University of Tehran

COMPUTER SCIENCE DEPARTMENT

COMPUTATIONAL NEUROSCIENCE

REPORT 7

Convolution and Pooling Connections

Student Name	Student ID
AMIR NADERI	610398126

Lecturer in charge:

M.Ganjtabesh

Submission Date : 14/07/2023

In this project we are trying to make convolution and pooling connections between our Izhikevich layers and analyze the behavior of spikes for each connection using different parameters.

1 Convolution

In mathematics, convolution is an operation performed on two functions (f and g) to produce a third function that expresses how the shape of one is modified by the other. Convolution is one of the most important operations in signal and image processing. It could operate in 1D (e.g. speech processing), 2D (e.g. image processing). In image processing, convolution is the process of transforming an image by applying a kernel over each pixel and its local neighbors across the entire image. The kernel is a matrix of values whose size and values determine the transformation effect of the convolution process. The Convolution Process involves placing the Kernel Matrix over each pixel of the image, multiplies each value of the Kernel with the corresponding pixel it is over, Then sums the resulting multiplied values and returns the resulting value as the new value of the center pixel. This process is repeated across the entire image. So if we were to process the image of higher resolutions the network parameters will be very high and require higher computational power and won't scale for larger images. No one wants to train millions of parameters for the small images, therefore we use convolution layers.
Here is a picture of convolving a filter on an image:

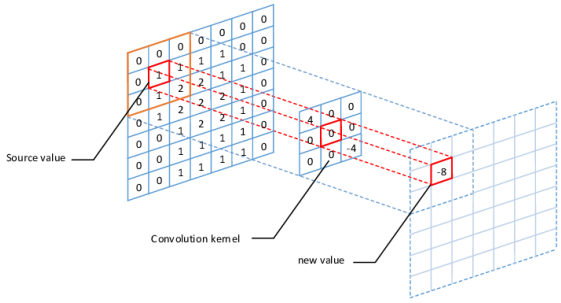

Figure 1: Convolution

Padding will be useful for us to extract the features in the corners of the image. during the convolution process the corner pixels of the image will be part of just a single filter on the other hand pixels in the other part of the image will have some filter overlap and ensure better feature detection, to avoid this issue we can add a layer around the image with 0 pixel value and increase the possibility of feature extraction in the corners also. This padding will also help us to keep the size of the image same even after the convolution operation.
In below you can see an example of padding:

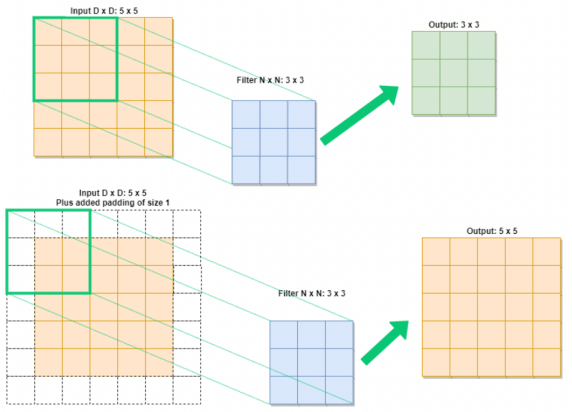

Figure 2: Padding

Also we have stride, the step size of the kernel when sliding through the input data. It can be used for down-sampling the input. See an example of stride in the process of convolution:

Figure 3: Stride

2 Pooling

Pooling is the process of extracting the features from the image output of a convolution layer. This will also follow the same process of sliding over the image with a specified pool size and kernel size.
Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, tiling sizes such as 2×2 are commonly used. Global pooling acts on all the neurons of the feature map.
There are two types of pooling is available:

- Max Pooling
- Average Pooling

Max Pooling is being used widely and it will just keep the highest number in the pool and discard the rest. By getting the highest value in each pool we will be

getting the significant features of the image, the lower values are not the features at all or not significant features to be able to use in the model. Pooling will also have padding and strides and reduces the image size.
Here is the picture of pooling:

