Filters and Feature Extraction

Amir Faridi - 610300087

Department of Computer Science, University of Tehran

Computational Neuroscience

1. Introduction

In this project, I will explore the use of the Difference of Gaussian (DoG) and Gabor filters for extracting specific features from images. Additionally, I will encode some of the results using Time-to-First-Spike and Poisson encoding methods. Finally, I will attempt to train a simple neural network to learn certain characteristics of the given dataset.

2. Difference of Gaussian (DoG) and Gabor

Starting with the DoG filter, I will analyze the impact of each parameter—σ_1, σ_2, and *size*—on the filter's output. Throughout this project, the *stride* parameter will be set to 1, and padding will be applied to maintain the output image's size. Additionally, zero-mean normalization will be applied to the image to make the output more interpretable. The following two images will be used to analyze the filters and encodings.

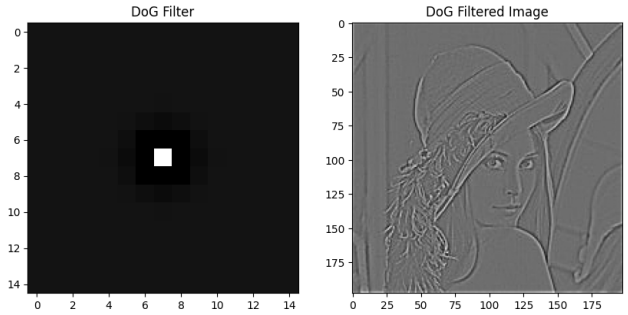

Figure 1. *Parameters:* $\sigma_1 = 0.5$, $\sigma_2 = 1$, $size = 15$

This figure illustrates a DoG filter with parameters $\sigma_1 = 1$, $\sigma_2 = 0.5$, and $size = 15$, creating an on-center filter. As anticipated from an on-center filter, the resulting image highlights the edges of the input image. Notably, the white areas of the eyes are visible, demonstrating the on-center effect of the filter. Whereas the next image shows the off-center version of this filter.

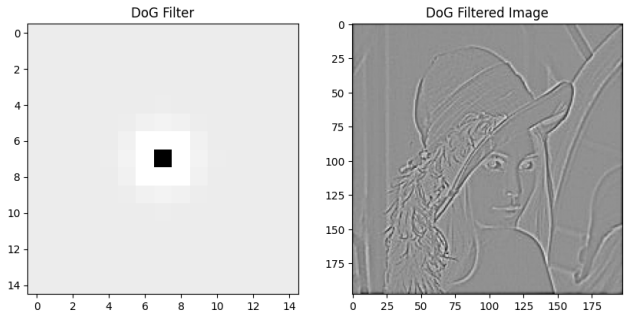

Figure 2. *Parameters:* $\sigma_1 = 1$, $\sigma_2 = 0.5$, $size = 15$

As expected, the off-center version of the previous filter highlights the image edges in an off-center manner, which is evident from the appearance of the eyes. Unlike the previous result, the black areas of the eyes are emphasized, showcasing the off-center effect of the filter.

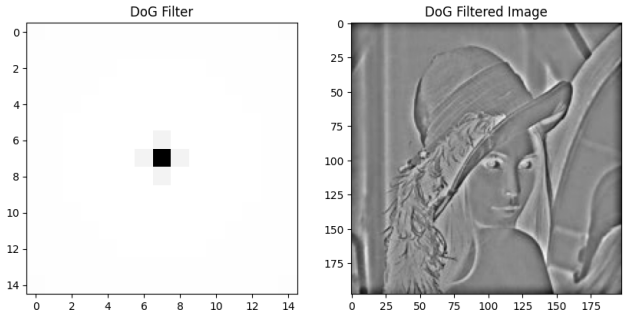

Figure 3. *Parameters:* $\sigma_1 = 5$, $\sigma_2 = 0.55$, $size = 15$

Increasing the value of σ_2 to 5 while keeping σ_1 at 0.5 results in a filter that still highlights the edges of the image in an off-center manner. However, the surrounding area where the filter is applied appears significantly whiter than in the previous image. This effect of the parameter adjustment is also evident in the filter itself.

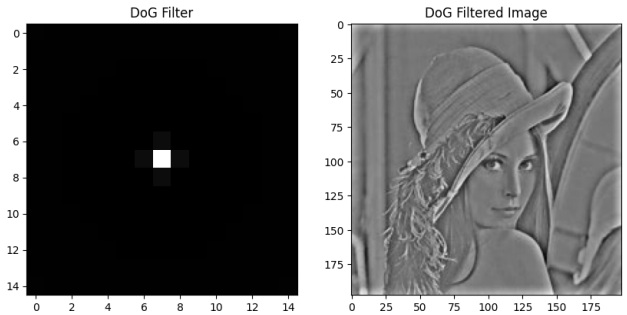

Figure 4. *Parameters:* $\sigma_1 = 0.5$, $\sigma_2 = 5$, $size = 15$

Same argument as the previous image, with the difference that the filter is now on-center. The effect of σ_1 being much higher than σ_2 is evident in the filter's appearance, as the surrounding area is significantly darker than the surrounding of the figure 1.

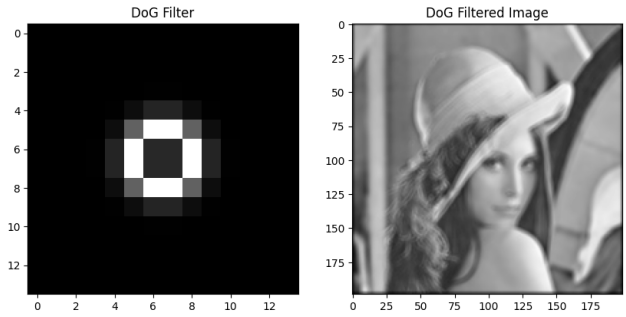

Figure 5. *Parameters:* $\sigma_1 = 1, \sigma_2 = 0.5, size = 14$

When the filter's *size* is an even number with the given σ parameters, it does not distinctly correspond to an off-center or on-center filter; instead, it resembles a Gaussian blur filter. This occurs because the filter's center and a small surrounding area function like an averaging filter, as evident in both the output image and the filter itself.

