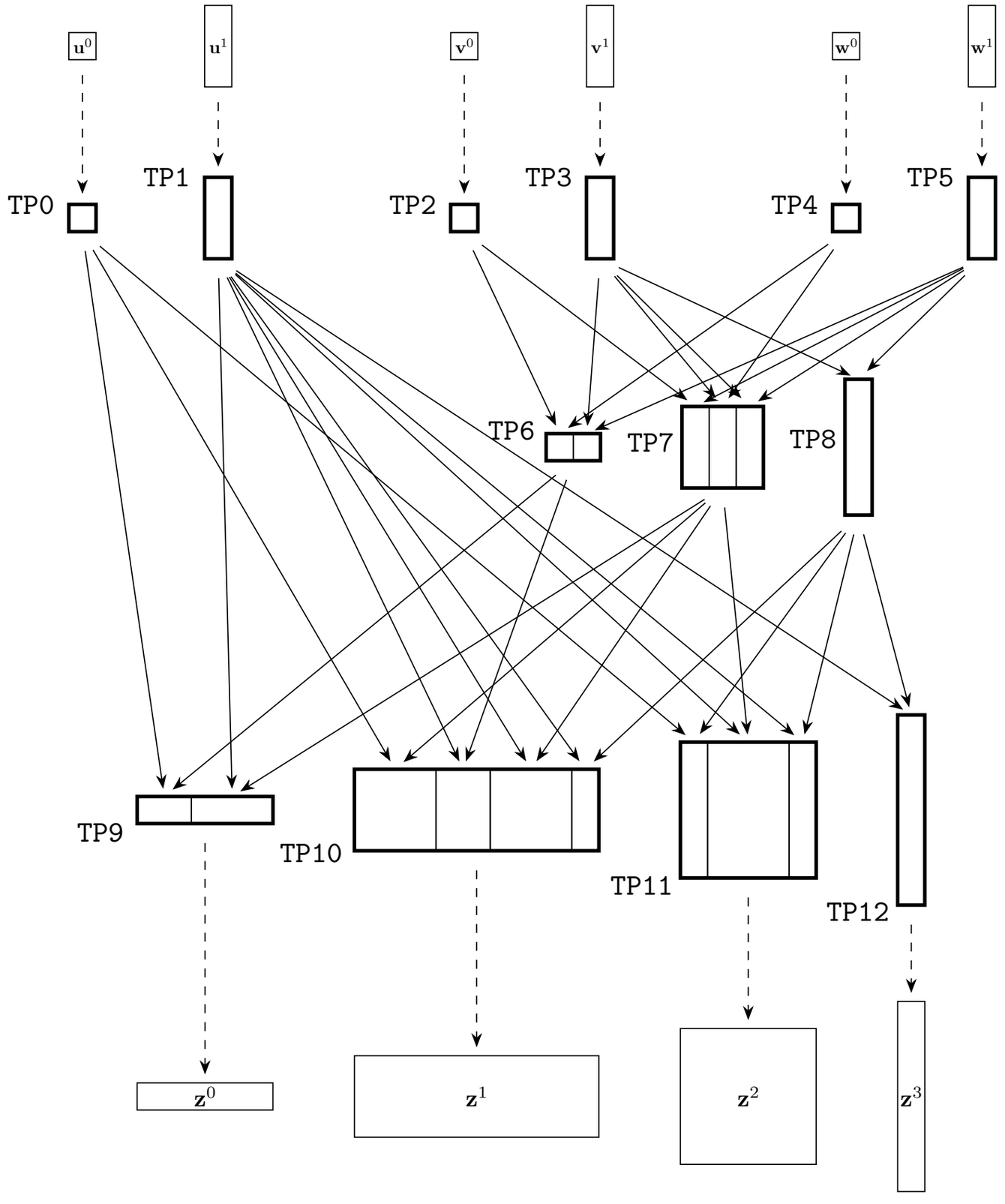


Result




$\mathbf{u}^1$
$\otimes$
$\mathbf{v}^1$
$\mapsto$
$\mathbf{z}^0$
$\mathbf{z}^1$
$\mathbf{z}^2$