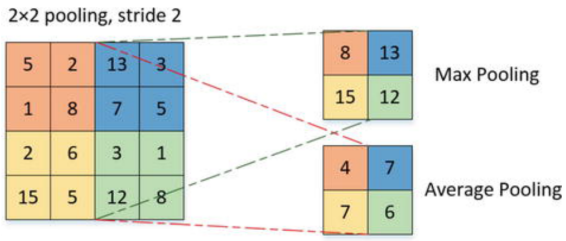

Figure 4: Pooling

So after convolution and pooling we will be having an image which is small and having all the features of the original image. Also the number of weight parameters in the network is very less because each pixel in the resultant image is connected to the pixels in the part of the original image(size of the pool) and not all the pixels of the original image. This will improve the model performance and accuracy.
the effect of the kernel is important too. We learned in the last project there are some filters like DoG that extract light/dark points in the darker/lighter background and some filters like Gabor which extract lines with different orientations and sizes.

3 Spiking Neural Network With Convolution and Pooling Connections

First look at our original image:

3.1 Ex1

Figure 5: Original Image

In the first step we choose a kernel for the process of convolution. We chose DoG filter with these set of parameters:

- kernel size = 7
- $\sigma_1 = 2$
- $\sigma_2 = 4$

Now look at the filter and the convolved image obtained by this DoG filter:

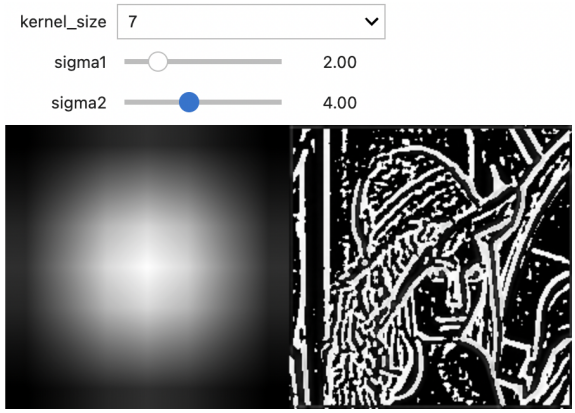

Figure 6: Filter and Convolved Image

As you see the points are extracted very well. Now we encode this convolved image using TTFS encoding with time window of 15. See the result:

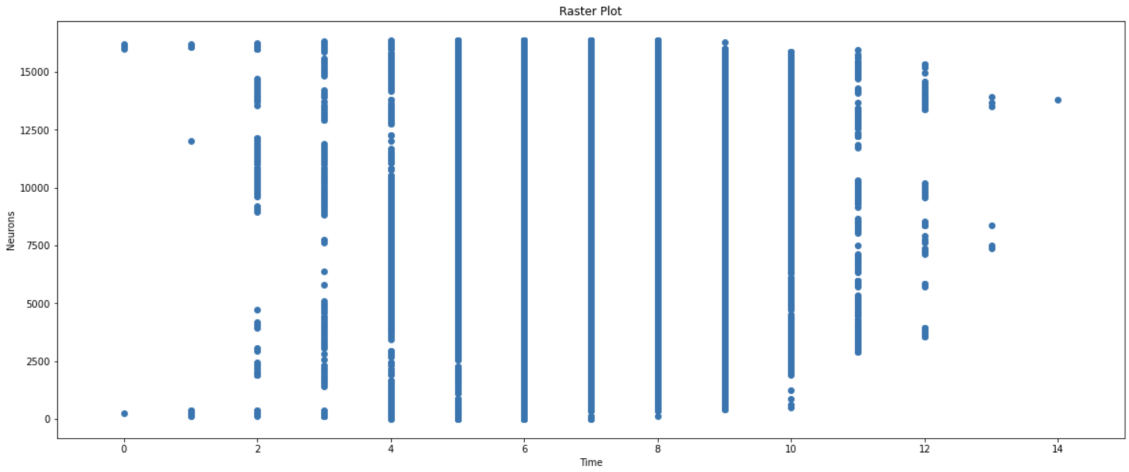

Figure 7: TTFS encoding of Convolved Image

Now see the features come out in every iteration of convolution:

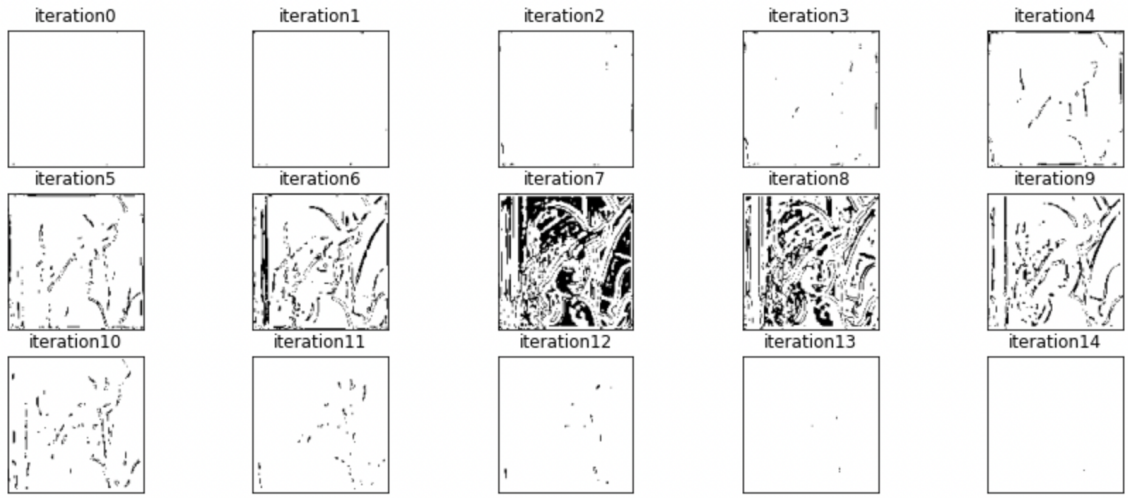

Figure 8: Features in every iteration of TTFS encoding for convolution

Now it's time to add pooling connection to our SNN and see the result. The point is when we are engaging with spiking neural networks, Pooling will have different meaning. Here we are performing local max operation by simply propagating the first spike emitted by a given group of pre-synaptic in the last layer. It means when we convolve our filter on the image, in every iteration of the simulation, neurons in the next layer will have a receptive field on the previous layer and if there is a spike in that receptive field, we can transmit that spike in to the next layer by pooling connection and also that neuron will spike.
So we apply the pooling connection and we will see the spikes in the second layer (our model has two layers, input of the first one is the TTFS encoding of the convolved image obtained by our DoG filter and there is a pooling connection between the first and the second layer, so the output spikes of the second layer is the activity of adding pooling into the model) :

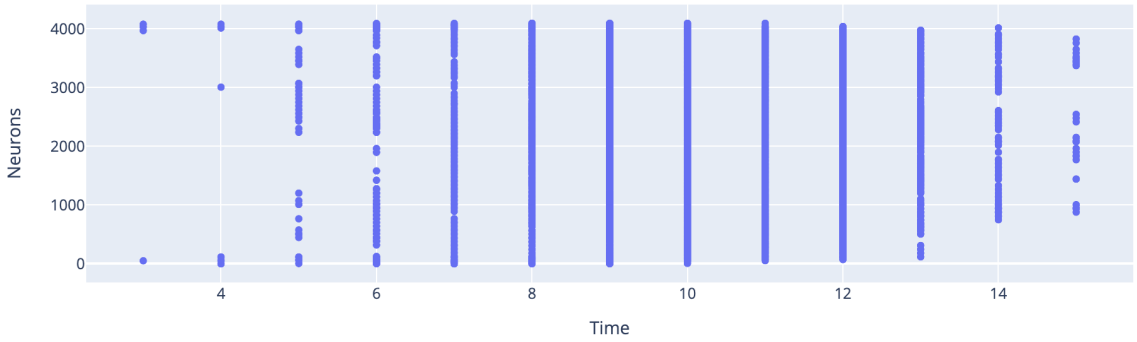

Figure 9: spikes produced by the activity of pooling layer

Now heck the feature extracted in every iteration of the pooling layer:

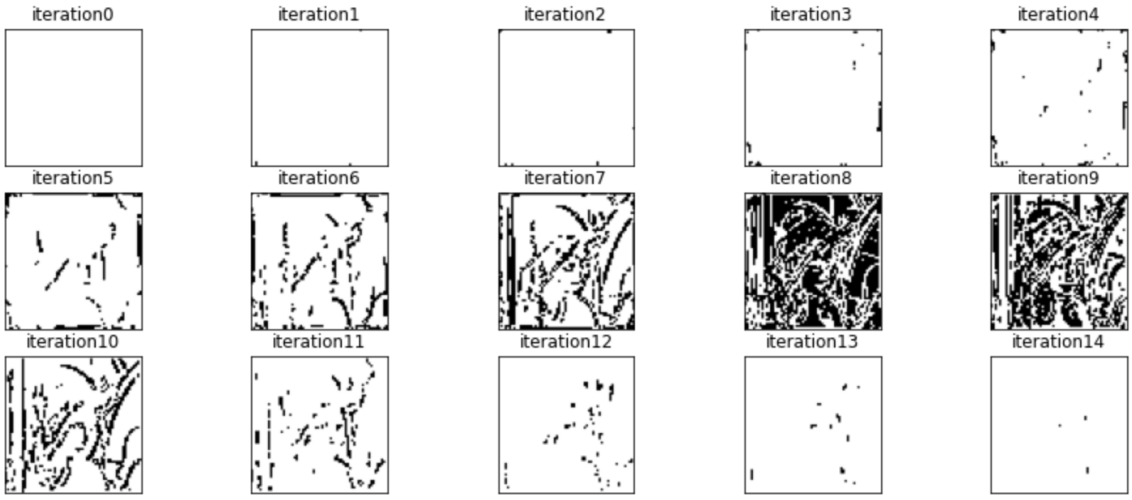

Figure 10: Features in every iteration of pooling layer

As you can see the feature almost remained the same, also the resolution decreased and this is good for computation and complexity. Besides, pooling provides the ability to learn invariant features and also acts as a regularizer to further reduce the problem of overfitting.
Here the pooling size was (2, 2) and the pooling stride was 2. so if in every 2x2 receptive field of the last layer there is a spike, the neuron in the second layer would spike too. in figure 9, neurons start firing at the iteration 3 and if we increase the pooling size the receptive field will be bigger and more neurons will spike. check the raster plot below for better understanding:

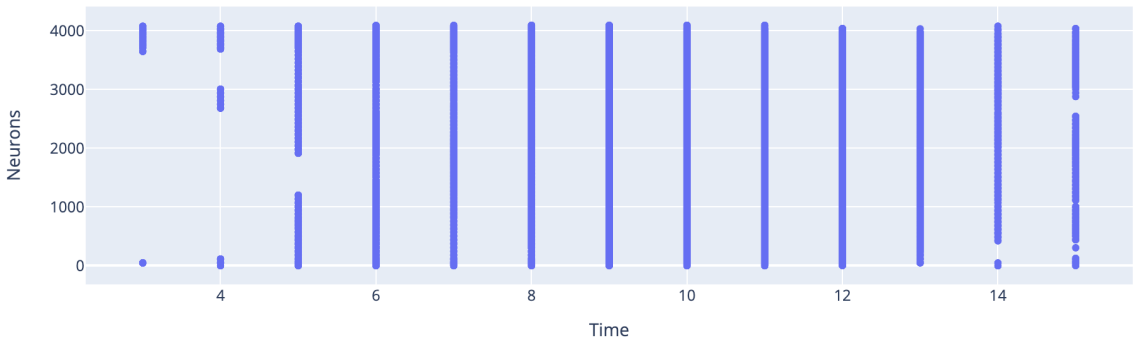

Figure 11: spikes produced by the activity of pooling layer with bigger pooling size

Here the pooling size is 12 and if we compare this figure with figure 9, we'll see we have more spikes in every iteration. Now check the features extracted in every iteration:

