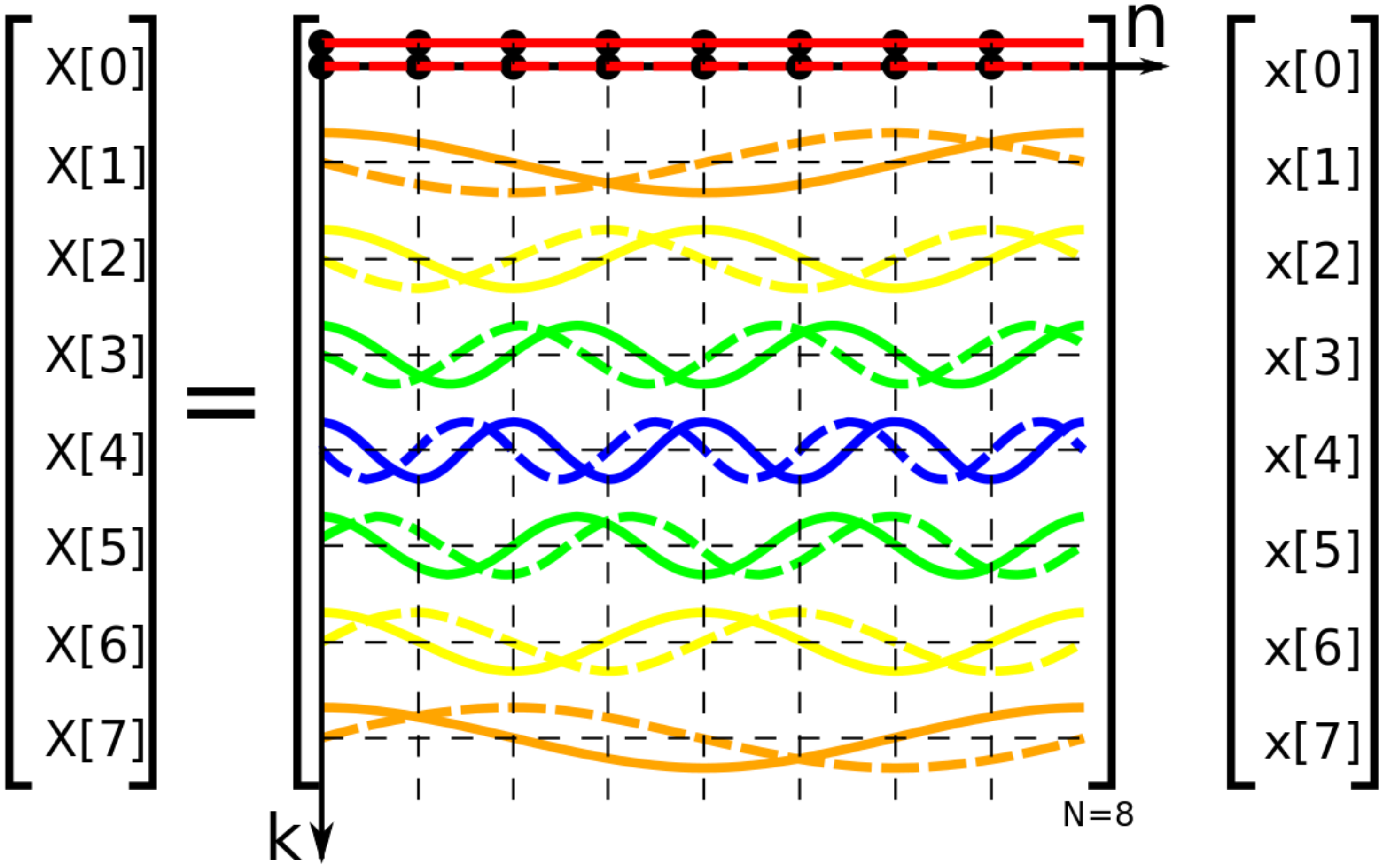
Revisiting convolutions



Definition

Definition *The convolution of two functions f and g is denoted by $*$ as the integral of the product of the two functions after one is reversed and shifted*

