The local translation invariance property implies that if the translation t is small relative to the scale 2^J, that is:

$$|t| << 2^J \Rightarrow |t|/2^J << 1$$

then $\Phi(x)$ and $\Phi(x_t)$ are nearly identical since:

$$\|\Phi(x) - \Phi(x_t)\|_2 \leq C \cdot \underbrace{\frac{|t|}{2^J}}_{<<1} \cdot \|x\|_2 << 1$$

An equivariant representations may be useful in its own right. For example, if you are considering the force acting on a body, the force is equivariant with respect to translations and rotations of the body. Equivariant representations are also important for extracting invariant representations. Indeed, suppose $\tilde{\Phi}(x)$ is a translation equivariant representation. Then:

$$\phi(x) = \langle \alpha, \nabla(\tilde{\Phi}(x)) \rangle = \sum_n \alpha(n) \nabla(\tilde{\Phi}(x)(n)) \in \mathbb{R} \qquad (*)$$

is invariant to translations of x, that is:

$$\phi(x_t) = \phi(x)$$

Notice that correlation/convolution yield translation equivariant representations since

$$(x_t * w)(n) = (x * w)(n-t)$$

Fully invariant representations $\Phi(x) = (\phi_\lambda(x))_{\lambda \in \Lambda}$ consisting of components $\phi_\lambda(x)$ defined through $(*)$ are useful when one has a known global invariance prior. This may be the case in data driven problems coming from chemistry, physics, biology, and statistical modeling of time series, among other contexts. For example, the potential energy of a many body system is invariant to global translations and rotations of the system, and the statistics of a stationary stochastic process are invariant to translations. These types of globally invariant representations are also useful for processing data that can be modelled as an abstract graph $G = (V, E)$, consisting of vertices V connected by edges E. In order to compare two graphs G_1 and G_2 we need a representation $\Phi(G)$ that is invariant to the order in which the vertices are enumerated.

On the other hand, in image processing tasks in computer vision, global translation invariance is often too inflexible. Rarely does one encounter large, global translations (or rotations) of images, but smaller translations and rotations are more common.

A pooling operation increases local translation invariance by a factor of 2. Therefore if we incorporate J pooling operations in our neural network, the local translation invariance of the network will be up to scale 2^J. Note this only works because the linear operations are translation equivariant convolution operators.

Notice that the translation invariance properties, even the one with scale 2^J, still refer to translations that act on the whole signal. Many signal deformations of interest act locally on the signal. We can define these mathematically as diffeomorphisms, and think of them as generalized translations. In order to develop this framework, it is useful to model the data point x as a function. We have:

- 1D: $x : \mathbb{R} \to \mathbb{R}$
- 2D: $x : \mathbb{R}^2 \to \mathbb{R}$
- 3D: $x : \mathbb{R}^3 \to \mathbb{R}$

The discrete data can be considered a sampling of these functions, that is $x(n)$ is the evaluation of x at $n \in \mathbb{Z}$ and on the computer we store $(x(n))_{0 \le n < N}$.

Let us consider $x : \mathbb{R} \to \mathbb{R}$, that is 1D signals, keeping in mind that everything can be generalized to 2D and 3D signals.

Let $\tau : \mathbb{R} \to \mathbb{R}$ with $\tau \in C^2(\mathbb{R})$ and

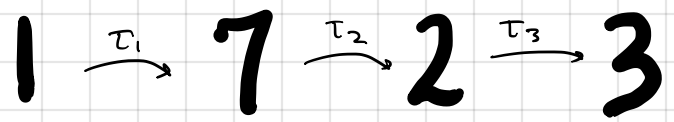
$$\|\tau'\|_\infty = \sup_{u \in \mathbb{R}} |\tau'(u)| \le \frac{1}{2}$$

Then the mapping $u \mapsto u - \tau(u)$ is a diffeomorphism with displacement field τ, that is, $u \in \mathbb{R}$ gets moved to $u - \tau(u)$, which displaces u by $\tau(u)$. We can model deformations of our data through such diffeomorphisms as:

$$x_\tau(u) = x(u - \tau(u))$$

Notice that if $\tau(u) = t$, then this operation is a translation. But this model allows us to study "local" translations and other operations that deform the data locally.