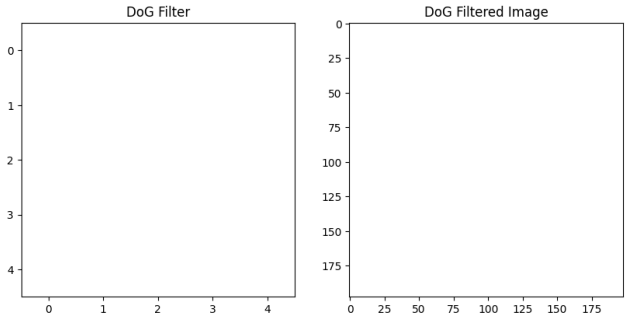

Figure 6. *Parameters:* $\sigma_1 = 1, \sigma_2 = 1, size = 5$

This filter does not an interesting set of parameters. When both σ_1 and σ_2 are equal, there is no difference in the gaussian equations and they cancel each other out, resulting in a 0 filter and a 0 image.

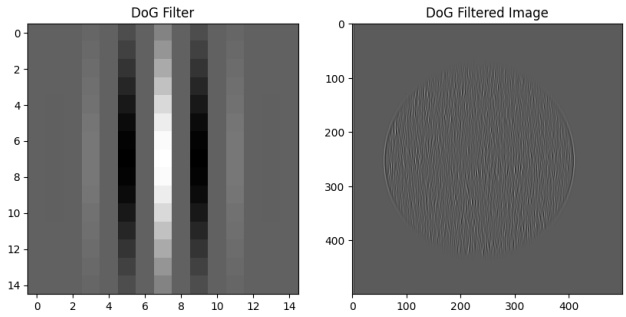

Figure 7. *Parameters:* $\lambda = 4, \theta = \pi/2, \sigma = 2, \gamma = 0.5, size = 15$

Moving onto the Gabor filter, I'll use a circle image with noises in the middle with different orientations which will capture the effect of each gabor filter with different parameters.

With this set of parameters, the filter is oriented at $\pi/2$, which is expected to target the vertical edges of the image. This is confirmed by the result, where vertical edges are more pronounced. As we move towards the top and bottom of the circle, the edges become less visible since they are not vertical. Conversely, the edges on the left and right sides of the circle are more prominent, as they align with the vertical orientation of the filter. Same argument is valid for the noises in the middle of the shape.

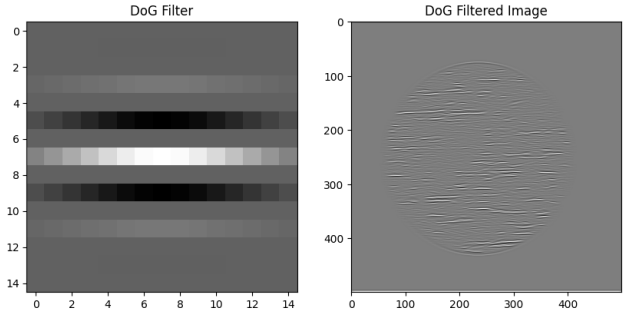

Figure 8. *Parameters:* $\lambda = 4, \theta = \pi, \sigma = 2, \gamma = 0.5, size = 15$

Same argument as the previous figure is valid except the orientation of this filter is π which is horizontal. The filter is expected to target the horizontal edges of the image, which is confirmed by the result. The horizontal edges of the circle are more pronounced, while the vertical edges are less visible. The noises in the middle of the circle are also affected by the filter, with the horizontal edges being more prominent than the vertical ones.

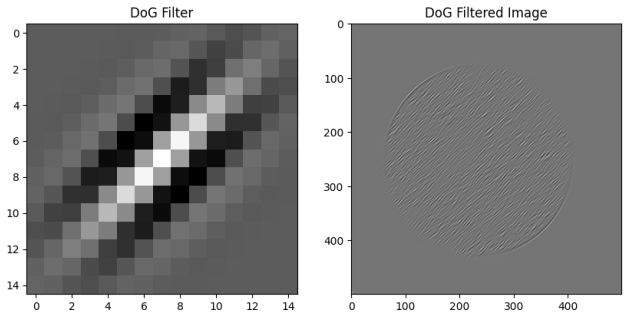

Figure 9. *Parameters:* $\lambda = 4, \theta = \pi/4, \sigma = 2, \gamma = 0.5, size = 15$

Same argument as above for orientation equals to $\pi/4$.

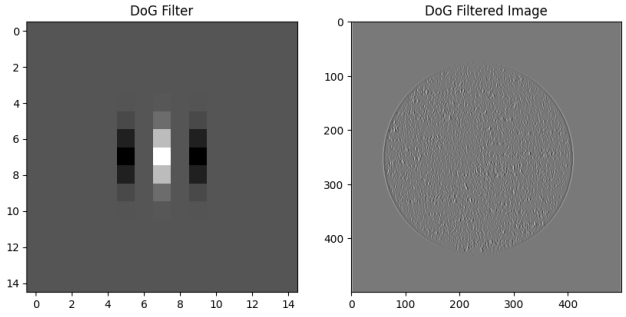

Figure 10. *Parameters:* $\lambda = 4, \theta = \pi/2, \sigma = 0.5, \gamma = 0.5, size = 15$

The parameter σ represents the spatial extent. It controls the spread of the Gaussian envelope within the filter. Larger values of σ result in broader receptive fields, capturing lower spatial frequencies. Conversely, smaller σ values focus on finer details and higher spatial frequencies. The relationship between σ and the filter's bandwidth is inversely proportional. By decreasing the value of σ, we could expect the filter to narrow the Gaussian envelope which results in a sharper filter

with a higher spatial frequency response and become more sensitive to fine details and edges. This can be seen from the result image and the filter itself.