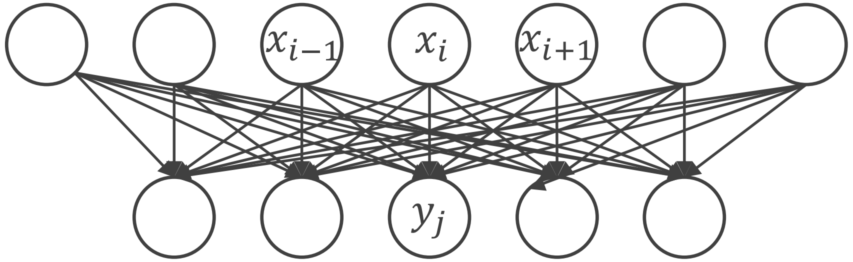
$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(t-\tau)\, d\tau = \int_{-\infty}^{\infty} f(t-\tau) g(\tau)\, d\tau$$

- For images $a_{rc} = \boldsymbol{x} * \boldsymbol{w} = \sum_{i=-a}^{a} \sum_{j=-b}^{b} x_{r-i,c-j} \cdot w_{ij}$

- To generalize to graphs, we must understand convolutions to their core
- Let's check some properties of what makes convolution a convolution

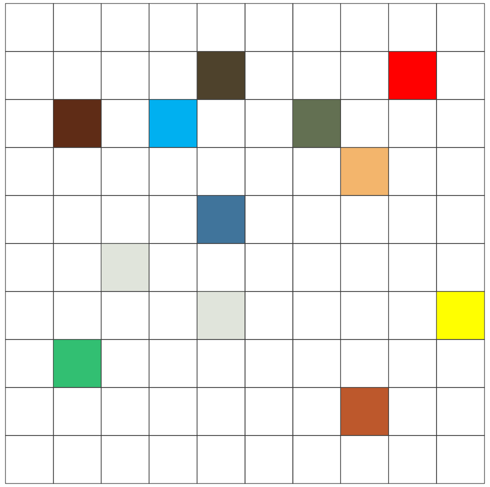
Convolutions matrices

- Fully connected ⇔ Full matrix multiplication

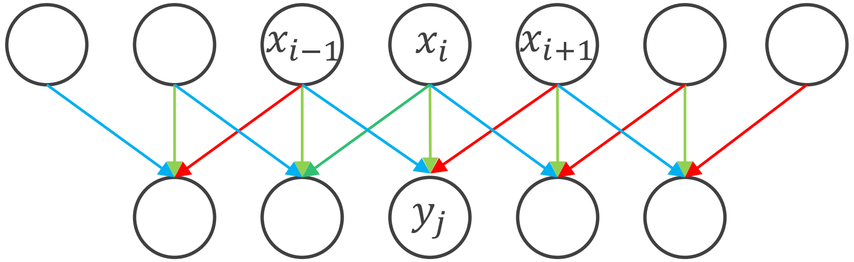


$$y_j = w_{j1}x_1 + \cdots + w_{jn}x_n$$
$$= \sum_i w_{ji}x_i$$

Weight matrix w

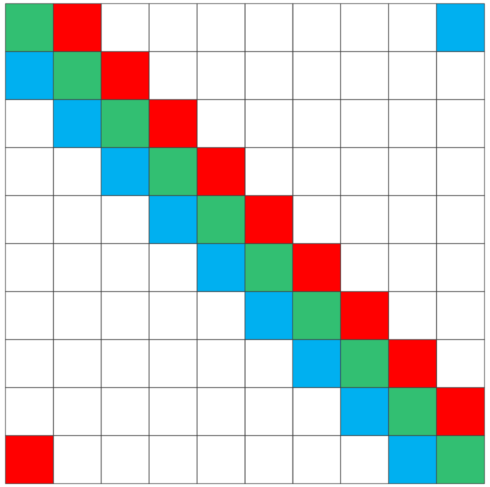


- Convolution ⇔ Block diagonal matrix multiplication
  - Sharing weights after shifting them



$$y_j = w_{j,i-1}x_{i-1} + w_{j,i}x_i + w_{j,i+1}x_{i+1}$$

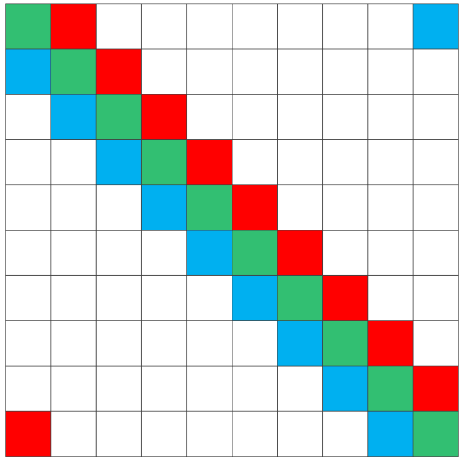
Convolutional weight matrix w



Convolutions as circulant matrices

- Convolutional matrices $C(w)$ are circulant
- Multidiagonal matrices
  - Each column (row) as above but shifted once to the right (below)

$$C(w) = \begin{array}{l} \begin{bmatrix} \boxed{\begin{matrix} c_1 \\ c_2 \\ c_3 \end{matrix}} & \begin{matrix} c_3 \\ c_1 \\ c_2 \end{matrix} & \begin{matrix} c_2 \\ c_3 \\ c_1 \end{matrix} \end{bmatrix} \\ \quad\;\; w \end{array}$$



https://towardsdatascience.com/deriving-convolution-from-first-principles-4ff124888028

