SOM Exercise

Richard Stöckl, 11908080 and Victor Olusesi, 11776826 and Richard Binder, 01425185

We implemented Task (f) of the coding assignment: SOM comparison. We chose Aggregation as coloring, comparing one against n maps. Our GitHub Repo is a fork from the original PySOMVis project and available under this link.

In [1]:
```
# Switch between bokeh or matplotlib backend
# backend = "bokeh"
backend = "matplotlib"

import os.path
import numpy as np
import pandas as pdcoding
import gzip
import panel as pn
import holoviews as hv
import matplotlib
import matplotlib.pyplot as plt
from holoviews import opts
from bokeh.plotting import show
import math
from SOMToolBox_Parse import SOMToolBox_Parse
from pysomvis import PySOMVis
from IPython.display import Image, display, Markdown
```

In [2]:
```
%matplotlib inline
matplotlib.use('TkAgg')
hv.extension(backend)
```

In [3]:
```
def show_layout(layout):
    render = hv.render(layout)
    if backend == "bokeh":
        show(render)
    elif backend == "matplotlib":
        display(render)
```

Usage

To use our comparison implementation, use any of the SOMs in the chainlink or 10clusters datasets folders and call the show_comparison_histo function. For examples, see the "Comparison" section.

Implementation

The implementation is based on the mathematical summary of this TU Wien master thesis by Doris Baum.

Get input data and class data

In [4]:
```
chainlink_base_folder = "datasets\\chainlink"
chainlink_idata = SOMToolBox_Parse(os.path.join(chainlink_base_folder, 'chainlink.v
chainlink_classes = SOMToolBox_Parse(os.path.join(chainlink_base_folder, 'chainlink
cluster_base_folder = "datasets\\10clusters"
cluster_idata = SOMToolBox_Parse(os.path.join(cluster_base_folder, '10clusters.vec'
cluster_classes = SOMToolBox_Parse(os.path.join(cluster_base_folder, '10clusters.cl
```

Get a comparison histogram between one main SOM and one or more comparison SOMs

The comparison implementation involves two key steps for each provided comparison SOM. Firstly, for every input vector, the position of the unit it is mapped to is determined in both the main SOM and the comparison SOM. Then the Euclidean distance between these positions is computed and added to the total distance of the corresponding unit. This distance is individually tracked for each unit in the distance matrix at the same position as the units position in the main SOM. Additionally, a seperate counter, of which every unit has its own, is incremented, to keep track of the amount of vectors mapped to a unit. Once this process is complete, the average distance for each unit is obtained by dividing its total distance by the counter value. This process is repeated for every additional comparison SOM. In the final step, the average distances are summed up across all the comparison SOMs and divided by the number of comparison SOMs.

In [5]:
```
def get_comparison_histo(_m, _n, _weights_main, _idata, _weights_compare):
    _s = len(_weights_compare)
    distanceMatrix = np.zeros((_s, 2, _m , _n))

    for k, w in enumerate(_weights_compare):
        w = w['arr']
        #distance calculation
        for vector in _idata:
            position =np.argmin(np.sqrt(np.sum(np.power(_weights_main - vector, 2),
```

```
            x1 = position % _n
            y1 = position // _n

            position2 =np.argmin(np.sqrt(np.sum(np.power(w - vector, 2), axis=1)))
            x2 = position2 % _n
            y2 = position2 // _n

            distance = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)

            distanceMatrix[k][0][y1][x1] += distance
            distanceMatrix[k][1][y1][x1] += 1

        #average calculation
        for j in range(_n):
            for i in range(_m):
                if distanceMatrix[k][1][i][j] == 0:
                    distanceMatrix[k][0][i][j] = 0
                else:
                    distanceMatrix[k][0][i][j] = distanceMatrix[k][0][i][j] / dista

    distanceMatrixAvg = distanceMatrix[:, 0, :, :]
    distanceMatrixAvg = distanceMatrixAvg.sum(axis=0)/_s

    norm = distanceMatrixAvg.max() - distanceMatrixAvg.min()
    if norm <= 0.0000001:
        norm = 0.0000001
    distanceMatrixNormalized = (distanceMatrixAvg - distanceMatrixAvg.min())/norm

    return distanceMatrixNormalized
```

