# Interactive CEDAR tutorial—Part 3

Mathis Richter

January 22, 2020

## 5 Spatial relational concepts

Our architecture can now extract the relative position between a target object and a reference object. Let us try to have the architecture give a response about what kind of spatial relation there is between target and reference, for example "the red object is *to the left* of the green object". But first, we have to convert the representation of the relative spatial position into a discrete relational concept (for example LEFT OF or ABOVE. How do we arrive at discrete concepts like this? In DFT, we can create a representation of a categorical concept through an individual neural dynamic node (zero-dimensional field).

Add four such neural dynamic nodes to your architecture, using the step `node`. Name the steps according to the relations they will represent: LEFT OF, RIGHT OF, ABOVE, and BELOW. All of these neural nodes will eventually receive input from the relational field. However, we only want the nodes to become active if the peak in the relational field is in their respective area of the field. For example, if the peak is to the left of the center, we only want the node for the concept LEFT OF to become active. We do this by specifying connection weights between the relational field and each node. Each node has a unique connection pattern, which can be implemented using the `SpatialTemplate` step.[1] To create the pattern between the relational field and a node, feed both the output of that step and the output of the relational field into a `ComponentMultiply` step. From there, feed the activation into a `Projection` step (using the `maximum` setting), which projects it onto 0D, and from there into a `StaticGain` step. From there, the activation is fed into the node.

Tune the parameters of the four connection patterns so that the nodes become active for appropriate peak positions in the relational field.

[1] Due to a bug in CEDAR, if you use the option `invert sides`, you will have to add 0.475 to the output. You can use the `AddConstant` step for that.

The four nodes now represent *categorical concepts* of relations. The decision boundaries between the categories are determined by the connection patterns. If we think of these concepts as connected to words that describe the concept, for example the word "left" for the concept LEFT OF, we can interpret the activation of a concept node as a response. Go through an example where you "ask" the architecture: "Where is the red object with respect to the green one?" until you get a response. (You can also use different colors.) Take note of everything you have to manipulate in the architecture for this to work.

# 6 Color concepts

Now, let us make this a bit more realistic. Until now, we have selected objects based on their color, but we manipulated the input to the color field manually. Let us instead add color concepts that we can then activate as if we were feeding words (like "red") to the architecture.

Add four more dynamic neural nodes (`node`) and name them "red", "green", "blue", and "yellow". These nodes will represent color concepts. In order for the concepts to represent the correct colors, we will have to create connection patterns from each node to the color field. Add a `GaussInput` step, which we will use to approximate the pattern for one node. Connect the output of that step, as well as the output of one of the nodes to a `ComponentMultiply` step. Feed the output of the `ComponentMultiply` step into the color field. Add a `Boost` step as input to the node; this way you can activate the color concept by "verbal input" (by activating the boost). Now, when the node is active, it projects a patterned connection into the color field.

Set up the patterned connections for all four color concepts. Tune the connection patterns so that the color concepts match the colors of the respective objects in the scene. This is probably easiest when looking at the activation of the three-dimensional color-space field.

Go through an example where you "ask" the architecture: "Where is the red object with respect to the green one?" (You can also use different colors.) Use only boost inputs[2] to interact with the architecture.

Could you also ask a question like: "What is to the left of the green object"? Does the architecture have to be extended for this? In what way? Go ahead and make that change, if you like, and go through the exemplary question. :)

[2]If you give the boosts meaningful names, you can use the "Boost control" widget to control your architecture.