Circulant matrices commute

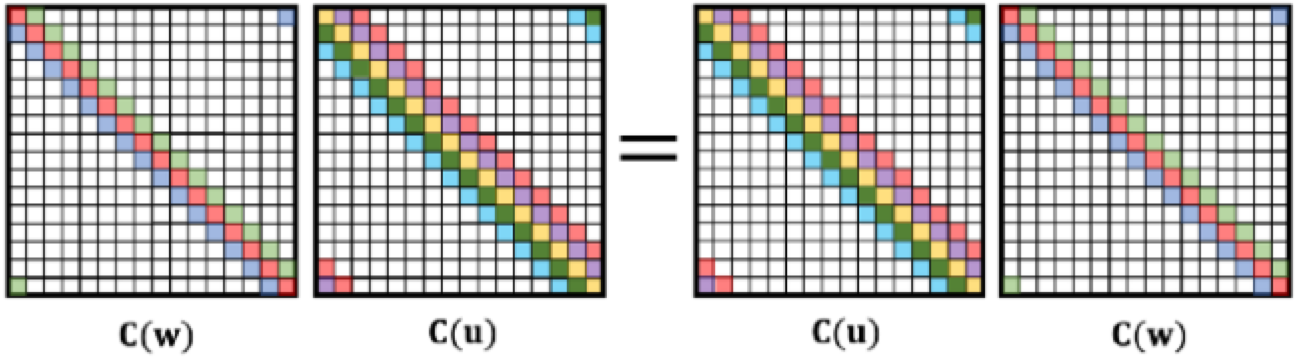
- In general

$$A \cdot B \neq B \cdot A$$

- For circulant matrices

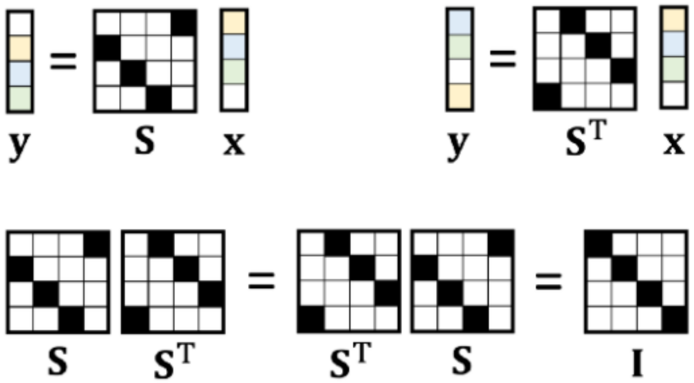
$$C(w) \cdot C(u) = C(u) \cdot C(w)$$

- $\Rightarrow$ Convolutions commute



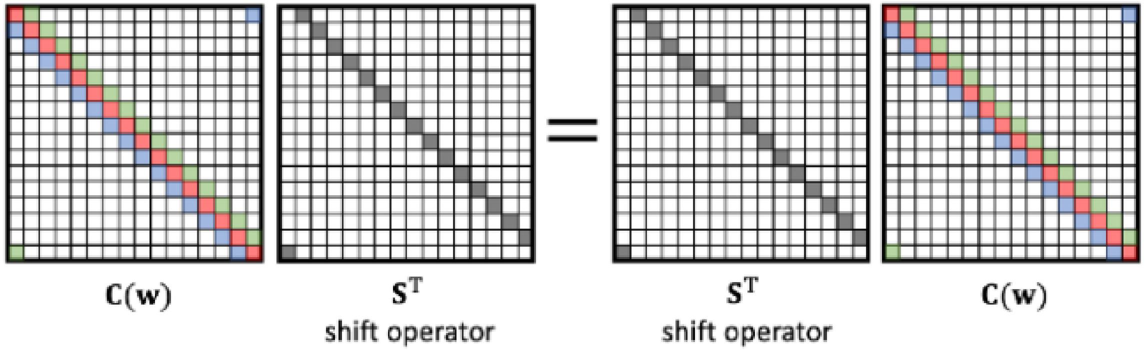
The shift operator

- For $w = [0, 1, \dots, 0] \Rightarrow C(w)$ the right-shift operator
  - Similar to convolution: 'shift once' to the right
  - Transpose for left-shift
- The shift operator is an orthogonal matrix
- The shift operator is also circulant