Get a hit histogram of a SOM

In [6]:
```
#HitHistogram
def get_histo(_m, _n, _weights, _idata):
    hist = np.zeros(_m * _n)
    for vector in _idata:
        position =np.argmin(np.sqrt(np.sum(np.power(_weights - vector, 2), axis=1))
        hist[position] += 1

    norm = hist.max() - hist.min()
    if norm <= 0.0000001:
        norm = 0.0000001
    hist = (hist - hist.min())/norm

    return hist.reshape(_m, _n)
```

Show comparison histogram and hit histogram of main and comparison SOMs

In [7]:
```
def show_comparison_histo(main, compare, show_SOMs=True, base_folder=chainlink_base
    # Change filenames to filepath
    weights_main = SOMToolBox_Parse(os.path.join(base_folder, 'compare', main)).rea
```

```
weights_compare_path = []
weights_compare = []
for f in compare:
    f = os.path.join(base_folder, 'compare', f)
    weights_compare_path.append(f)
    w = SOMToolBox_Parse(f).read_weight_file()
    weights_compare.append(w)

soms = []

# Show SOMs
if show_SOMs:
    som = get_histo(weights_main['ydim'], weights_main['xdim'], weights_main['a
    som = hv.Image(som).opts(xaxis=None, yaxis=None)
    soms.append(som.relabel('Main SOM Hit Hist').opts(cmap='jet'))
    for w in weights_compare:
        som = get_histo(w['ydim'], w['xdim'], w['arr'], _idata['arr'])
        som = hv.Image(som).opts(xaxis=None, yaxis=None)
        soms.append(som.relabel('SOM Hit Hist').opts(cmap='jet'))

# Show Comparison between Main and Comparison SOMs
comparison = get_comparison_histo(weights_main['ydim'], weights_main['xdim'], w
comparison = hv.Image(comparison).opts(xaxis=None, yaxis=None)
soms.append(comparison.relabel('Colored Comparison').opts(cmap='jet'))
layout = hv.Layout(soms)
show_layout(layout)
```

Datasets

All of our SOMs were trained on the chainlink and clustering dataset with the Java based SOMToolbox. The small SOM has 10x10 units, and the large SOM has 60x100 units.

Where not mentioned otherwise, training used the following parameters:

`learnRate=0.3, sigma=7, randomSeed=7` for the chainlink dataset and
`learnRate=0.3, randomSeed=7` for the clustering datasets.

Comparisons

Each comparison has one main SOM and one or more SOMs that the main SOM is compared with. We visualize all SOMs (main and comparison) and the resulting colored comparison map between them. The main SOM is always the map labeled "Main SOM" and the resulting comparison is always the map labeled "Colored Comparison". All other graphs are the SOMs that the main SOM is compared with.

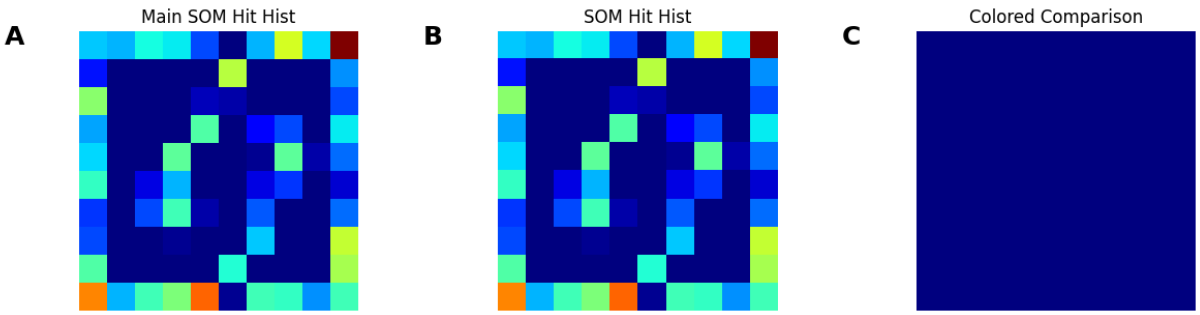
Comparing a SOM with itself

When a SOM is compared with itself, there shouldn't be any differences. The comparison

histogram is therefore 0 in all units. Its visualization has the same color in all units. We use a small SOM trained on the chainlink dataset, and a small SOM trained on the 10-clusters dataset.

