Dihedral equivariant convolutional layers

March 31, 2017

0.1 Symmetry

In image recognition, it happens frequently that the target result is invariant under some spacial transformations. For instance, the vertical mirror is often used in the recognition of common objects (cars, cats, dogs, ...). With astrophysical images, the target result is often invariant under rotations, mirrors and scales. Instead of taking all the symmetries into account, it can be numerically interesting to restrict them to a symmetry group which is computationally simple. In this project, the dihedral group has been used. It is formed of rotations of 90 degrees and mirrors (see figure 1). This group is computationally simple because it does not require any interpolation. The absence of interpolation ensures no loss of information.

0.1.1 Augmentation

The symmetry group can be used to generate more images to train the NN, this is called *augmentation*. By augmenting the training set with the transformations of the dihedral group, we get 8 times more images for the training. It can be also used to improve the prediction by computing a result for each transformation of the image and merge all these in one final prediction.

0.1.2 Invariant neural network

At the end of the training, the NN will eventually discover by itself all the symmetries of the problem.

Rather than relying on the training to discover the symmetries we can enforce some of them from the beginning (Dieleman et al., 2016). The idea is to create an architecture that is itself invariant under some symmetry group G, that is to say a function f satisfying equation (1) for all set of parameters W.

$$\forall W,\ f(x,W) = f(\tau x, W) \quad \forall \tau \in G \tag{1}$$

In the following, the mathematical notions of group and representation will allow us to express precisely a sufficient condition on the layers to ensure the invariance. Two layers satisfying this condition will be derived in details. The mathematical formalism will help us a lot in the derivation of the more complicated one.

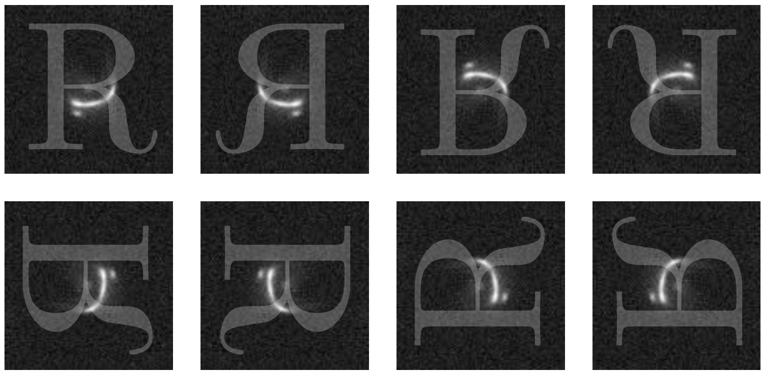

Figure 1: The dihedral group is composed of 8 transformations

To found what condition must satisfy our layers, we need to look how the values of the intermediate tensors will transform if the input image x is replaced by its transformation by an element of the group τx. The values of the intermediate tensors must transform in a way that their informations are preserved. Let y be the value of some intermediate tensor and y' be its value after having replacing the input x by τx, This transformation bringing y into y' should depend only on τ and be invertible. This must be the case because y and y' must contain the same information just in a different form that only depend on τ.

Therefore to build up an invariant NN, the output of each layer must transform with a representation of the group $(G, \circ)$. A representation D is a mapping from the group to linear transformations such that

- $D[e] = 1$ where e is the identity of the group and 1 is the identity application on the vectorial space.
- $D[\tau_1 \circ \tau_2] = D[\tau_1]D[\tau_2]$

As pictured in "equation" (2), if the input x is changed into τx, each intermediate tensor must transform with some representation. By doing so, we can ensure that the output changes with the trivial representation[1]; in other words, it will be invariant.

$$\begin{tikzcd} x \arrow[r, "\text{layer 1}"] & y_1 \arrow[r, "\text{layer 2}"] & y_2 \arrow[r] & \cdots \arrow[r] & p \\ \tau x \arrow[r, "\text{layer 1}"] & D_1[\tau]y_1 \arrow[r, "\text{layer 2}"] & D_2[\tau]y_2 \arrow[r] & \cdots \arrow[r] & p \end{tikzcd} \tag{2}$$

Therefore each layer must be *equivariant.* A function g is called G-equivariant if

$$\exists D_1, D_2, \forall x \forall \tau \in G, \quad g(D_1[\tau]x) = D_2[\tau]g(x)$$

where D_1 and D_2 are two representations.