Circulant matrices ⇔ Translation equivariance

- Circulant matrices enable translation equivariance to convolutions
  - Change the location of the input
  - The results will be the same (but shifted)



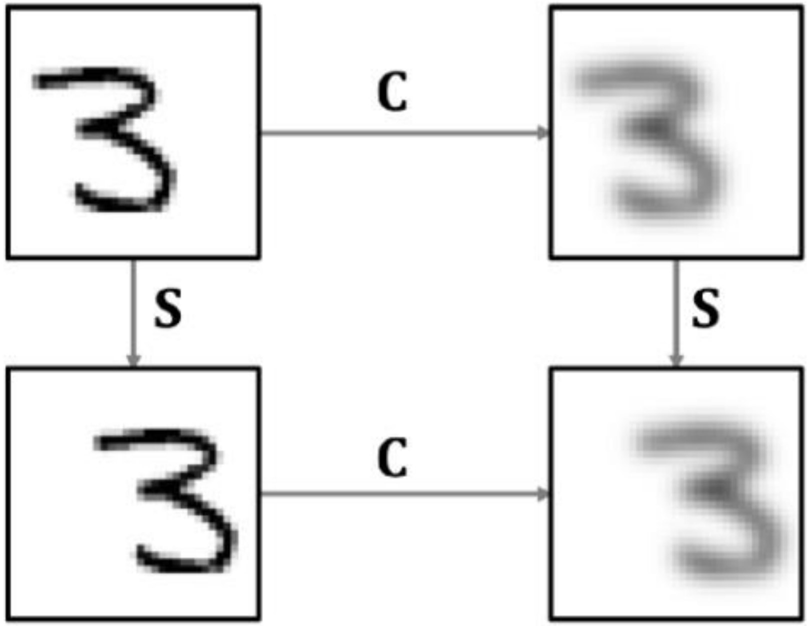


Illustration of shift equivariance as the interchangeability of shift and blur operations.

All circulant matrices have the same eigenvectors!

https://github.com/mitmath/1806/blob/master/lectures/Circulant-Matrices.ipynb

- The eigenvalues of the circulant matrices are different
- But the eigenvectors always the same!
  - The eigenvectors of the "translation transformation/operator"

All circulant matrices have the same eigenvectors!

https://github.com/mitmath/1806/blob/master/lectures/Circulant-Matrices.ipynb

- The eigenvalues of the circulant matrices are different
- But the eigenvectors always the same!
  - The eigenvectors of the "translation transformation/operator"

In [26]:
```
A = circulant([-1, 2, 1, 0, 0])
print('Circulant matrix')
print(A)

eigvals, eigvecs = np.linalg.eig(A)
print('\nEigenvalues')
print(eigvals)

print('\nEigenvectors')
print(eigvecs)
```

```
Circulant matrix
[[-1  0  0  1  2]
 [ 2 -1  0  0  1]
 [ 1  2 -1  0  0]
 [ 0  1  2 -1  0]
 [ 0  0  1  2 -1]]

Eigenvalues
[ 2.  +0.j   -1.19+2.49j -1.19-2.49j -2.31+0.22j -2.31-0.22j]

Eigenvectors
[[ 0.45+0.j    0.14-0.43j  0.14+0.43j -0.36+0.26j -0.36-0.26j]
 [ 0.45+0.j   -0.36-0.26j -0.36+0.26j  0.45+0.j    0.45-0.j  ]
 [ 0.45+0.j   -0.36+0.26j -0.36-0.26j -0.36-0.26j -0.36+0.26j]
 [ 0.45+0.j    0.14+0.43j  0.14-0.43j  0.14+0.43j  0.14-0.43j]
 [ 0.45+0.j    0.45+0.j    0.45-0.j    0.14-0.43j  0.14+0.43j]]
```