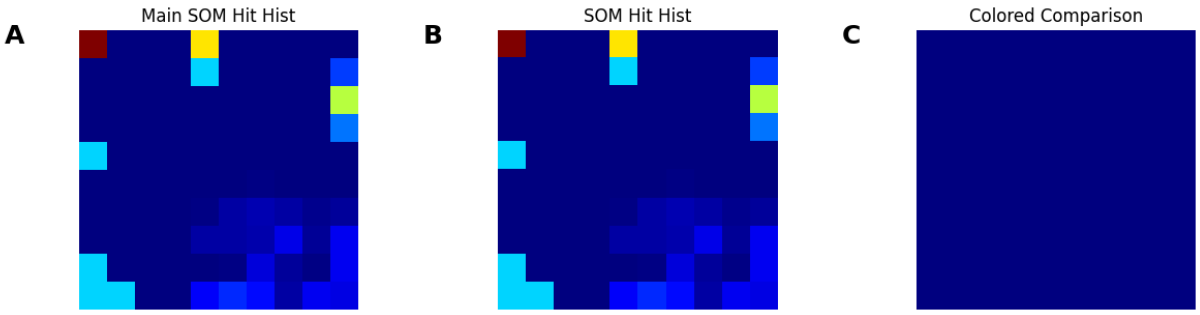
Small SOM Chainlink Dataset

In [8]:
```
show_comparison_histo(
    main="chainlink_1000.wgt.gz",
    compare=["chainlink_1000.wgt.gz"],
)
```



Small SOM Clustering Dataset

In [9]:
```
show_comparison_histo(
    base_folder=cluster_base_folder,
    _idata=cluster_idata,
    main="10clusters_1000.wgt.gz",
    compare=["10clusters_1000.wgt.gz"],
)
```



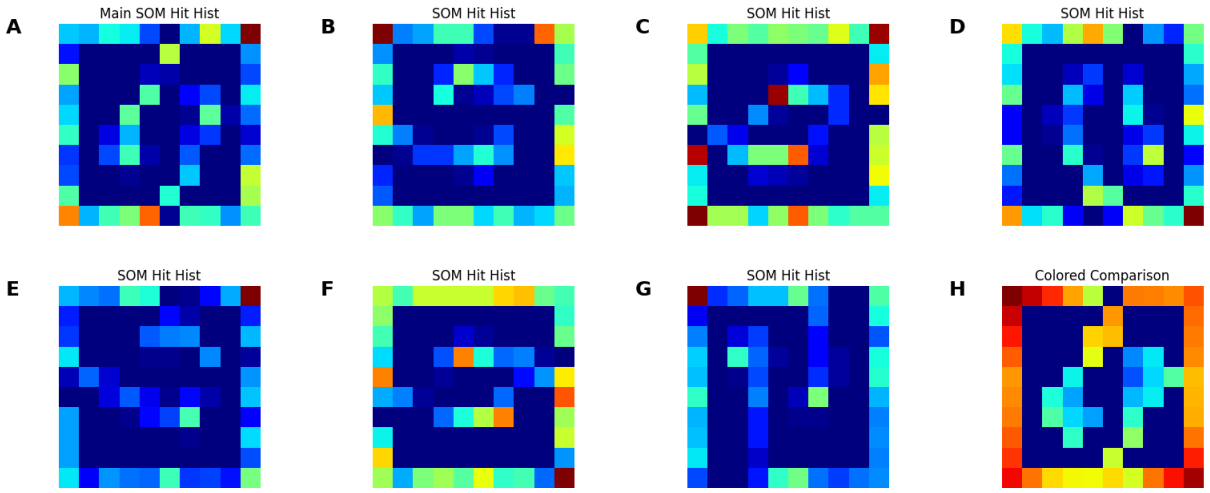
Comparing a SOM with different random seeds

In this experiment, Main and Comparison SOMs are trained with exactly the same parameters. The SOMs only differ by the random seed that was used during training. The Main SOM has seed 7, the others have seeds 1-6. We can see the non-deterministic nature of SOM-training here. The Comparison histogram shows large differences in almost all relevant units.