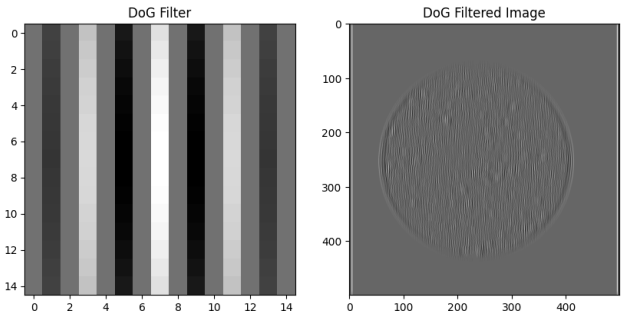

Figure 11. *Parameters:* $\lambda = 4, \theta = \pi/2, \sigma = 5, \gamma = 0.5, size = 15$

And by increasing the σ, the filter becomes more sensitive to lower spatial frequencies and captures broader details. This can be seen from the result image and the filter itself as well.

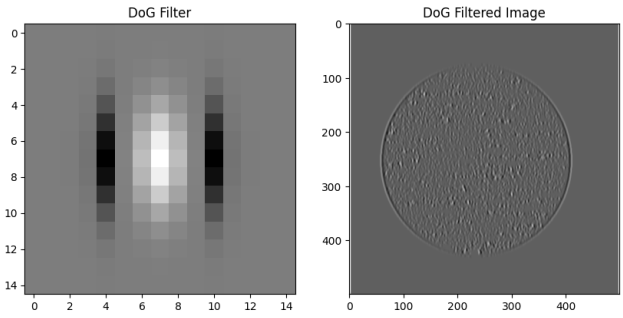

Figure 12. *Parameters:* $\lambda = 10, \theta = \pi/2, \sigma = 1, \gamma = 0.5, size = 15$

The parameter λ, which is the wavelength of the filter, determines the size of the filter's receptive field. The wavelength determines the width of the strips in the Gabor function. In other words, λ influences the size of the patterns the filter can capture.

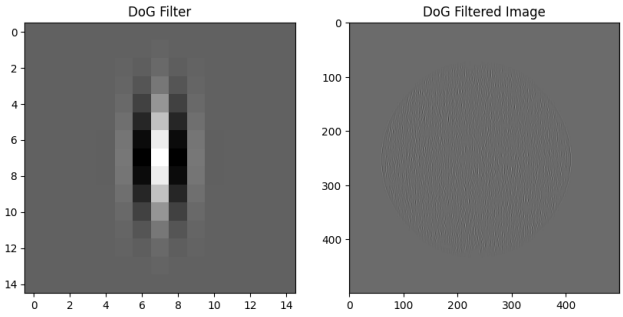

Figure 13. *Parameters:* $\lambda = 2, \theta = \pi/2, \sigma = 1, \gamma = 0.5, size = 15$

Also by decreasing λ, the filter becomes more sensitive to higher spatial frequencies and captures finer details; which can be seen from the figure.

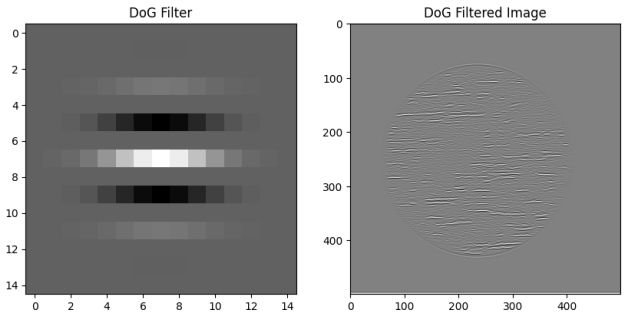

Figure 14. *Parameters:* $\lambda = 4, \theta = \pi, \sigma = 1, \gamma = 1, size = 15$

γ captures the aspect ratio of the filter. It specifies the ellipticity of the support of the Gabor function. When $\gamma = 1$, the support is circular, while $\gamma < 1$ results in an elliptical support.

DoG for RGB images

The main idea for implementing the DoG filter for RGB images is to apply the filter to each channel separately and then combine the results to form the final output. This approach allows the filter to capture the edges and features of each color channel independently. The following images illustrate the results of applying the DoG filter to an RGB image.