In [40]:
```
v = np.random.rand(3)
z = np.zeros(2)
A = circulant(np.append(v, z))
print('Circulant matrix')
print(A)

eigvals, eigvecs = np.linalg.eig(A)
print('\nEigenvalues')
print(eigvals)

print('\nEigenvectors')
print(eigvecs)
```

```
Circulant matrix
[[0.87 0.   0.   0.61 0.13]
 [0.13 0.87 0.   0.   0.61]
 [0.61 0.13 0.87 0.   0.  ]
 [0.   0.61 0.13 0.87 0.  ]
 [0.   0.   0.61 0.13 0.87]]

Eigenvalues
[1.61+0.j   0.42+0.48j 0.42-0.48j 0.95+0.5j  0.95-0.5j ]

Eigenvectors
[[ 0.45+0.j    0.14-0.43j  0.14+0.43j -0.36-0.26j -0.36+0.26j]
 [ 0.45+0.j   -0.36-0.26j -0.36+0.26j  0.45+0.j    0.45-0.j  ]
 [ 0.45+0.j   -0.36+0.26j -0.36-0.26j -0.36+0.26j -0.36-0.26j]
 [ 0.45+0.j    0.14+0.43j  0.14-0.43j  0.14-0.43j  0.14+0.43j]
 [ 0.45+0.j    0.45+0.j    0.45-0.j    0.14+0.43j  0.14-0.43j]]
```

Circulant eigenvectors $\Leftrightarrow$ Shift eigenvectors

- All circulant matrices have the same eigenvectors (or better eigenspace)
  - The shift operator is a circulant matrix
- The circulant eigenvectors are the eigenvectors of shift
  - No wonder they are the same: shift is always the same
- Any convolution with any filter $\boldsymbol{w}$ involves the same eigenvectors!

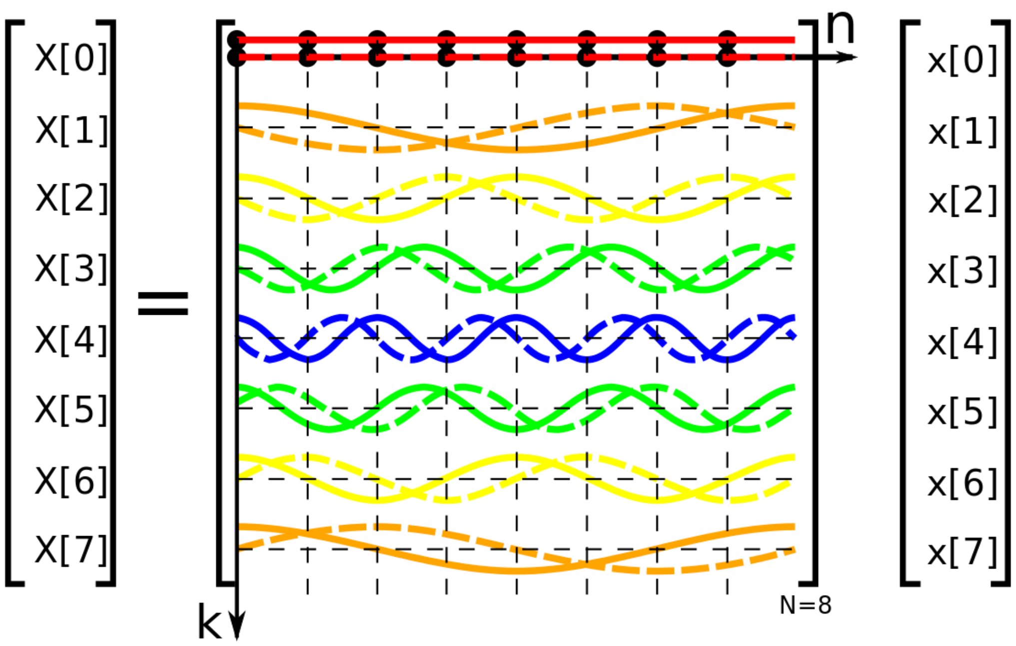
Eigenvectors of circulant matrices

- The k-th eigenvector of $n \times n$ circulant

$$e^{(k)} = \begin{bmatrix} \omega_n^{0 \cdot k} \\ \omega_n^{1 \cdot k} \\ \omega_n^{2 \cdot k} \\ \vdots \\ \omega_n^{(n-1) \cdot k} \end{bmatrix}, \text{where } \omega_{\mathrm{n}} = \exp(\frac{2\pi \cdot i}{n})$$