Small SOM Chainlink Dataset

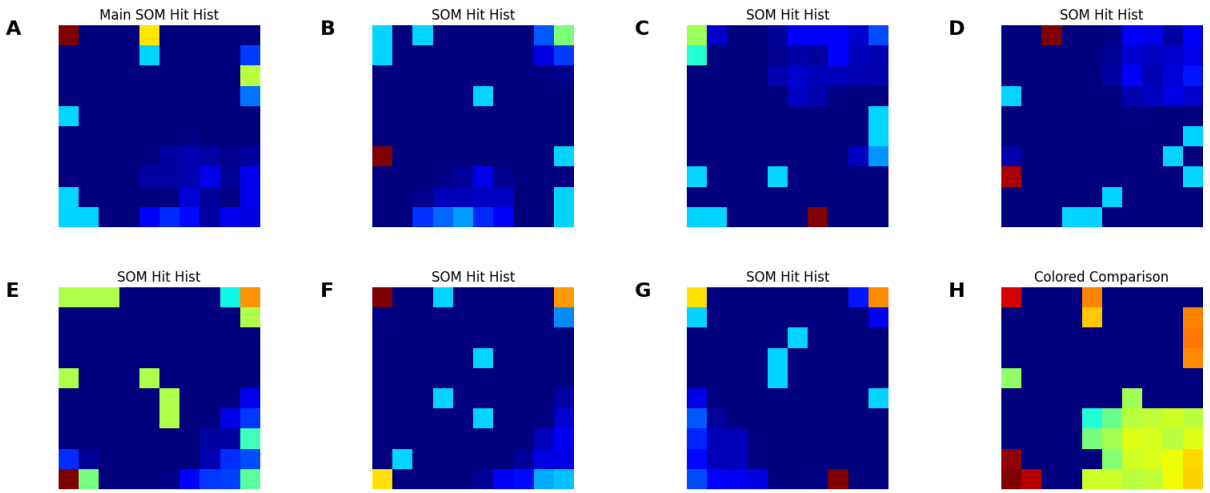
In [10]:
```
show_comparison_histo(
```

```
    main="chainlink_1000.wgt.gz",
    compare=["chainlink_1000_1.wgt.gz",
             "chainlink_1000_2.wgt.gz",
             "chainlink_1000_3.wgt.gz",
             "chainlink_1000_4.wgt.gz",
             "chainlink_1000_5.wgt.gz",
             "chainlink_1000_6.wgt.gz"]
)
```



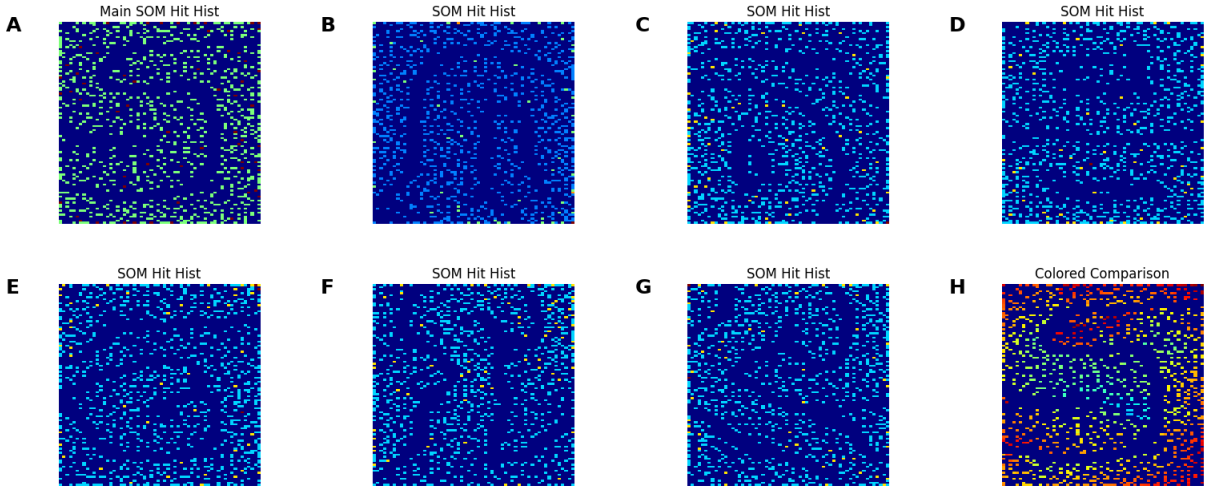
Small SOM 10-clusters Dataset

In [11]:
```
show_comparison_histo(
    base_folder=cluster_base_folder,
    _idata=cluster_idata,
    main="10clusters_1000.wgt.gz",
    compare=["10clusters_1000_1.wgt.gz",
             "10clusters_1000_2.wgt.gz",
             "10clusters_1000_3.wgt.gz",
             "10clusters_1000_4.wgt.gz",
             "10clusters_1000_5.wgt.gz",
             "10clusters_1000_6.wgt.gz"]
)
```