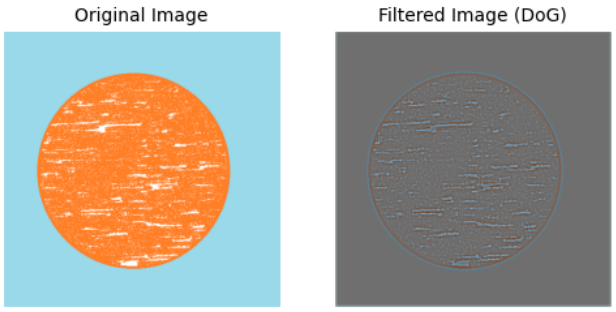

Figure 15. *Parameters:* $\sigma_1 = 1, \sigma_2 = 3\ size = 9$

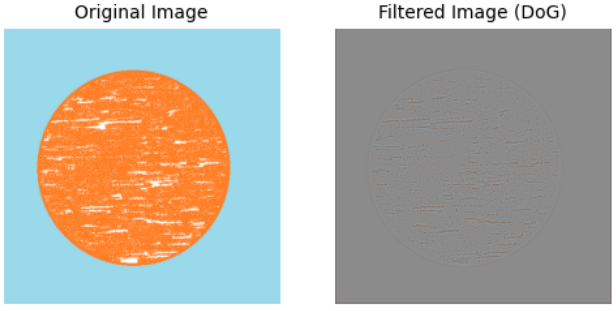

Figure 16. *Parameters:* $\sigma_1 = 1, \sigma_2 = 0.5\ size = 15$

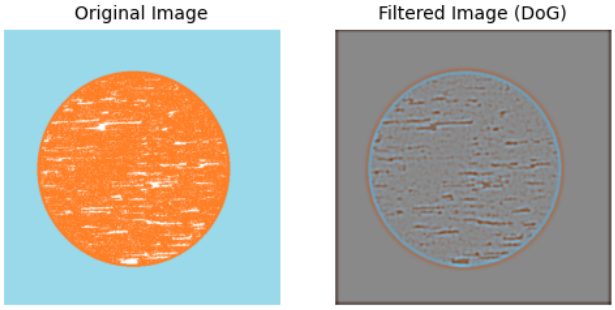

Figure 17. *Parameters:* $\sigma_1 = 4$, $\sigma_2 = 3$ *size* $= 15$

Figure 18. *Parameters:* $\sigma_1 = 1$, $\sigma_2 = 0.6$ *size* $= 15$

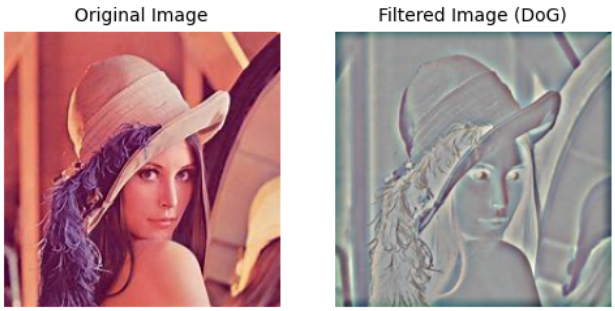

Figure 19. *Parameters:* $\sigma_1 = 1$, $\sigma_2 = 0.6$ *size* $= 15$

3. Encodings

In this section, I will encode some of the results above using Time-to-First-Spike and Poisson methods to illustrate the difference between them and that the how parameters and time-window in the encodings can affect the results.

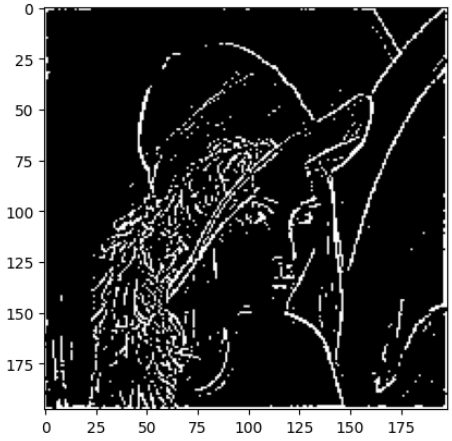

Figure 20. *Parameters:* $\sigma_1 = 0.5$, $\sigma_2 = 1$, *size* $= 15$

This is the sum of the Intensity-to-Latency encoding on the time dimension, where the result is the same image, only the edges are captured.

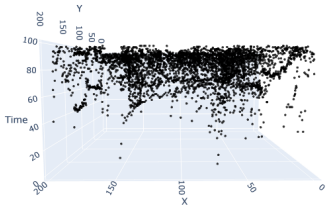

Figure 21. *This is the same encoding as the above figure, from another perspective.*

Here we can see the time dimension of the encoding, where the most density increases as the value of time increases. And if we look at the 3D graph from above, we can see the encoded image.

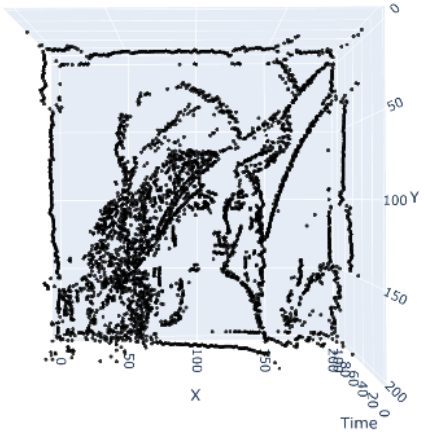

Figure 22. $time_window = 100$, $threshold = 0.1$, $sparsity = 0.1$

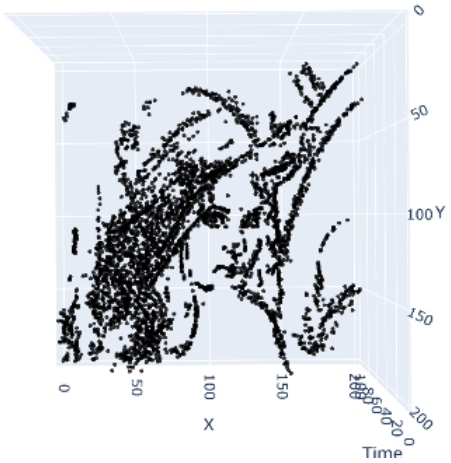

Figure 23. *Filtered image in figure 2 encoded with Intensity-to-Latency encoding.*

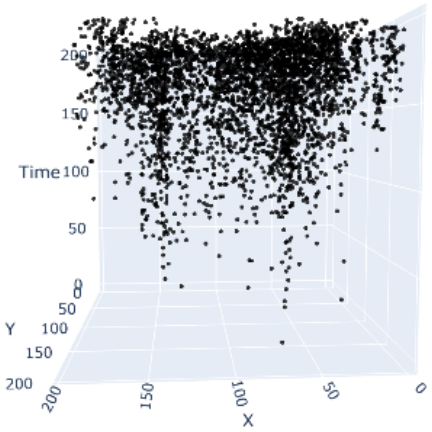

Figure 24. *Encoded image of figure 2 with a time window of 200.*

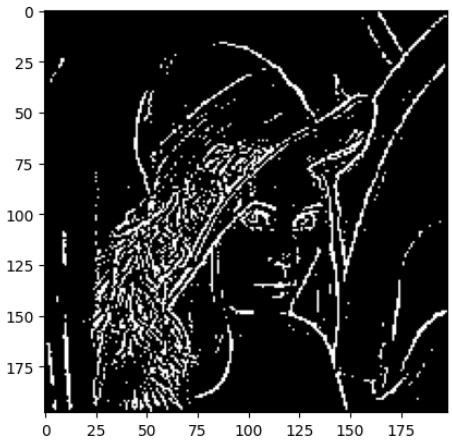

Figure 25. *Same image as above, summed over the time dimension.*

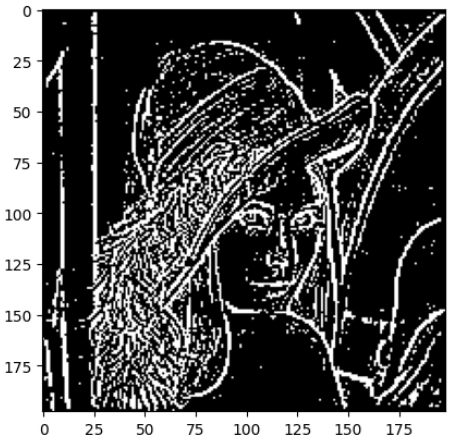

Figure 26. *Encoded image of figure 2 with a time window of 100 and a sparsity of 0.2.*

increasing the sparsity of the encoding results in a more sparse representation of the image, where the number of spikes are increased.

