In the case of an image, we have a 3D array, but instead of considering this finite scenario, we will look at $\mathbb{R}^3$ or in other words "an infinite image". Almost the same calculations will be involved, but with more labels in the finite case.

Assume we have a function on $\mathbb{R}^3$, $f: \mathbb{R}^3 \to \mathbb{R}^C$. It could be the output of a neural networks that assigns to the initial image, its feature map. So, c denotes the channel feature map.

Consider the space

$$L^2(\mathbb{R}^3, \mathbb{R}^C) := \left\{ F: \mathbb{R}^3 \to \mathbb{R}^C \,\middle|\, \int_{\mathbb{R}^3} dx \, \|F(x)\|^2 < \infty \right\}$$

of square integrable functions. On it, we can define a translation operator $T_t: L^2 \to L^2$,

$$(T_t f)(x) = f(x-t)$$

This is in other words, a representation of

$(\mathbb{R}^3, +) \longrightarrow GL(L^2(\mathbb{R}^3, \mathbb{R}^C))$.

We will next look at an anzatz of how the operation of a neural network could look like:

$$(\mathcal{I}_k f)(x) := \int_{\mathbb{R}^3} dy\, K(x,y)\, f(y)$$

$$\mathcal{I}_K : L^2(\mathbb{R}^3, c^{in}) \to L^2(\mathbb{R}^3, c^{out}), \quad K : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^{c^{out} \times c^{in}}$$

Where c^{in}, c^{out} can represent the input and output feature maps of the neural network.

First symmetry (or equivariance) is translational. If our $\mathcal{I}_k$ satisfies this, then

$$\mathcal{I}_k(T_t f) = T_t(\mathcal{I}_k f), \quad \forall t \in \mathbb{R}^3.$$

In other words, if the input is translated, then the feature maps translate by the same factor.

$$I_k(T_t f)(x) = \int_{\mathbb{R}^3} dy\, K(x,y)\, f(y-t)$$

$$= \int_{\mathbb{R}^3} d\tilde{y}\, K(x, \tilde{y}+t)\, f(\tilde{y})$$

$$= \left(T_t (I_k f)\right)(x) = (I_k f)(x-t)$$

$$= \int_{\mathbb{R}^3} dy\, K(x-t, y)\, f(y)$$

The two are actually equal $\Leftrightarrow K(x-t, y) = K(x, y+t)$, $(\forall)\, y \in \mathbb{R}^3$.

If we change y to $y-t \Rightarrow K(x,y) = K(x-t, y-t)$

So, the kernel has a "spatial weight sharing", the weights at (x,y) are the same weights at $(x-t, y-t)$.

If we choose $t = y \Rightarrow K(x,y) = K(x-y, 0)$

$\Rightarrow$ the kernel depends only on $x-y$. So,

$K : \mathbb{R}^3 \to \mathbb{R}^{C_{in} \times C_{out}}$.

If we have nonlinearities that suppose they are local, meaning $(\sigma f)(x) := \sigma_x(f(x))$ A different function at each $x \in \mathbb{R}^3$. If we require symmetry:

$$(\sigma(T_t f))(x) = (T_t(\sigma f))(x)$$

Hence, $(\sigma(T_t f))(x) = \sigma_x(T_t f(x))$

$= \sigma_x(f(x-t)) = (T_t(\sigma f))(x) = (\sigma f)(x-t)$

$= \sigma_{x-t}(f(x-t))$

$\Leftrightarrow \sigma_x = \sigma_{x-t} \Rightarrow \sigma_x = \sigma_0 := \sigma$. Therefore, the nonlinearity is a constant, independent of the position x.

Let's have a look at another crucial operation in convolutional networks i.e. local max pooling (LMP).

$LMP: L^2(\mathbb{R}^3, \mathbb{R}^C) \to L^2(\mathbb{R}^3, \mathbb{R}^C)$,

$f \mapsto \max_{y \in R_x} f(y)$, where R_x is a region where the operation is performed.

Symmetry again, requires:

$$(LMP(T_t f))(x) = (T_t(LMPf))(x)$$

$$\Leftrightarrow \max_{y \in R_x} (T_t f)(y) = \max_{y \in R_x} f(y-t) = \max_{y \in R_x - t} f(y)$$

$$= (T_t(LMPf))(x) = (LMPf)(x-t) = \max_{y \in R_{x-t}} f(y)$$

$\Leftrightarrow R_x - t = R_{x-t} \Leftrightarrow$ the pooling windows are shared (have the same size in the context of CNN's.).