Interactive CEDAR tutorial—Part 2

Mathis Richter

January 22, 2020

4 Coordinate transforms

We will now extend the model so that it can determine spatial relations between objects in the scene. When we are done, the model will be able to determine, for instance, whether a red object in the scene is to the left or to the right of a green object. For this, we will have to extract the relative position between objects, that is, the position of one object with respect to the position of another object. We will call the first object the *target object* and the other object the *reference object*. Extracting relative positions from absolute positions amounts to a coordinate transformation. The transformation takes the absolute position and shifts the coordinate system, centering it on the position of the reference object. In DFT, coordinate transforms can be implemented using a neural transformation field of higher dimensionality that is read out diagonally, as explained in the lecture. In CEDAR, we use a convolution operation to approximate that neural transformation field.

Start to explore this idea in CEDAR with a `Convolution` step. In its parameters, set the mode of the convolution to `full`. The `Convolution` step takes input from two sources: the first input (top) represents the positions of the target objects, the second input (bottom) represents the reference object. Try this first with simulated "objects". For the target objects, create a small "scene", which you can build from three different `GaussInput` steps that you sum with the `Sum` step. For the reference object, use a single `GaussInput` step, making sure that you flip the output of that step both horizontally and vertically using a `Flip` step.

Plot the position of the target objects, the reference object, and the output of the convolution. Convince yourself that the convolution approximates a coordinate transformation. Move the position of the target objects and the reference object to make sure everything is working as expected.

Once this works, replace the simulated input to the convolution with input from two two-dimensional fields, the *target field* and the *reference field*

(keeping the `Flip` step). We can now connect these fields to the architecture that you have created in previous tasks. Both the *target field* and the *reference field* should receive input from the two-dimensional *space field* (see Task 4). They should only form peaks when there is a peak in the *space field* and when they get additional homogeneous input. That homogeneous input comes from two `Boost` steps, each of which feeds into one of the fields. Tune both fields to form sustained peaks (working-memory); that is, the peaks should remain stable, even if you remove the input from the *space field* as well as from the boost. Feed the output of the convolution into another new field, the *relational field*, which is two-dimensional as well. Make sure that you set the size of this field to match its input from the `Convolution` step. Why is the output of the convolution step suddenly larger than its input?

Now, guide your model to select a red object as the target object. This is achieved once there is a sustained peak in the *target field*. You will have to manually manipulate the input to the *color field* and the homogenous boost input to the *target field*. Afterward, guide the model to select a green object as the reference object, which is achieved once there is a sustained peak in the *reference field*. Convince yourself that the model is extracting the relative position of the target object with respect to the reference object. This is achieved if there is a peak at the correct position in the *relational field*.

What happens if there are multiple red objects in the scene? What happens if there is only a single red object but multiple green objects? What happens if there are both multiple target objects and multiple reference objects? Why? Would it work with a neural transformation field instead of a convolution? Why?