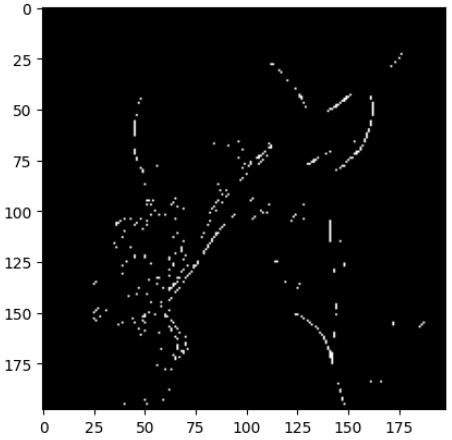

Figure 27. *Encoded image of figure 2 with a time window of 100 and a sparsity of 0.01.*

But by decreasing the sparsity, the encoding captures the most dense part of the filtered image, which is the most visible and dence edges.

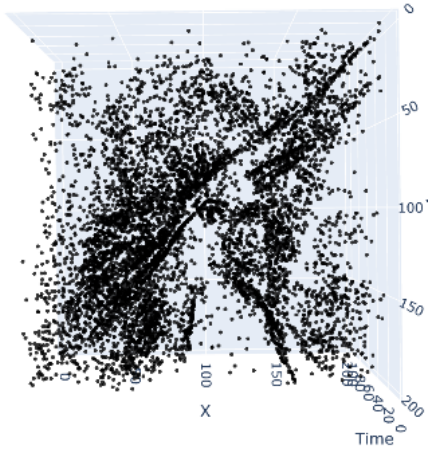

Figure 28. *Filtered image of figure 2 encoded with Poisson encoding.* $ratio = 0.001$

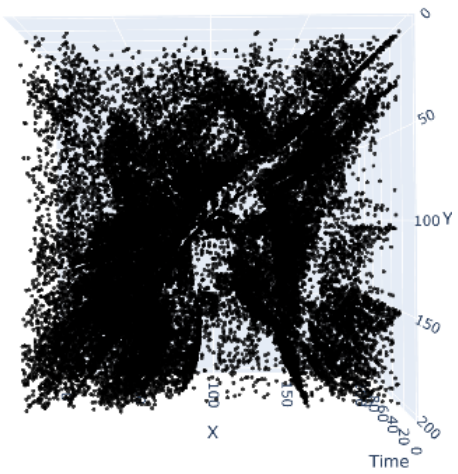

Figure 29. *Filtered image of figure 2 encoded with Poisson encoding.* $ratio = 0.005$

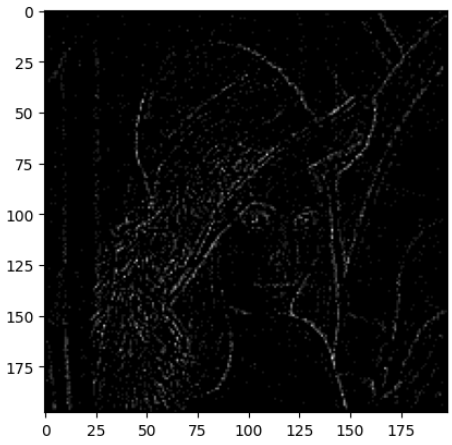

Figure 30. *Filtered image of figure 2 encoded with Poisson encoding and summed over the time dimension.* $ratio = 0.001$

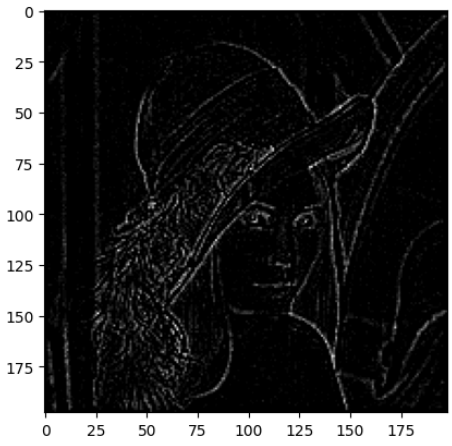

Figure 31. *Filtered image of figure 2 encoded with Poisson encoding and summed over the time dimension.* $ratio = 0.005$

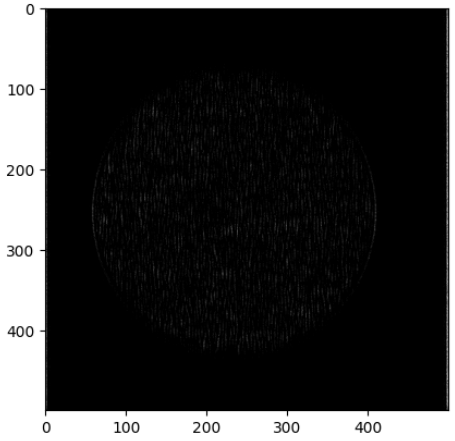

Figure 32. *Filtered image of figure 7 encoded with Poisson encoding and summed over the time dimension.* $ratio = 0.1$

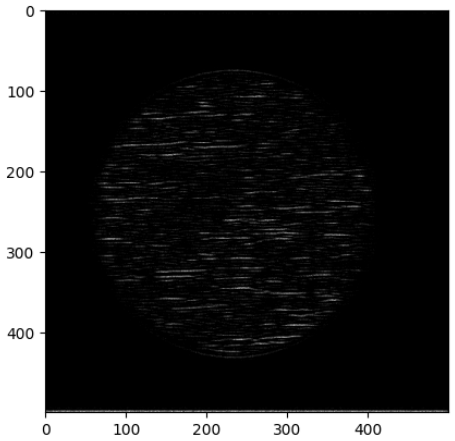

Figure 33. *Filtered image of figure 8 encoded with Poisson encoding and summed over the time dimension.* $ratio = 0.1$

There are some factors in this encoding. One is the randomness, where the coded image will have some noise which is the result of the nature of this method. Eyes of humans, see and encode the surroundings in a similar way, theres is always noises in our vision and it's not perfect. So it seems reasonable for this method to be more accurate in the sense of human vision.