$\mathbf{u}^0$
$\mathbf{u}^1$
$\otimes$
$\mathbf{v}^0$
$\mathbf{v}^1$
$\otimes$
$\mathbf{w}^0$
$\mathbf{w}^1$
$\mapsto$
$\mathbf{u}^0 \otimes \mathbf{v}^0 \otimes \mathbf{w}^0$
$\mathbf{u}^0 \otimes \mathbf{v}^1 \otimes \mathbf{w}^1$
$\mathbf{u}^1 \otimes \mathbf{v}^0 \otimes \mathbf{w}^1$
$\mathbf{u}^1 \otimes \mathbf{v}^1 \otimes \mathbf{w}^0$
$\mathbf{u}^1 \otimes \mathbf{v}^1 \otimes \mathbf{w}^1$
$\mathbf{u}^0 \otimes \mathbf{v}^0 \otimes \mathbf{w}^1$
$\mathbf{u}^0 \otimes \mathbf{v}^1 \otimes \mathbf{w}^0$
$\mathbf{u}^1 \otimes \mathbf{v}^0 \otimes \mathbf{w}^0$
$\mathbf{u}^0 \otimes \mathbf{v}^1 \otimes \mathbf{w}^1$
$\mathbf{u}^1 \otimes \mathbf{v}^0 \otimes \mathbf{w}^1$
$\mathbf{u}^1 \otimes \mathbf{v}^1 \otimes \mathbf{w}^0$
$\mathbf{u}^1 \otimes \mathbf{v}^1 \otimes \mathbf{w}^1$
$\mathbf{u}^1 \otimes \mathbf{v}^1 \otimes \mathbf{w}^1$


$\mathbf{u}^0$
$\mathbf{u}^1$
$\mathbf{v}^0$
$\mathbf{v}^1$
$\mathbf{w}^0$
$\mathbf{w}^1$
TP0
TP1
TP2
TP3
TP4
TP5
TP6
TP7
TP8
TP9
TP10
TP11
TP12
$\mathbf{z}^0$
$\mathbf{z}^1$
$\mathbf{z}^2$
$\mathbf{z}^3$

```
1   TPprogram  CGproduct(){
2     TPpart0 (l=0)[0m (n=1){
3       input(0,0);
4     }
5     TPpart1 (l=1) (n=1){
6       input(0,1);
7     }
8     TPpart2 (l=0) (n=1){
9       input(1,0);
10    }
11    TPpart3 (l=1) (n=1){
12      input(1,1);
13    }
14    TPpart4 (l=0) (n=1){
15      input(2,0);
16    }
17    TPpart5 (l=1) (n=1){
18      input(2,1);
19    }
20    TPpart6 (l=0) (n=2){
21      CG(2,4)[0];
22      CG(3,5)[1];
23    }
24    TPpart7 (l=1) (n=3){
25      CG(2,5)[0];
26      CG(3,4)[1];
27      CG(3,5)[2];
28    }
29    TPpart8 (l=2) (n=1){
30      CG(3,5)[0];
31    }
32    TPpart9 (l=0) (n=5){
33      output(0);
34      CG(0,6)[0];
35      CG(1,7)[2];
36    }
37    TPpart10 (l=1) (n=9){
38      output(1);
39      CG(0,7)[0];
40      CG(1,6)[3];
41      CG(1,7)[5];
```

1. Compositional structure

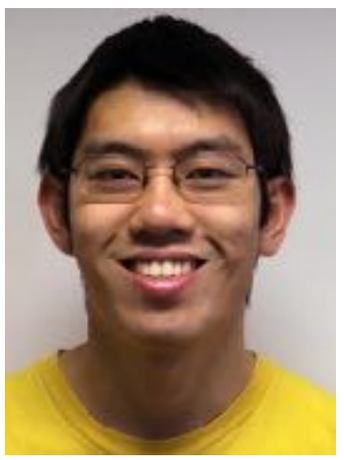
2. Covariance

   → Fourier space activations

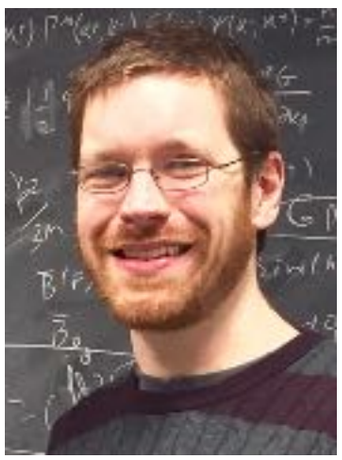
   Clebsch-Gordan nonlinearities

Shubhendu Trivedi

Hy Trong Son



Horace Pan



Brandon Anderson