- Collecting all eigenvectors

$$\Phi = [e^{(0)} \quad e^{(1)} \quad e^{(2)} \quad \dots \quad e^{(n-1)}]$$



Circulant matrix eigenvectors ⇒ Discrete Fourier Transform

- This looks a lot like Discrete Fourier Transform
  - The computer friendly Fourier Transform

$$
\begin{aligned}
X_k &= \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}kn} \\
&= \sum_{n=0}^{N-1} x_n \cdot \left[\cos\left(\frac{2\pi}{N}kn\right) - i \cdot \sin\left(\frac{2\pi}{N}kn\right)\right],
\end{aligned}
\qquad \text{(Eq.1)}
$$

- Convolution ⇔ Discrete Fourier Transform

Matrix diagonalization

$$
\begin{aligned}
\boldsymbol{x} * \boldsymbol{w} &= C(\boldsymbol{w}) \cdot \boldsymbol{x} \\
&= \boxed{(\boldsymbol{\Phi} \cdot \boldsymbol{\Lambda}(\mathbf{w}) \cdot \boldsymbol{\Phi}^*)} \cdot \boldsymbol{x} \\
&= \underbrace{\boldsymbol{\Phi} \cdot \underbrace{\left(\boldsymbol{\Lambda}(\mathbf{w}) \cdot \underbrace{(\boldsymbol{\Phi}^* \cdot \boldsymbol{x})}_{\text{Discrete Fourier Transform (with DFT matrix)}}\right)}_{\text{Inner product with eigenvalues of weight matrix}}}_{\text{Inverse Discrete Fourier Transform (with Inverse DFT matrix)}}
\end{aligned}
$$

Convolution theorem

Convolution theorem

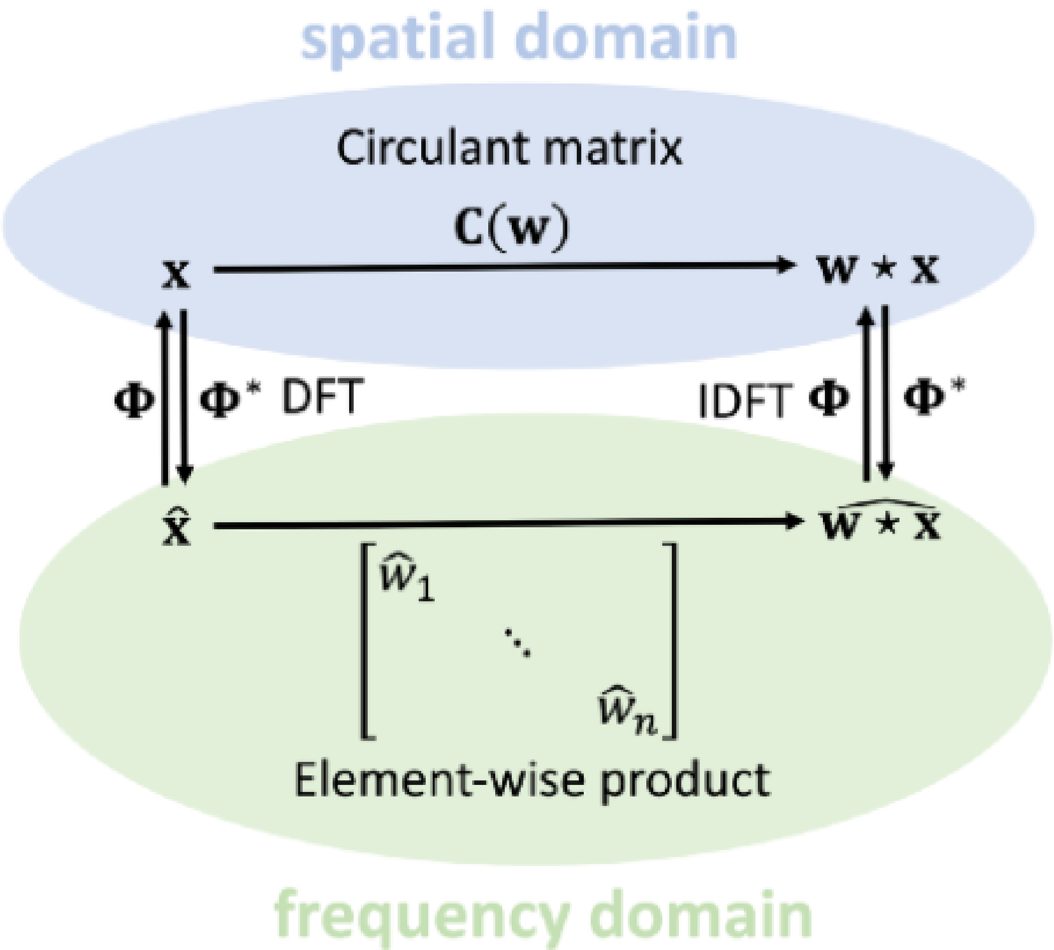
- The Fourier of a convolution equal to dot product of individual Fouriers

