# HW2P1 – TOY PROBLEM

This Toy problem aims at helping you understand what your outputs look like and the reasoning any mistakes you are making. In this way, you can debug your problems in a more fine-grained way.

**Folder Structure:**

| toyproblem

- toy.py
- value.py (inputs and other verification values are given here)

**Let's take a look at the network which we have constructed:**

The network will be dealing with inputs which will be of 1-d and 2-d vectors, corresponding to conv1d and conv2d. The testing can be done for the following methods:

a) Verify the output of each layer after activation

b) Verify the gradients with respect to weights, biases and inputs

We want you to use whatever code you have written to complete hw1.py file and play around to get through the test of toy.py. You will start off with a single input and then try out the example for a batch of inputs.

**How to test?**

We have set up verification values in **value.py**, which will be imported into **toy.py**. Run four ***"TODO"*** section of **toy.py**, which will respectively test CONV1D using single input and a batch of inputs, CONV2D using a batch of inputs and CONV2D_DILATION. Once you pass each section, it will print out *"Congratulations! You pass the *** test!"*. If any assert error raises, please refer the value provided in **value.py** by the variable names in corresponding **conv_test()** and debug your code.

**Neural network structure:**

**General:**

Input ->Conv1(stride=1) -> Activation (ReLU) -> Conv2 (stride=2) -> Criterion -> Target

For convenience, we set only one convolution layer for Conv2D with dilation and set a fake derivative of loss for backpropagation.

There are three diagrams about Conv1D, Conv2D and Conv2D_dilation.

*Please notice the diagrams we provide ignore the batch size dimension and channel dimension while we need to take them into consideration in our toy problem.*

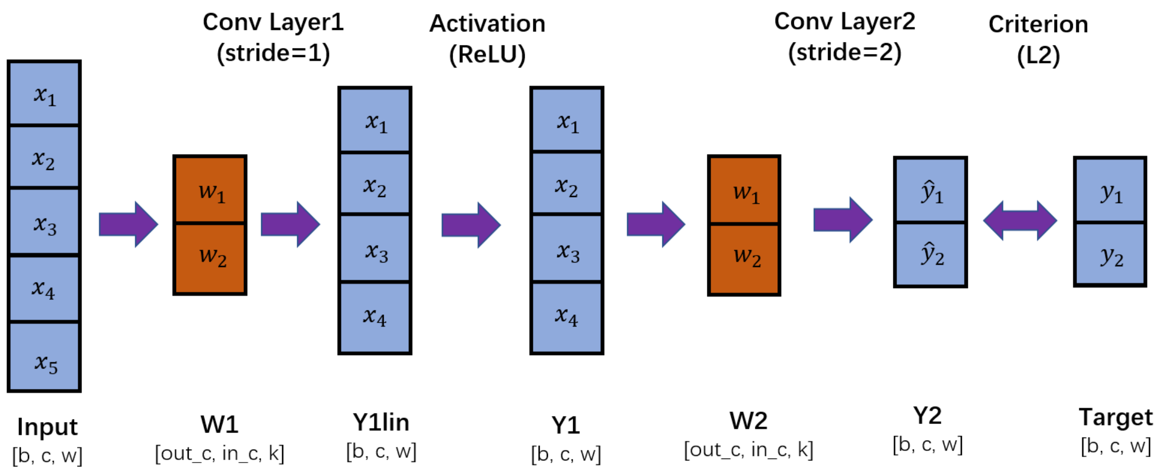
**Notations:**

$$b\text{: }batch\ size \quad c\text{: }channel\ number \quad w\text{: }width \quad h\text{: }height \quad k\text{: }kernel\ size$$