Another factor is the ratio, which is the rate of the spikes in the encoding. By increasing the ratio, the number of spikes in the encoding will increase, which will result in a more dense representation of the image. This can be seen from the figures above. Also decreasing it, the coding will capture the more important parts of the filtered image which are the most visible edges. Also it it obvious that by increasing the size of the image, the count of the spikes in the coded image will be more for the same image and ratio.

4. SNN

In this section I will try to create and analyze the behavior of a simple Spiking Neural Network with two layers. The first layer will be the input layer with size of the input images which is set to 100×100 throughout the project. The input images will be filtered using the DoG filter, encoded with one of the Intensity-to-Latency or Poisson encoding, and passed to the second layer using a convolutional connection between two layer.

In the second layer, there will be f feature pages which are expected to learn the frequent features of the input images. Input images are selected from the CelebA dataset randomly, and resized to 100×100 to be fed into the network. The connections of the network will be traied using the STDP rule. Also in the parameter tables, I will mentions the details of the model and the mechanisms that are used for that specific experiment.

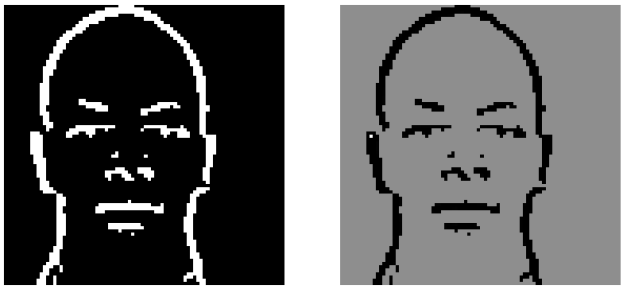

Figure 34

Table 1. *Parameters*

encoding	TTFS
time window	40
filter size	15
sigma 1	1.5
sigma 2	1
feature count	1
a plus	1
a minus	1
init weight	0.5
kwta	1
iterations	500
feature size	86

Here the image on the left is the encoded image and the other one is the weights of the synapses between the input layer and the second one which has only one feature page. The size of the page is set to the size of the input image which is 86×86. And since there are no randomness, noise or another input image, the weights are perfectly separated in a way that constructs the face will all of its details.

Figure 35

Table 2. *Parameters*

encoding	poisson
poisson ratio	2
time window	45
filter size	15
sigma 1	1.5
sigma 2	1
feature count	1
a plus	1
a minus	1
init weight	0.5
kwta	1
iterations	500
feature size	86

Here I used poisson encoding instead of TTFS and used the exact same parameters as the previous experiment. The encoded image has some noise and randomness, but it makes the image look more realistic as well as the extracted feature.

Now if the number of features is increased, since the size of the pages are as large as the input images, all features will look the same since there is only one thing to learn which is the only input image. This is illustrated in the following figures.

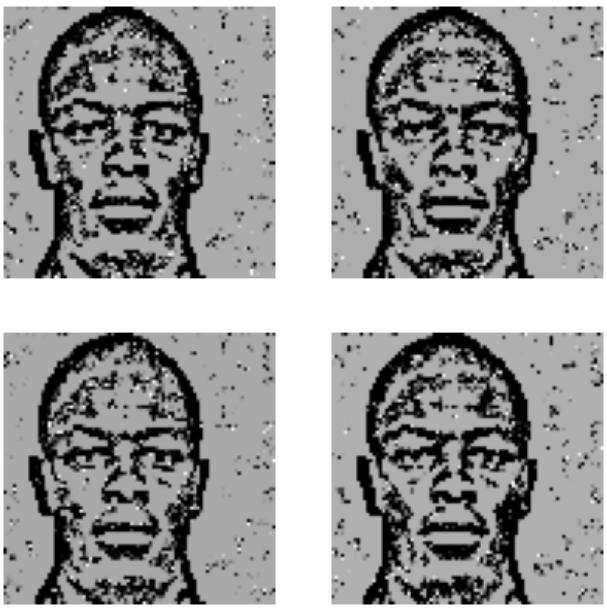

Figure 36

Table 3. *Parameters*

encoding	poisson
poisson ratio	2
time window	45
filter size	15
sigma 1	1.5
sigma 2	1
feature count	4
a plus	1
a minus	1
init weight	0.5
kwta	1
kwta dimension	0
lateral range	40
iterations	500
feature size	86

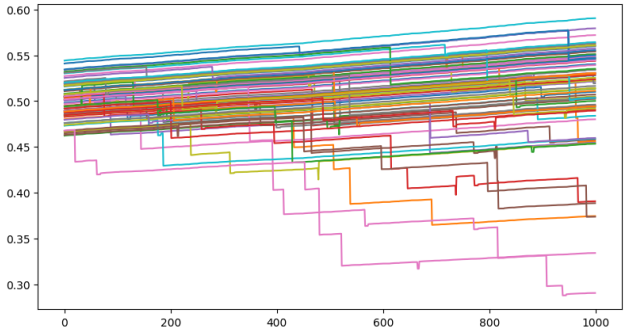

Figure 39. *Weights of the synapses*

As mentioned, all 4 features look the same. There are some extra parameter and mechanisms are added since the number of features are increased. The kwta mechanism will impact only to the index of neurons in the different feature pages. Also the lateral range is the range of the lateral connections of a neuron. Each neuron in the second layer will inhibit 40 neurons in its surroundings.