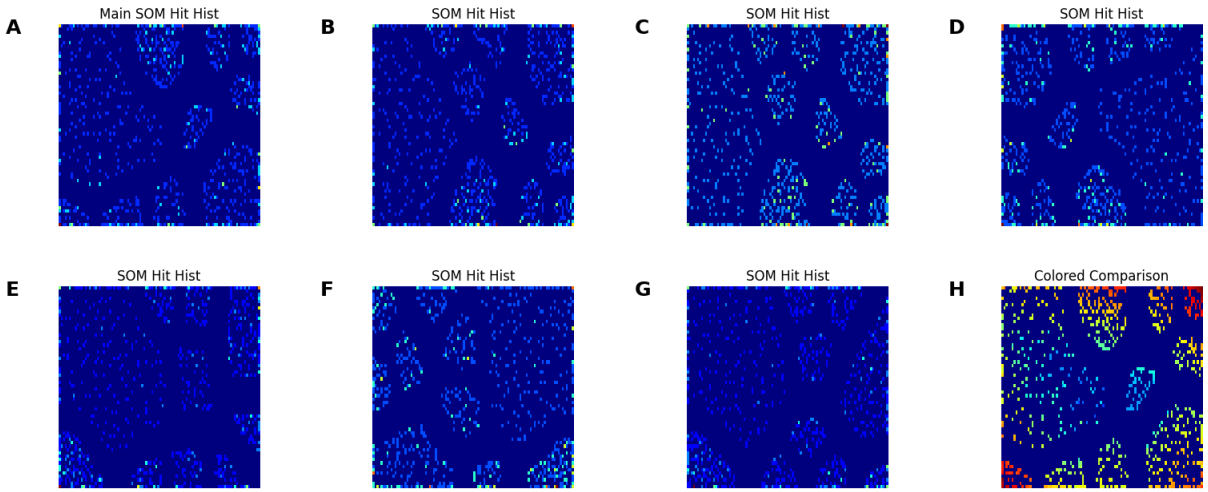
Large SOM Chainlink Dataset

In [12]:
```
show_comparison_histo(
    main="chainlink_L_100000.wgt.gz",
    compare=["chainlink_L_100000_1.wgt.gz",
             "chainlink_L_100000_2.wgt.gz",
             "chainlink_L_100000_3.wgt.gz",
             "chainlink_L_100000_4.wgt.gz",
             "chainlink_L_100000_5.wgt.gz",
             "chainlink_L_100000_6.wgt.gz"]
)
```



Large SOM Clustering Dataset

In [13]:
```
show_comparison_histo(
    base_folder=cluster_base_folder,
    _idata=cluster_idata,
    main="10clusters_L_100000.wgt.gz",
    compare=["10clusters_L_100000_1.wgt.gz",
             "10clusters_L_100000_2.wgt.gz",
             "10clusters_L_100000_3.wgt.gz",
             "10clusters_L_100000_4.wgt.gz",
             "10clusters_L_100000_5.wgt.gz",
             "10clusters_L_100000_6.wgt.gz"]
)
```



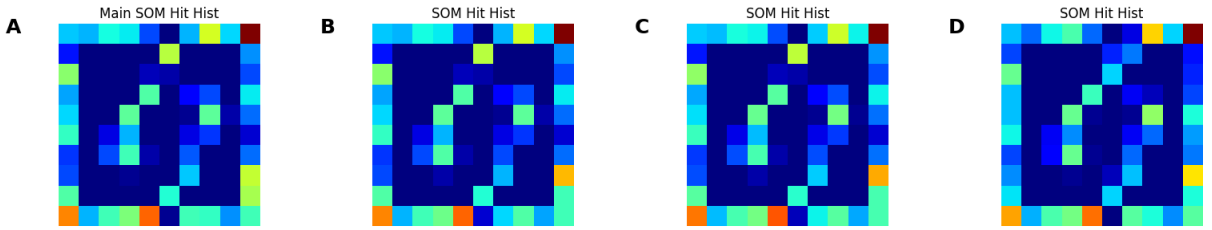
Comparing SOMs from neighbouring training iterations