The following representations will be used:

- *Trivial* or *invariant representation*: on any vectorial space, it is the representation which associates the identity application to any element of the group.
- *Defining representation*: The ensemble of square images made of pixels forms a vectorial space of dimension equal to the number of pixels. The dihedral transformation (rotations by 90 degrees, mirrors) acting of such images can be viewed as permutation matrices. These matrices form a representation. We call this representation the defining representation.
- *Regular representation*: it acts on a vectorial space of dimension equal to the order of the group. On a basis in which each element is associated with an element of the group, the representation is defined as $D[\tau]|\sigma\rangle = |\tau \circ \sigma\rangle$.

It is also important to note that the tensor product of two representations is a representation: $D[\tau] = D_1[\tau] \otimes D_2[\tau]$.

Now that we have defined properly these notions, we can use a lighter notation. In the following, the representation which is being used can be inferred from the space on which the group is acting; therefore we will use τ instead of $D[\tau]$.

0.1.3 Dihedral convolution

The aim of this section is to build a modification of the convolutional layer equivariant under the dihedral group.

For the sake of simplicity, we will consider that the images and the filters are square. At the end, only the filters and not the images will be rotated, therefore the input image will not be restricted to be square.

Let us denote by $Conv(x, F)$ the convolution of the image x with the filter F. A property of the convolution is that if we transform both the image and the filter with an element of the dihedral group, its equivalent to transform the result of the convolution (3).

$$Conv(\tau x, \tau F) = \tau Conv(x, F), \qquad \tau \in G \tag{3}$$

[1]The trivial representation is $\forall \tau \in G, D[\tau] = 1$

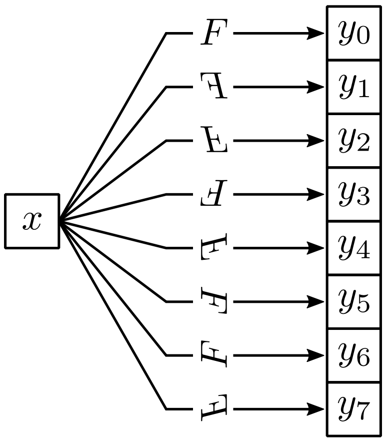

Figure 2: Dihedral convolution from representation $x \longrightarrow \tau x$ to $y_i \longrightarrow \tau y_{\tau^{-1}\circ i}$

First equivariant layer First lets derive an equivariant layer such that its input image transforms with the defining representation $x \longrightarrow \tau x$. This will be the case for at least the first layer. Let see that the convolution alone is not equivariant, it transforms as follows (4)

$$Conv(x, F) \longrightarrow Conv(\tau x, F) = \tau Conv(x, \tau^{-1} F) \tag{4}$$

And this is not equivariant because there is no representation of G (that is to say no linear transformation depending only on τ) which does this transformation.

Instead, if we build the our layer so that it computes various convolutions (not only one), we can force the result of the layer to transform with a representation. To keep the mathematical expressions short, let us choose an ordering of the dihedral group and make a bijection between the integers from 0 to 7 with the elements of the group. The following order has been chosen: $\{\mathrm{R}, \reflectbox{R}, \scalebox{1}[-1]{R}, \rotatebox{180}{R}, \rotatebox{90}{R}, \reflectbox{\rotatebox{90}{R}}, \rotatebox{270}{R}, \reflectbox{\rotatebox{270}{R}}\}$[2]. The new layer is described in equation (5)

$$\boxed{y_i = Conv(x, iF) \quad i \in \{0, \ldots, 7\}} \tag{5}$$

Now when $x \longrightarrow \tau x$ then $y_i \longrightarrow y'_i = \tau Conv(x, \tau^{-1} \circ iF) = \tau y_{\tau^{-1}\circ i}$ which now is a representation. Indeed, it is the tensor product of the defining representation acting on the spacial components with the regular representation acting on the channel components. So we managed to build an equivariant layer (equation (5), illustrated in figure 2).