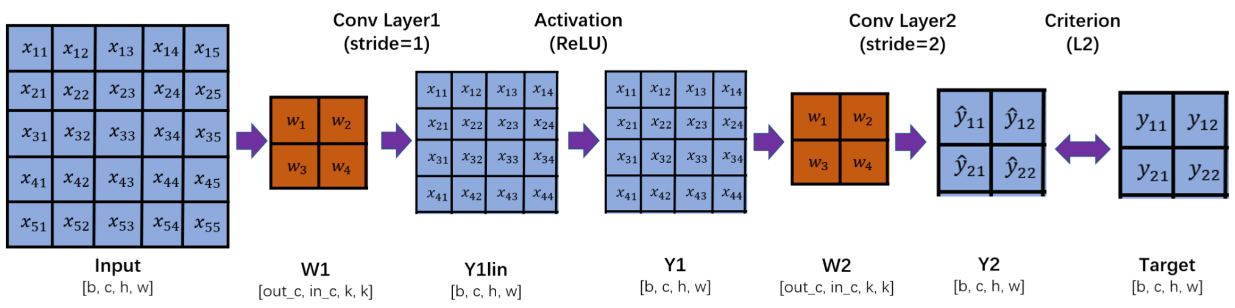
There are three diagrams about Conv1D, Conv2D and Conv2D_dilation

*Ignore bias term for convenience*

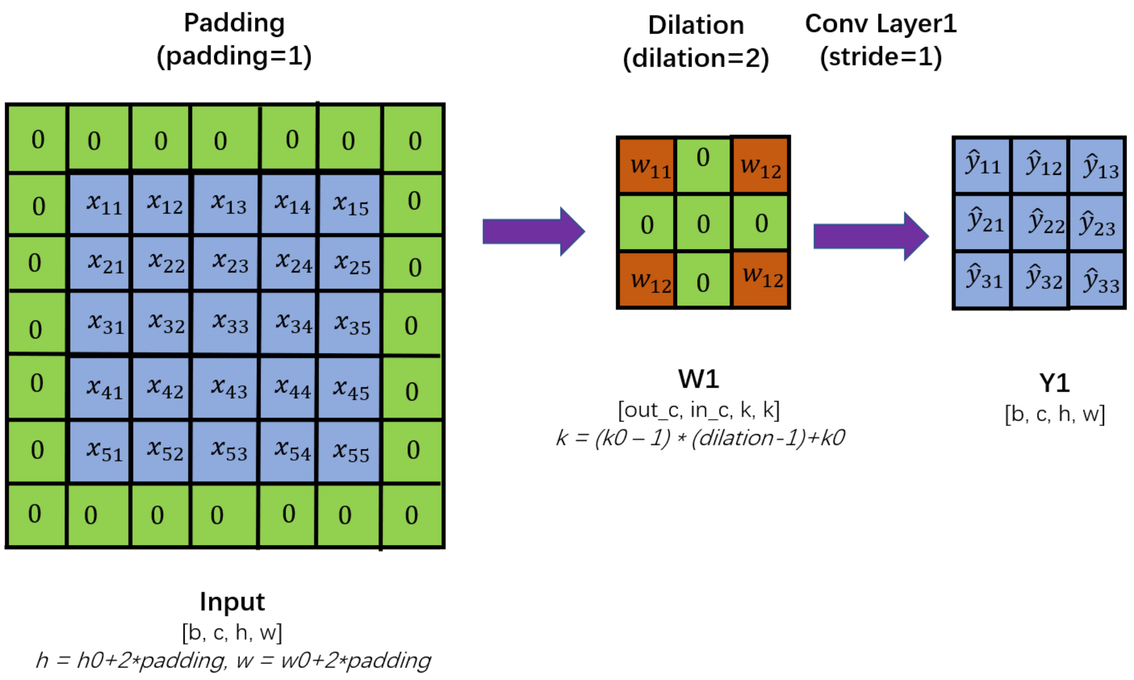
- Conv1D:



l Conv2D:



l Conv2D with dilation:



h0, w0 and k0 respectively represent the original size before padding or dilation.

For more details, please refer to the appendix 7.1.

**Values:**

Please refer to the comments of **toy.py** and all the values you are processing are stored in **value.py**.

(HINTS: all values are written in upper class, and you can print them in toy.py by their names instead of looking forward them in value.py)

Once you collect all four "Congratulations", you can be confident to say that your convolution functions are correct! If you observe carefully, the outputs and the gradients seem large and keep in mind that there was only a single forward pass and a single backward pass. Imagine if we do these 20 or 30 times, the gradients will explode. This happens because of poor weight initialization and owing to ReLU activation function which does not squash the output unlike Sigmoid and Tanh. You can now try the same experiment with different activation functions, like Sigmoid or Tanh.

Best luck for your homework!