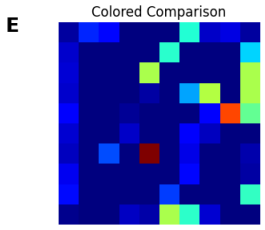
The main small SOM is trained in 1000 iterations and compared to the SOMs after 1001, 1005 and 1100 iterations. The main large SOM is trained in 100,000 iterations and compared to the SOMs after 100,100 and 110,000 iterations. All SOMs are trained with the same seed (7) to ensure deterministic training. The comparison shows there there is only small differences between the SOMs because the SOM does not change much in a low number of iterations, and the SOM also does not change much when training is close to conversion.

We also provide the comparison result of the Java SOMToolbox. There are some minor differences between our implementation and the SOMToolbox in the case of the small SOM, but overall the results are similar. For the large SOM, there seems to be an issue in the Java SOMToolbox. It seems as though units with only one vector are not accurately compared in the Toolbox, and the result only shows differences in some units. Since our large SOM has more more units than vectors, almost all units contain only one vector. The SOMToolbox might be using thresholding for the pairwise distances, whereas we do not.

Small SOM Chainlink Dataset

In [14]:
```
show_comparison_histo(
    main="chainlink_1000.wgt.gz",
    compare=["chainlink_1001.wgt.gz",
             "chainlink_1005.wgt.gz",
             "chainlink_1100.wgt.gz"]
)
```

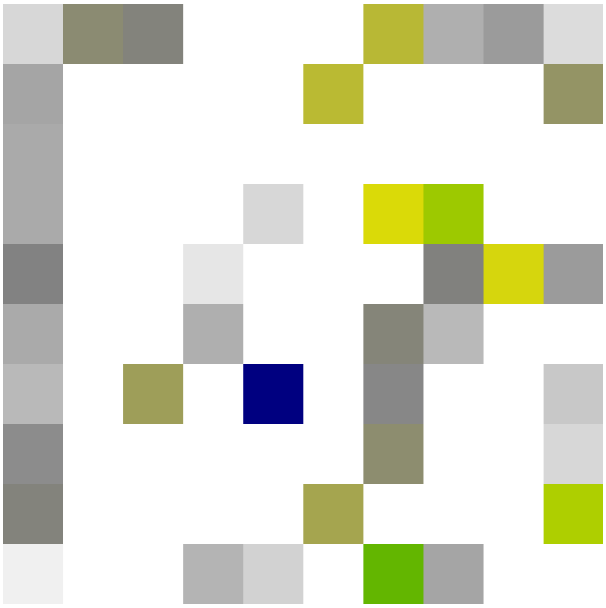




Small SOM Chainlink Dataset - SOMToolbox comparison

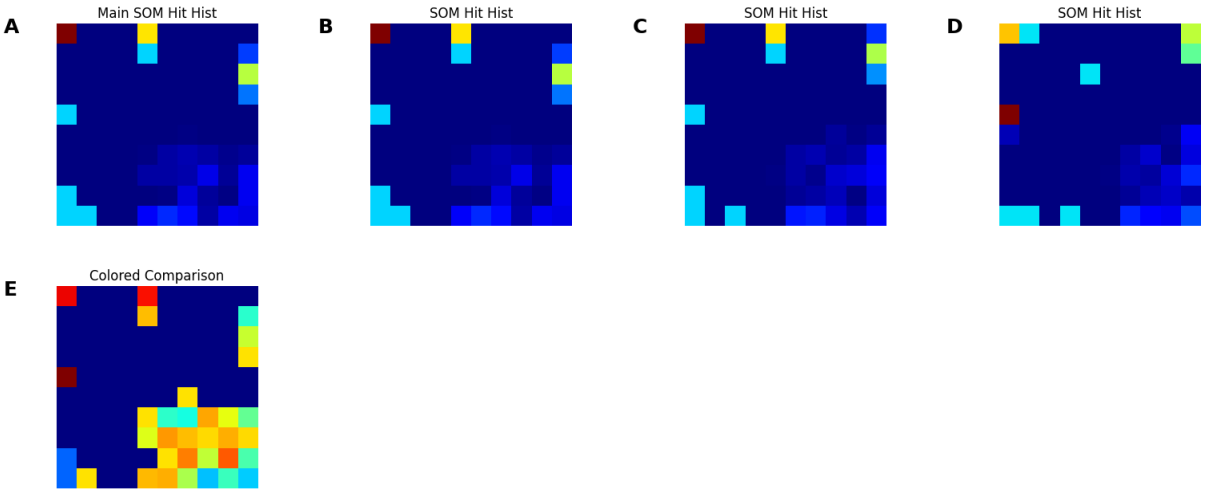
In [15]:
```
image_path = os.path.join(chainlink_base_folder, 'compare', 'compare_1000_to_1001_1
display(Markdown(f"Java SOMToolbox Comparison"))
display(Image(filename=image_path, width=300, height=300))
```