Unlike translations in which we may wish for a translation invariant representation, i.e. $\Phi(x_t) = \Phi(x)$ where $x_t(u) = x(u - t)$, encoding invariance over diffeomorphisms is too strong. Indeed the diffeomorphism group is infinite-dimensional (for data $x : \mathbb{R}^p \to \mathbb{R}$ the translation group is p-dimensional) and one can string together small diffeomorphisms to go between vastly different data points, e.g., in MNIST:



Therefore diffeomorphism invariance is far too strong since we would classify too many things as the same. Rather we seek a representation that is stable to diffeomorphisms, meaning:

$$\|\Phi(x) - \Phi(x_\tau)\|_2 \leq C \cdot \text{size}(\tau) \cdot \|x\|_2 \qquad (*)$$

Later we will discuss CNNs in which $\text{size}(\tau)$ depends on $\|\tau'\|_\infty$ and $\|\tau''\|_\infty$. In particular if τ is a translation then $\text{size}(\tau) = 0$, so $(*)$ will imply global translation invariance. If we want only translation invariance up to the scale 2^J we can amend $(*)$ as:

$$\|\Phi(x) - \Phi(x_\tau)\|_2 \leq C \cdot \left[2^{-J}\|\tau\|_\infty + \underbrace{\text{size}(\tau)}_{F(\|\tau'\|_\infty, \|\tau''\|_\infty)} \right] \|x\|_2$$

$2^{-J}\|\tau\|_\infty$ measures the translation part of τ.

Multiple channels

Since a Toeplitz weight matrix only implements one convolutional filter, the expressiveness of the network will be pretty limited. CNNs rectify this by using many filters in each layer. This also allows CNNs to encode additional invariants on top of translation invariance. We will explain how stacking multiple filters works using the VGG network as a model. This will also explain how color images are processed.

A color image x can be modeled as $x : \mathbb{R}^2 \to \mathbb{R}^3$, in which:

$$x(u) = (x_r(u), x_g(u), x_b(u)), \quad u = (u_1, u_2) \in \mathbb{R}^2$$

and where x_r is the red channel, x_g is the green channel, and x_b is the blue channel. The first hidden layer of the VGG network processes x using a bank of $3M_1$ filters, M_1 filters for each channel:

$$x \mapsto \left((x_r * w^{(1)}_{r,j})_{j=1}^{M_1}, (x_g * w^{(1)}_{g,j})_{j=1}^{M_1} (x_b * w^{(1)}_{b,j})_{j=1}^{M_1} \right)$$

Note this gives us $3M_1$ "images."
These responses are combined across the channels:

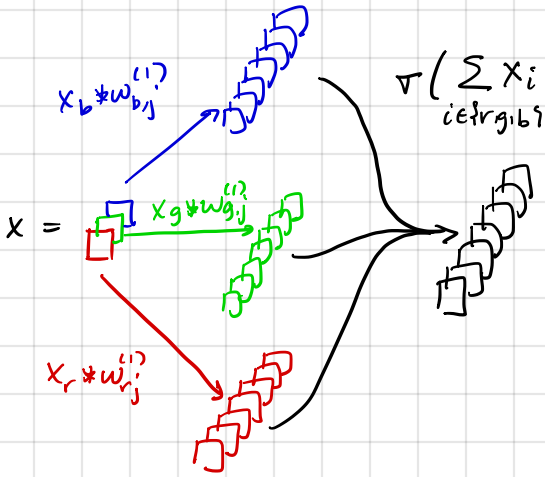
$$x \mapsto \sum_{i \in \{r,g,b\}} x_i * w^{(1)}_{ij}$$

and a nonlinearity is applied: $x \mapsto x^{(1)}_j = \sigma\left(\sum_{i \in \{r,g,b\}} x_i * w^{(1)}_{ij} \right), \quad 1 \leq j \leq M_1$

(e.g. ReLU)

Thus we have taken the original image $x : \mathbb{R}^2 \to \mathbb{R}^3$ and mapped it into a new M_1 channel image $x^{(1)} : \mathbb{R}^2 \to \mathbb{R}^{M_1}$ with:

$$x^{(1)}(u) = \left(x_j^{(1)}(u) \right)_{j=1}^{M_1}, \quad u \in \mathbb{R}^2$$



$$\sigma\left(\sum_{i \in \{r,g,b\}} x_i * w_{ij}^{(1)} \right) = x_j^{(1)}$$

First hidden layer of the VGG network

The second hidden layer works similarly, except that instead of having the three RGB channels as input it has the M_1 channels from the first hidden layer as input:

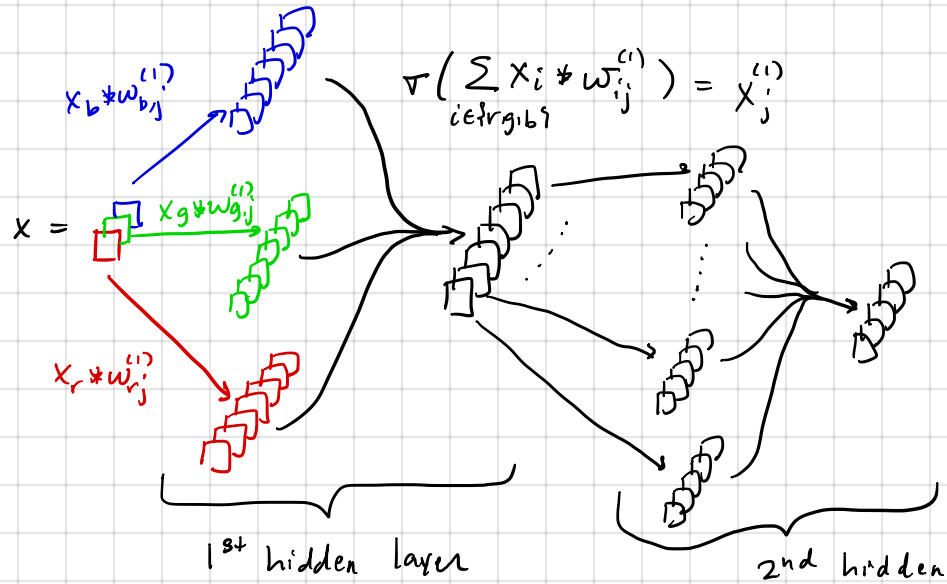
$$x^{(1)} = \left(x_j^{(1)} \right)_{j=1}^{M_1} \longmapsto x^{(2)} = \left(x_k^{(2)} \right)_{k=1}^{M_2}$$

$$x_k^{(2)} = \sigma\left(\sum_{j=1}^{M_1} x_j^{(1)} * w_{jk}^{(2)} \right), \quad 1 \le k \le M_2$$

$$= \sigma\left(\sum_{j=1}^{M_1} \sigma\left(\sum_{i \in \{r,g,b\}} x_i * w_{ij}^{(1)} \right) * w_{jk}^{(2)} \right)$$

Thus the output of the 2nd hidden layer has M_2 channels:

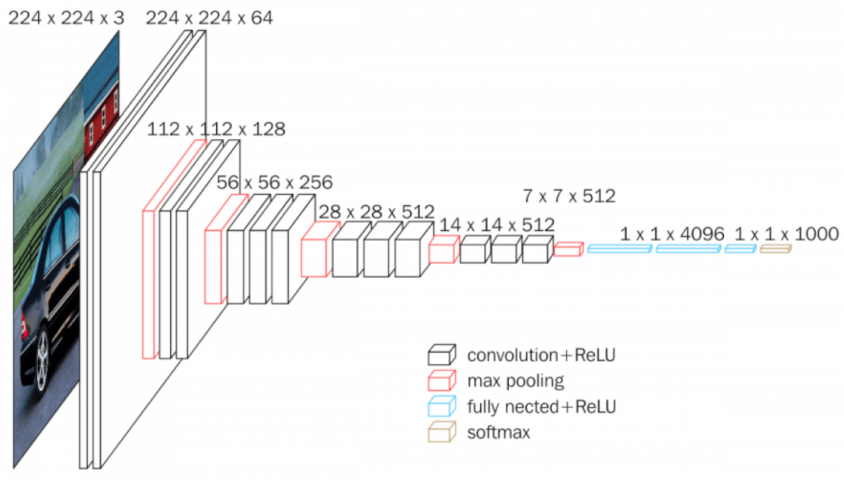
$$x^{(2)} : \mathbb{R}^2 \to \mathbb{R}^{M_2}$$



$$\sigma\left(\sum_{i \in \{r,g,b\}} x_i * w_{ij}^{(1)} \right) = x_j^{(1)}$$

1st and 2nd hidden layers of the VGG network

Subsequent layers work similarly. In some of the hidden layers a max pooling operation is also applied. Here is a diagram of the VGG network:



These stacks of filters within each layer give the CNN increased capacity for distinguishing between multiple types of signals. They also allow the CNN to encode invariants over groups other than the translation group. Convolution is equivariant with respect to the translation group, but not other groups such as the rotation group. However, the stack of filters can be learned to be equivariant with respect to other group actions. We will show later how this works for rotations.

CNNs from the perspective of approximation theory

The research on approximation theory for CNNs is limited and we will not spend much time on it. We mention three results:

1.) Poggio, et al (2017) - We already discussed this result at length. CNNs are a special case of compositional networks in which the weights are shared and the composed dimensions are organized geometrically.

2.) Zhou - "Universality of deep CNNs" (2018): Universal approximation by CNNs as the # of layers $L \to \infty$. additional

3.) Petersen & Voigtlaender - "Equivalence of approximation by CNNs and fully connected networks" (2018): Rates of approximation by CNNs and ANNs are the same.