But we are only done for the case when the input of the layer transforms with the defining representation. We now want to stack layers, so we need an equivariant operation which takes as input a tensor transforming with our new representation: $y_i \longrightarrow y'_i = \tau y_{\tau^{-1}\circ i}$.

Second equivariant layer Let denote by x_i the input of the new equivariant layer we are looking for and by y_j its output. We have room to make some choices to build up this second layer. Let us impose that

- Its input (x_i) and its output (y_i) transform both with the same representation ($x_i \longrightarrow \tau x_{\tau^{-1}\circ i}$)
- We use eight filters $F_i, i \in \{0...7\}$
- $y_j = \sum_i Conv(x_i, \mu(i,j) F_{\sigma(i,j)})$ where μ is an element of G and σ takes integer values between 0 and 7

We just have to find expressions for μ and σ. We can compute the transformation induced by the transformation of the input:

$$y'_j = \sum_i Conv(\tau x_{\tau^{-1}\circ i}, \mu(i,j) F_{\sigma(i,j)}) = \sum_i \tau Conv(x_{\tau^{-1}\circ i}, \tau^{-1} \circ \mu(i,j) F_{\sigma(i,j)}) = \sum_i \tau Conv(x_i, \tau^{-1} \circ \mu(\tau \circ i, j) F_{\sigma(\tau\circ i, j)}) \tag{6}$$

[2]This choice is arbitrary.

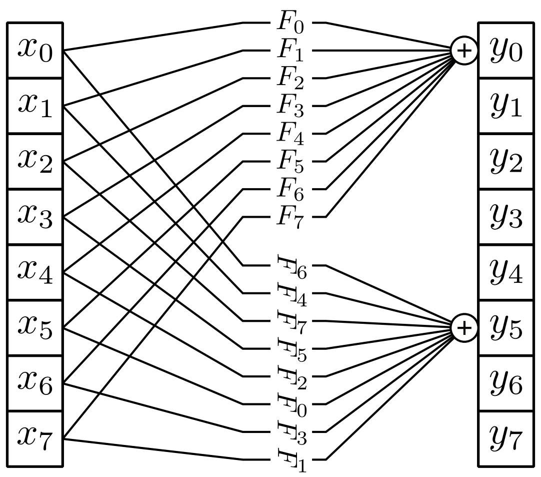

Figure 3: Dihedral convolution from representation $x_i \longrightarrow \tau x_{\tau^{-1}\circ i}$ to $y_i \longrightarrow \tau y_{\tau^{-1}\circ i}$

where we used the property of the convolution and reordered the sum. Using that we imposed how the y_js transform:

$$y'_j = \tau \sum_i Conv(x_i, \mu(i, \tau^{-1} \circ j) F_{\sigma(i,\tau^{-1}\circ j)}) \tag{7}$$

By identification of equations (6) and (7):

$$\begin{cases} \tau^{-1} \circ \mu(\tau \circ i, j) = \mu(i, \tau^{-1} \circ j) & \Leftrightarrow \mu(\tau \circ i, \tau \circ j) = \tau \circ \mu(i,j) \\ \sigma(\tau \circ i, j) = \sigma(i, \tau^{-1} \circ j) & \Leftrightarrow \sigma(\tau \circ i, \tau \circ j) = \sigma(i,j) \end{cases} \tag{8}$$

Several solutions to equations (8) exist: $\mu(i,j) = i$ or $\mu(i,j) = j$ and $\sigma(i,j) = i^{-1} \circ j$, $\sigma(i,j) = j^{-1} \circ i$ or $\sigma(i,j) = const$.

Let choose the solution such that $y_0 = \sum_i Conv(x_i, F_i)$ (note that the majority of the solutions are equivalent up to a transformation of the filters). $\mu(i,j) = j$ and $\sigma(i,j) = j^{-1} \circ i$ fulfill all the constraints. So we have found a new equivariant layer (9).

$$\boxed{y_j = \sum_{i=0}^{7} Conv(x_i, j F_{j^{-1}\circ i}) \quad j \in \{0, \ldots, 7\}} \tag{9}$$

The figure 3 illustrates this layer.