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \odot \mathcal{F}\{g\} \Rightarrow f * g = \mathcal{F}^{-1}\big\{\mathcal{F}\{f\} \odot \mathcal{F}\{g\}\big\}$$

- Convolution in "time/space" domain is equivalent to matrix multiplication in "frequency/spectral" domain
  - Frequency defined by Fourier bases $\exp(-\frac{i2\pi}{\mathrm{N}} \cdot kn)$
- Discrete case $\mathcal{F}\{f\}$ becomes a matrix multiplication with shift DFT matrix

$$\boldsymbol{w} * \boldsymbol{x} = \underbrace{\boldsymbol{\Phi}^{-1}\Big(\underbrace{\boldsymbol{\Lambda}(\mathbf{w})}_{\mathcal{F}\{f\}} \cdot \underbrace{(\boldsymbol{\Phi} \cdot \boldsymbol{x})}_{\mathcal{F}\{g\}}\Big)}_{\mathcal{F}^{-1}}$$

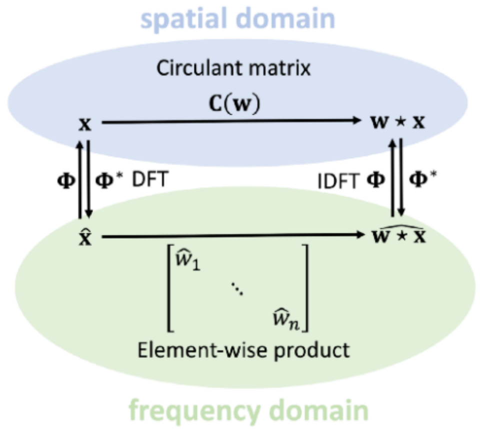
Convolution theorem: $\boldsymbol{x} * \boldsymbol{w} = \boldsymbol{\Phi} \cdot \left(\boldsymbol{\Lambda}(\mathbf{w}) \cdot (\boldsymbol{\Phi}^* \cdot \boldsymbol{x})\right)$



https://towardsdatascience.com/deriving-convolution-from-first-principles-4ff124888028

Convolution theorem: $\boldsymbol{x} * \boldsymbol{w} = \mathbf{\Phi} \cdot \left(\mathbf{\Lambda}(\mathbf{w}) \cdot (\mathbf{\Phi}^* \cdot \boldsymbol{x})\right)$

- If we can compute (inverse) Fouriers and their inverse fast, then we are game
- Fast Fourier Transform (FFT): A faster version of DFT
  - $O(n \log n)$ vs $O(n^2)$
  - Replace sliding window convolutions with very fast matrix multiplications
- Convolution as diagonalization of convolutional circulant matrix



https://towardsdatascience.com/deriving-convolution-from-first-principles-4ff124888028

So what?

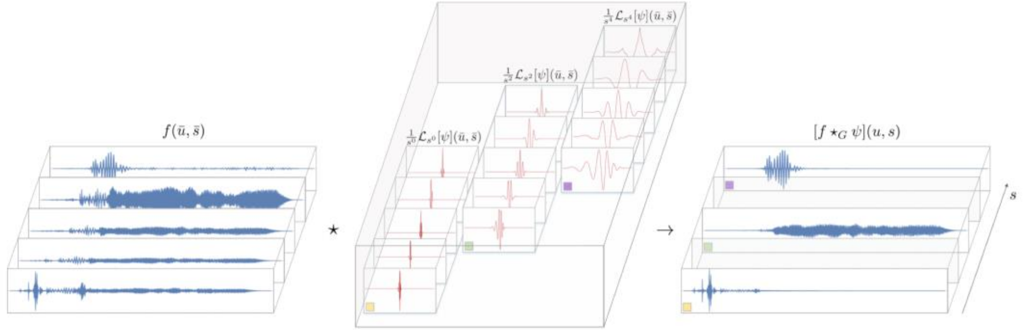
- A more core understanding of what actually convolutions do
- Can we generalize to other equivariances beyond translation?