Table 4. *Parameters*

encoding	poisson
poisson ratio	0.2
time window	50
filter size	15
sigma 1	1.5
sigma 2	1
feature count	3
a plus	0.05
a minus	0.05
init weight	normal(0.5, 0.02)
kwta	1
kwta dimension	0
lateral range	20
iterations	1000
feature size	86

Figure 37. *Input images*

In this experiment, the number of input images is increased to 6. Each image is presented for 50 iterations and after all images are presented, it's circle back to the first image for 1000 iterations. The learned features are like the average of the input faces, but doesn't look like any of them. A perfect nose can be seen in the results despite the fact that none of the input images have a perfect nose.

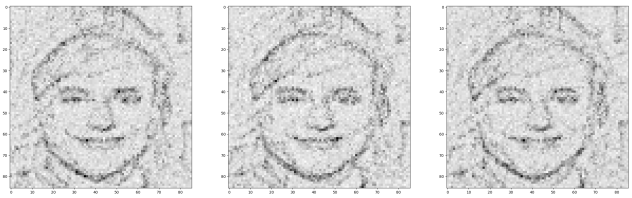

Figure 38. *Result features (weights)*

Figure 40. *Result features (weights)*

Table 5. *Parameters*

encoding	poisson
poisson ratio	0.2
time window	50
filter size	15
sigma 1	1.5
sigma 2	1
feature count	3
a plus	0.5
a minus	0.5
init weight	normal(0.5, 0.02)
kwta	1
kwta dimension	0
lateral range	20
iterations	1000
feature size	86

Table 6. *Parameters*

encoding	TTFS
poisson ratio	0.2
time window	50
filter size	15
sigma 1	1.5
sigma 2	1
feature count	3
a plus	0.05
a minus	0.05
init weight	normal(0.5, 0.02)
kwta	1
kwta dimension	0
lateral range	20
iterations	1000
feature size	86

The result of increasing the learning rates is obvious from the result. The parameters in this experiment are the same as previous one except that the learning rates. It made the weights to be more distinguishable and the features to be more clear.

Figure 41. *Input images*

Figure 42. *Result features (weights)*

Switching to the TTFS encoding method does not appear to significantly alter the results.

Figure 43. *Input images*

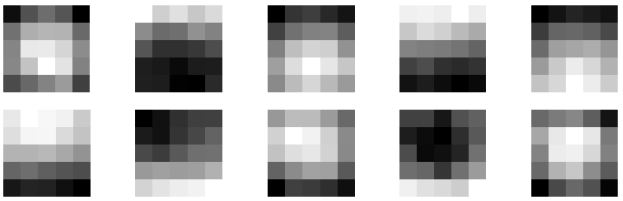

Figure 44. *Result features (weights)*

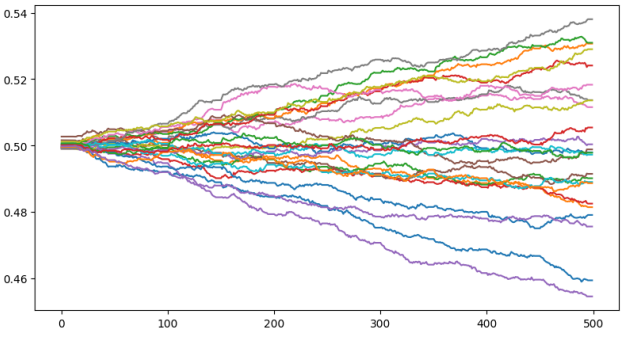

Figure 45. *Weights of the synapses*

Table 7. *Parameters*

encoding	poisson
poisson ratio	2
time window	45
filter size	15
sigma 1	10
sigma 2	1
feature count	10
a plus	0.9
a minus	0.008
init weight	normal(0.5, 0.001)
kwta	1
kwta dimension	0
lateral range	2
iterations	500
feature size	5

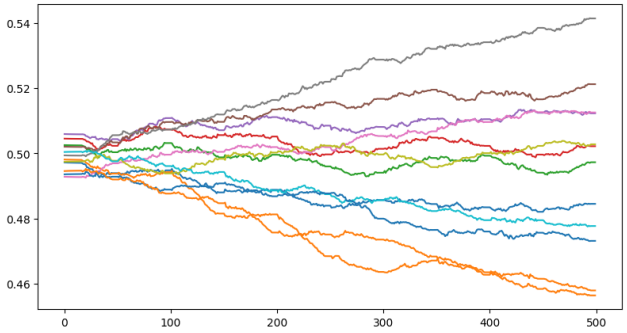

Figure 48. *Weights of the synapses*

Here I set the feature size to be 5. The result features cannot be interpreted as the previous ones, but we can say that they represent some frequent common features in the input images like edges, curves, etc.

Figure 46. *Input images*

Table 8. *Parameters*

encoding	poisson
poisson ratio	2
time window	45
filter size	15
sigma 1	10
sigma 2	1
feature count	10
a plus	1
a minus	0.009
init weight	normal(0.5, 0.005)
kwta	1
kwta dimension	0
lateral range	2
iterations	500
feature size	10

Same as above, extracted 10×10 features from the input images. Again they correspond to different frequent features in the inputs like edge, curves, etc.