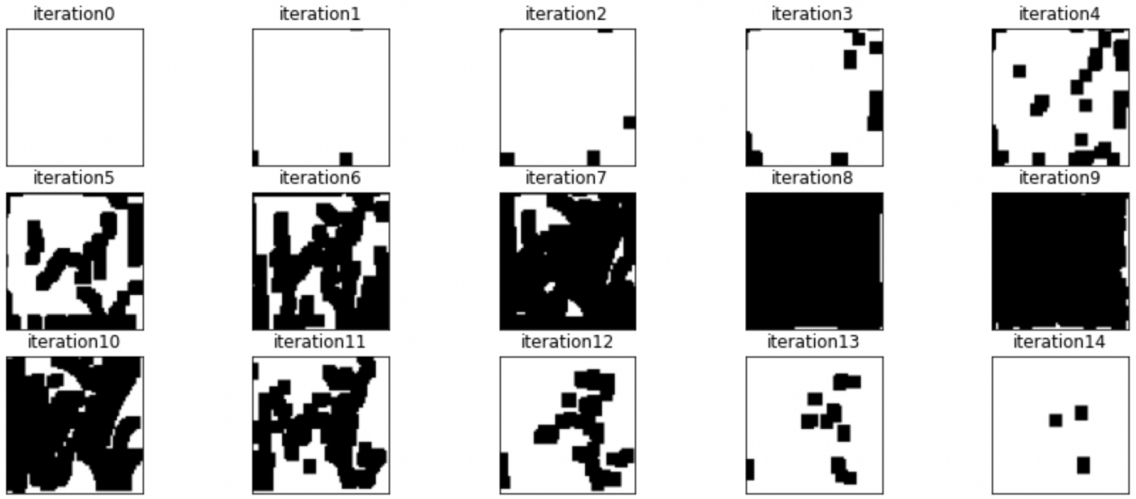

Figure 12: Features in every iteration of pooling layer

OOPS!
As you can see the features are not good for the model, that's because the pooling size of our filter is too large and the receptive is large, so we cannot extract features with details (like texture) from the concolved image. Therefore we should be careful about choosing the pooling size.
Now what will happen if we increase the stride of our pooling?
As we said before, stride is the number of pixels shifts over the input matrix of our image. The stride can reduce the resolution of the output. Now see the result of spiked when the pooling size is (2,2) and the pooling stride is 4:

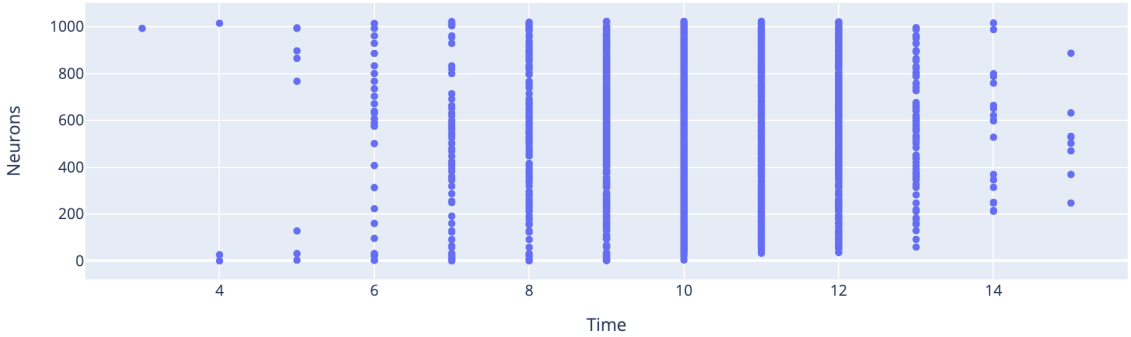

Figure 13: pikes produced by the activity of pooling layer with bigger pooling stride

The number of neurons is lower (because the resolution is decreased) and also we have less spikes in the iterations. See the features in every iteration now:

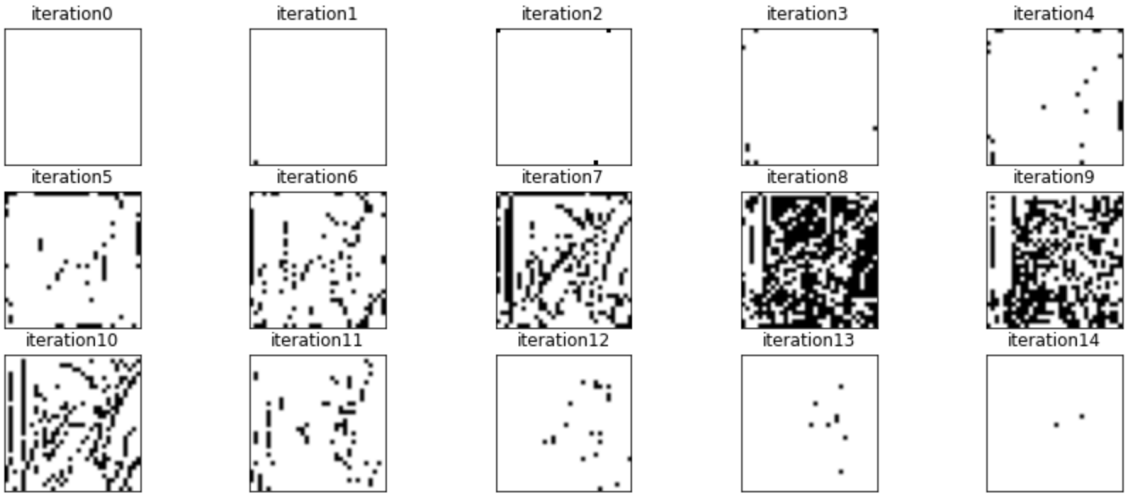

Figure 14: Features in every iteration of pooling layer

Well, the features are not good as figure 10. Because we down-sampled with a large parameter and we should be careful about choosing the pooling stride. It shouldn't be too large.

3.2 Ex2

In this part we chhose the parameter of our DoG filter like this:

- kernel size = 13
- $\sigma_1 = 2$
- $\sigma_2 = 5$

Now look at the filter and the convolved image obtained by this DoG filter:

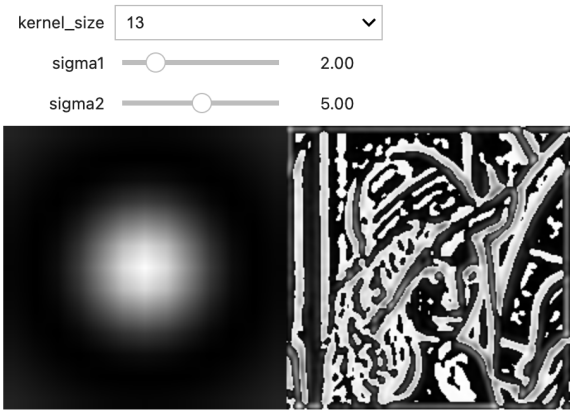

Figure 15: Filter and Convolved Image

With this filter we cannot extract details because the kernel size is large compared to Ex 1, therefore receptive field is larger. Larger kernels can capture more complex features but also increase the number of parameters. In this case the TTFS encoding of convolved image is:

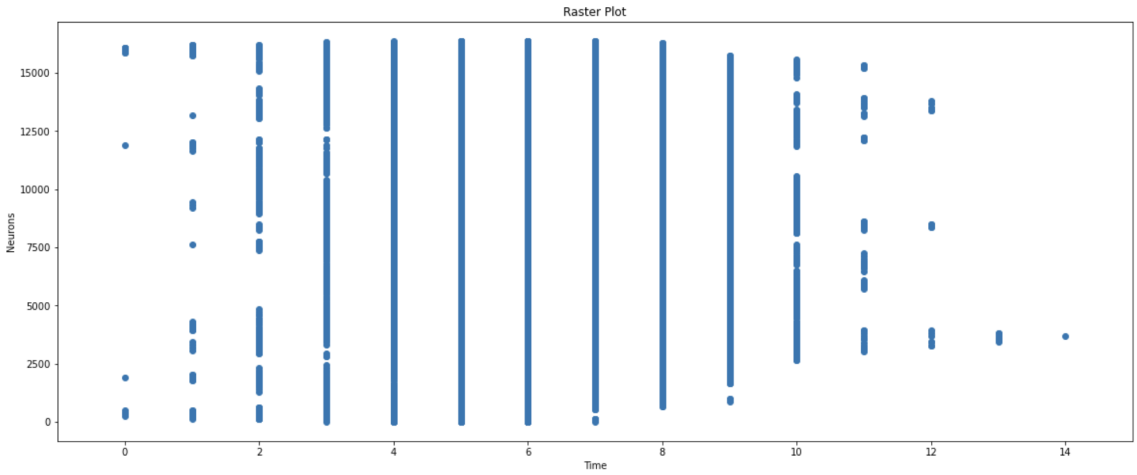

Figure 16: TTFS encoding of Convolved Image

Now see the features come out in every iteration of convolution:

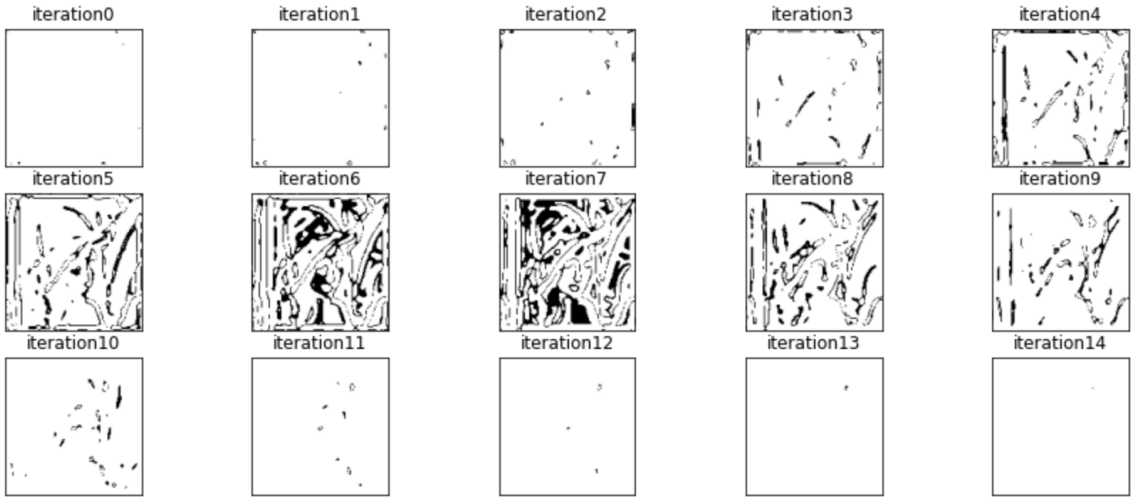

Figure 17: Features in every iteration of TTFS encoding for convolution

It's time to add pooling connection. First see the raster plot:

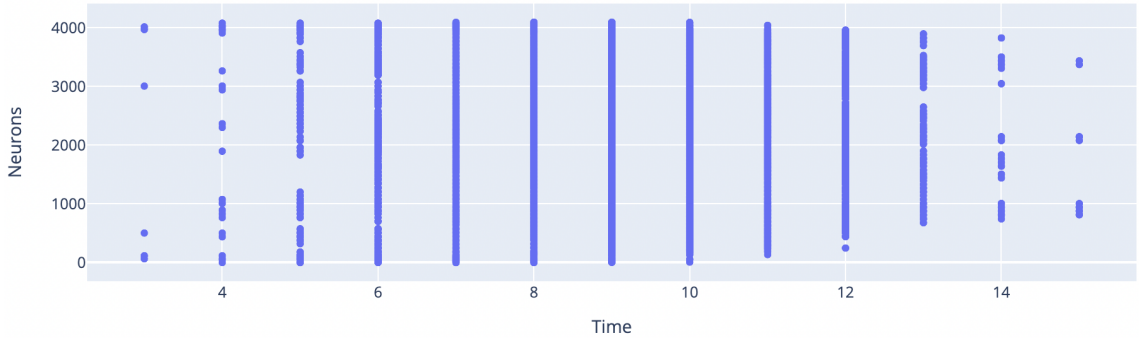

Figure 18: spikes produced by the activity of pooling layer

Here the kernel size of the pooling is (2,2) and the pooling stride is 2. At the end look at the features extracted from the pooling layer in each iteration:

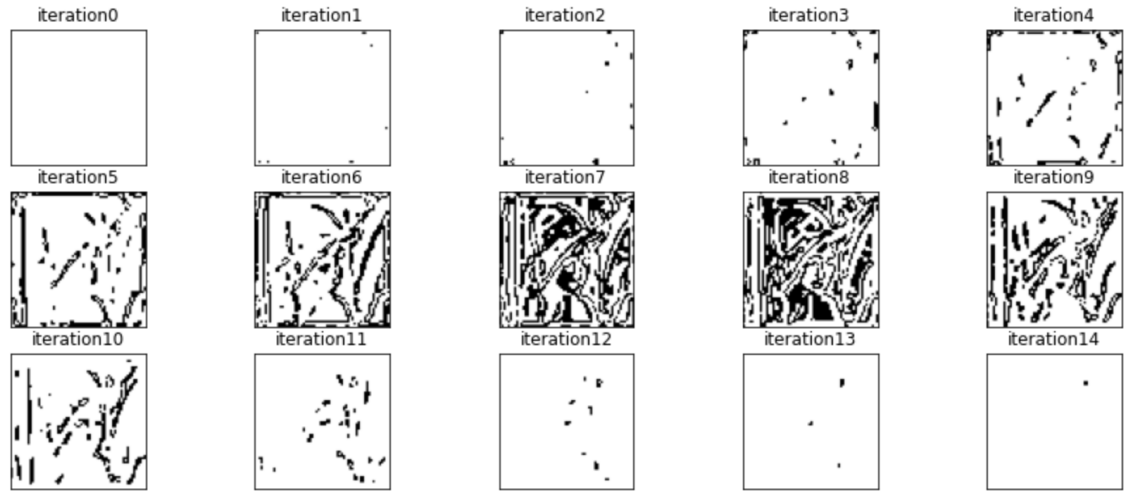

Figure 19: Features in every iteration of pooling layer

As you can see in the figure 18, we have spikes in every iteration.

4 References

[1] Convolution https://en.wikipedia.org/wiki/Convolution

[2] Connectivity Schemes Operation https://maktabkhooneh.org/course/%D8%A2%D9%85%D9%88%D8%B2%D8%B4-%D8%B1%D8%A7%DB%8C%DA%AF%D8%A7%D9%86-%D8%B9%D9%84%D9%88%D9%85-%D8%A7%D8%B9%D8%B5%D8%A7%D8%A8-%D9%85%D8%AD%D8%A7%D8%B3%D8%A8%D8%A7%D8%AA%DB%8C-mk744/%D9%81%D8%B5%D9%84-%D8%A7%D9%88%D9%84-%D8%B9%D9%84%D9%88%D9%85-%D8%A7%D8%B9%D8%B5%D8%A7%D8%A8-%D9%85%D8%AD%D8%A7%D8%B3%D8%A8%D8%A7%D8%AA%DB%8C-ch2077/%D9%88%DB%8C%D8%AF%DB%8C%D9%88-%D8%A7%D9%84%DA%AF%D9%88%D9%87%D8%A7%DB%8C-%D8%A7%D8%B1%D8%AA%D8%A8%D8%A7%D8%B7%DB%8C-%D8%A8%DB%8C%D9%86-%D9%86%D9%88%D8%B1%D9%88%D9%86-%D9%86%D9%88%D8%A7%D8%AD%DB%8C-%D9%85%D8%AE%D8%AA%D9%84%D9%81-%D9%85%D8%BA%D8%B2/

[3] Convolutional neural network https://en.wikipedia.org/wiki/Convolutional_neural_network