Group Equivariant Deep Learning

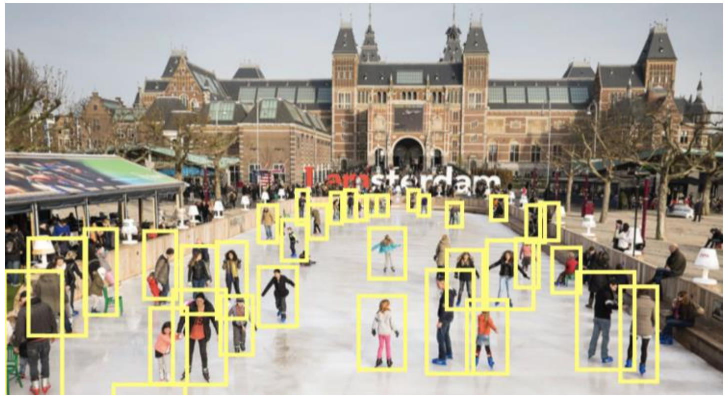
Group convolutional neural networks[1] (G-CNNs) improve over classical CNNs by:

- Allowing weight sharing beyond just translations
- Making geometric data augmentations obsolete
- Data efficiency (one example at a some pose is enough)
- Deal with context (relative poses, like capsule nets)

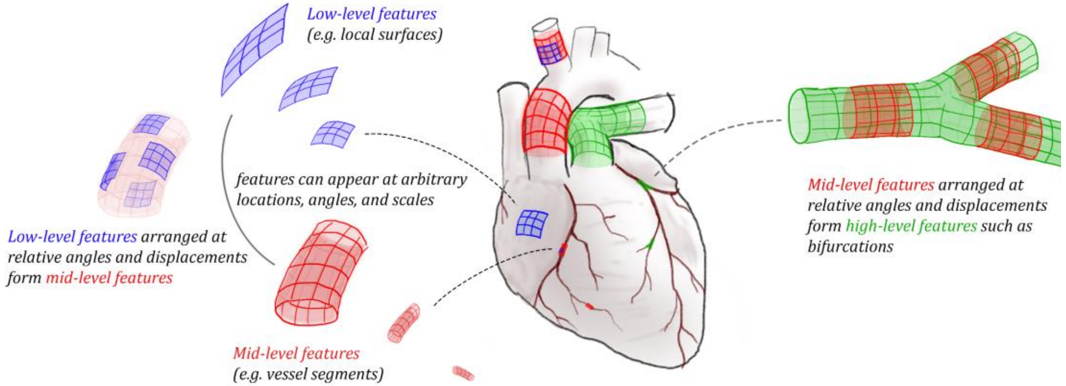
Symmetries in audio[5]
(translation, scale/pitch)



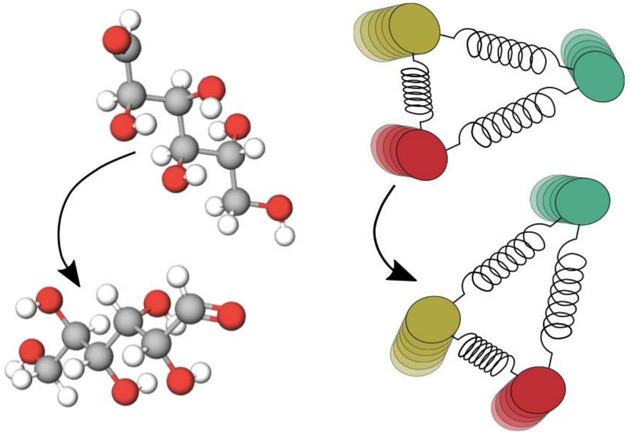
Symmetries in computer vision[3,4]
(translation, scale, rotation, perspective)



Symmetries in medical image analysis[2,3]
(translation, rotation, scale)



Molecular and Physical systems[6]
(translation, rotation, reflection)



[1] Cohen and Welling "Group equivariant convolutional networks" ICML 2016. [2] Bekkers and Lafarge et al. "Roto-translation covariant convolutional networks for medical image analysis." MICCAI 2018. [3] Bekkers "B-spline CNNs on Lie groups" ICLR 2020 [4] Sosnovik, Szmaja, and Smeulders "Scale-equivariant steerable networks." ICLR 2020 [5] Romero, Bekkers, Tomczak, Hoogeboom "Wavelet Networks: Scale Equivariant Learning From Raw waveforms." arXiv:2006.05259 (2020). [6] Finzi, Marc, et al. "Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data." ICML 2020.

So what?

- A more core understanding of what actually convolutions do
- Can we generalize to other equivariances beyond translation?
- Can we generalize to other structures, like graphs?