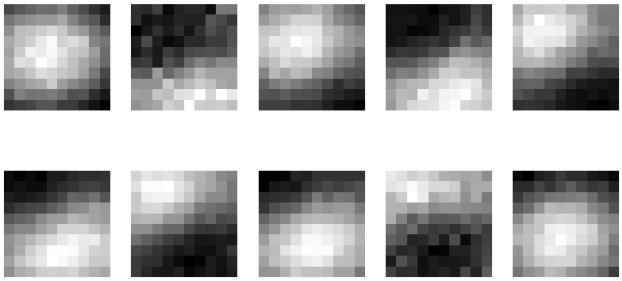

Figure 47. *Result features (weights)*

Figure 49. *Input images*

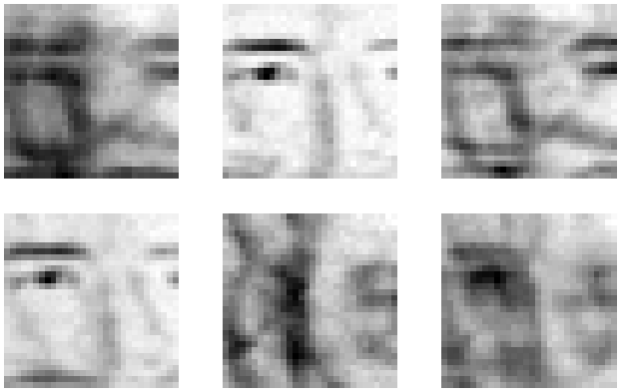

Figure 50. *Result features (weights)*

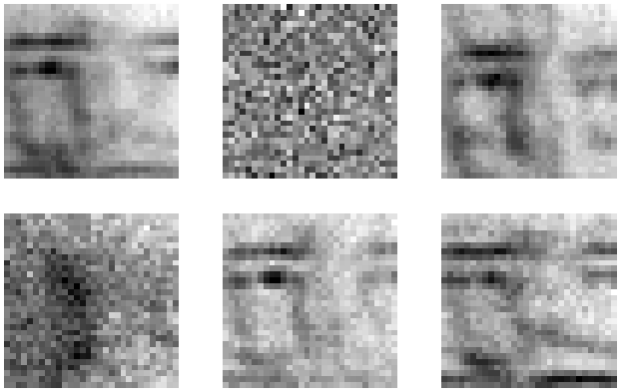

Figure 52. *Result features (weights)*

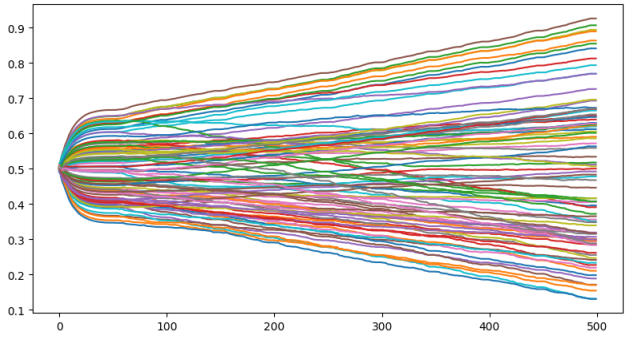

Figure 51. *Weights of the synapses*

Table 9. *Parameters*

encoding	poisson
poisson ratio	5
time window	45
filter size	15
sigma 1	1.5
sigma 2	1
feature count	6
a plus	1
a minus	1
init weight	normal(0.5, 0.005)
kwta	1
kwta dimension	0
lateral range	20
iterations	500
feature size	30

Table 10. *Parameters*

encoding	poisson
poisson ratio	2
time window	45
filter size	15
sigma 1	1.5
sigma 2	1
feature count	6
a plus	1
a minus	1
init weight	normal(0.5, 0.005)
kwta	1
kwta dimension	0
lateral range	20
iterations	500
feature size	30

The size of features in this experiment is set to 30×30. There are some clear features in the result like nose, eyes, and eyebrows.

This is the same as the previous experiment, but the poisson ratio in the encoding parameters are decreased and this resulted in a less clear and more noisy features. In the next experiment I increased the number of input images for the model to learn the average features of a face and not only one perticular face.

Figure 53. *Input images*

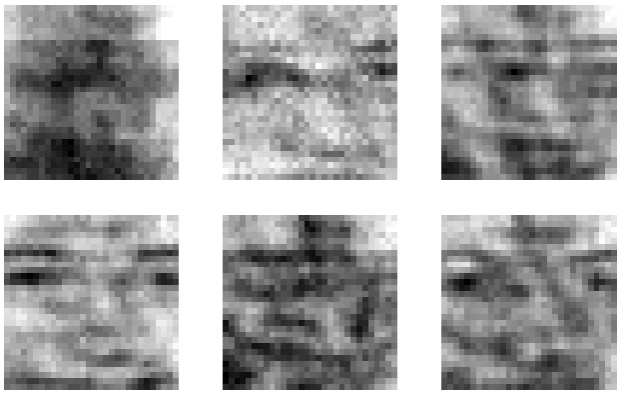

Figure 54. *Result features (weights)*

Table 11. *Parameters*

encoding	poisson
poisson ratio	5
time window	45
filter size	15
sigma 1	1.5
sigma 2	1
feature count	6
a plus	1
a minus	1
init weight	normal(0.5, 0.005)
kwta	1
kwta dimension	0
lateral range	20
iterations	500
feature size	30

The result of this experiment is the average of the input images. The features are not clear and they are more like the average of the input images.

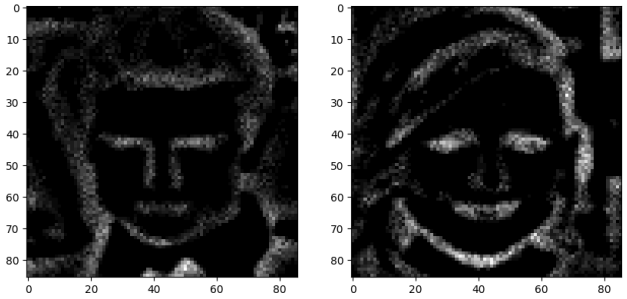

Figure 55. *Input images*

Figure 56. *Result features (weights)*

Table 12. *Parameters*

encoding	poisson
poisson ratio	2
time window	45
filter size	15
sigma 1	10
sigma 2	1
feature count	2
a plus	0.02
a minus	0.01
init weight	normal(0.5, 0.005)
kwta	1
kwta dimension	0
lateral range	80
iterations	1000
feature size	86

Here by tuning the parameters, I tried to make each feature to learn different input images. In this experiment, activity base Homostasis is used with the parameters $activity\ rate = 20$, $updating\ rate = 0.01$, and $decay\ rate = 1$.