0.1.4 Other layers

Other equivariant layers can be easily derived from the two treated in the last section or with equivalent techniques. If we work only with the representations :

$$\text{(defining}\times\text{)regular} \times \cdots \times \text{regular}$$

we can use the following tricks to construct lots of different equivariant layers:

1. Summing over an index is equivariant: $y_{jkl\ldots} = \sum_i x_{ijkl\ldots}$
2. The constant values (like the filters and the weights) should be indexed by $i^{-1} \circ j$: F, $F_{i^{-1}\circ j}$, $F_{i^{-1}\circ k, i^{-1}\circ k}$ and so on[3]
3. Image convolution should be on the form: $Conv(x_{\ldots}, iF_{\ldots})$

With these three rules we can for instance create the following equivariant layers: $y_i = \sum_{jk} Conv(x_{jk}, kF_{j^{-1}\circ i})$ or $y_{ijk} = x \cdot W_{i^{-1}\circ j, j^{-1}\circ k}$ (fully connected layer). The table 1 lists some examples.

The pool max with a $s \times s$ tile is equivariant for both $x \longrightarrow \tau x$ and $x_i \longrightarrow \tau x_{\tau^{-1}\circ i}$ representations as long as the dimension of the input image is a multiple of s.

[3]Because $i^{-1} \circ j = (\tau^{-1} \circ i) \circ \tau^{-1} \circ j$

[4]without mixing of data with different indices

Representation		Equation	Comment
Input	Output		
$x \longrightarrow \tau x$	$y_i \longrightarrow \tau y_{\tau^{-1} \circ i}$	$y_i = Conv(x, iF)$	useful to start the NN (see figure 2)
$x_i \longrightarrow \tau x_{\tau^{-1} \circ i}$	$y_i \longrightarrow \tau y_{\tau^{-1} \circ i}$	$y_j = \sum_i Conv(x_i, jF_{j^{-1} \circ i})$	(see figure 3)
$x_i \longrightarrow \tau x_{\tau^{-1} \circ i}$	$y_i \longrightarrow \tau y_{\tau^{-1} \circ i}$	$y_i = Conv(x_i, iF)$	parallel convolutions [4]
$x_i \longrightarrow \tau x_{\tau^{-1} \circ i}$	$y \longrightarrow \tau y$	$y = \sum_i Conv(x_i, iF)$	useful to get a smaller tensor (see figure 4)
$x_i \longrightarrow x_{\tau^{-1} \circ i}$	$y_i \longrightarrow y_{\tau^{-1} \circ i}$	$y_j = \sum_i x_i \cdot W_{j^{-1} \circ i}$	fully-connected layer
$x_i \longrightarrow x_{\tau^{-1} \circ i}$	$y_i \longrightarrow y_{\tau^{-1} \circ i}$	$y_i = x_i \cdot W$	parallel fully-connected layer

Table 1: Some equivariant layers under the dihedral group. With these layers, invariant NN under the dihedral group can be built.

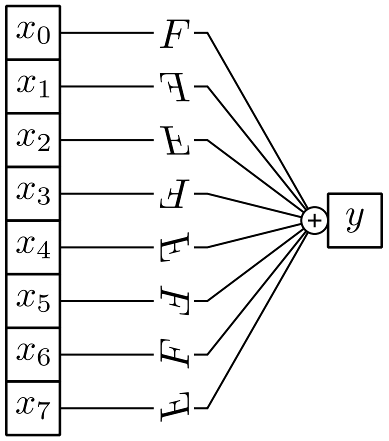

Figure 4: Dihedral convolution from representation $x_i \longrightarrow \tau x_{\tau^{-1} \circ i}$ to $y \longrightarrow \tau y$

References

S. Dieleman, J. De Fauw, and K. Kavukcuoglu. Exploiting Cyclic Symmetry in Convolutional Neural Networks. *International Conference on Machine Learning (ICML)*, 2016.