Java SOMToolbox Comparison



Small SOM Clustering Dataset

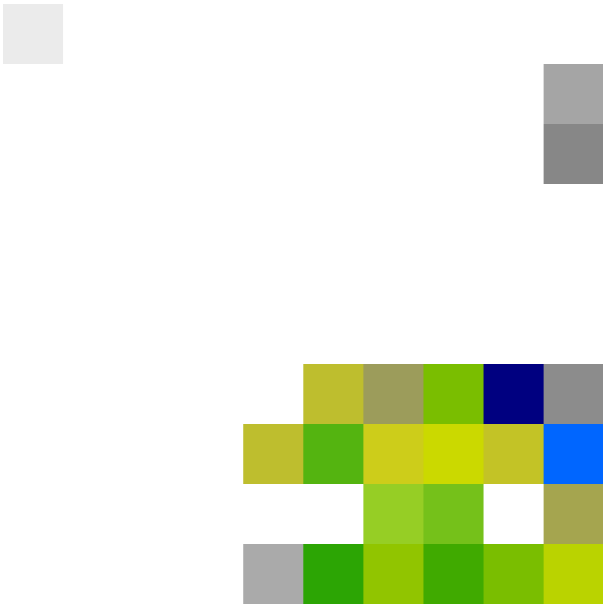
In [16]:
```
show_comparison_histo(
    base_folder=cluster_base_folder,
    _idata=cluster_idata,
    main="10clusters_1000.wgt.gz",
    compare=["10clusters_1001.wgt.gz",
             "10clusters_1005.wgt.gz",
             "10clusters_1100.wgt.gz"]
)
```



Small SOM Clustering Dataset - SOMToolbox comparison

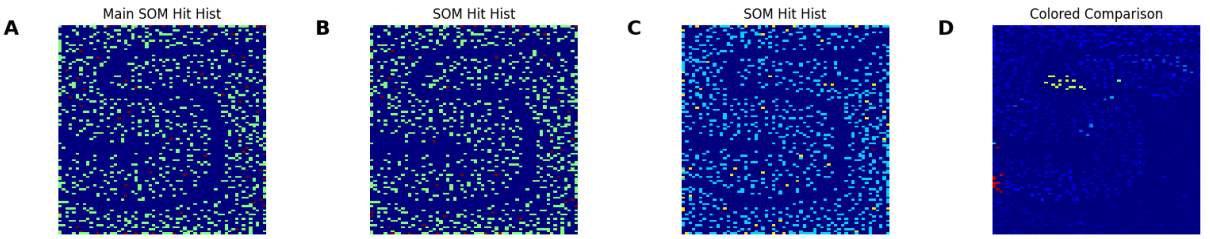
In [17]:
```
image_path = os.path.join(cluster_base_folder, 'compare', 'compare_1000_to_1001_100
display(Markdown(f"Java SOMToolbox Comparison"))
display(Image(filename=image_path, width=300, height=300))
```

Java SOMToolbox Comparison



Large SOM Chainlink Dataset

In [18]:
```
show_comparison_histo(
    main="chainlink_L_100000.wgt.gz",
    compare=["chainlink_L_100100.wgt.gz",
             "chainlink_L_110000.wgt.gz"]
)
```



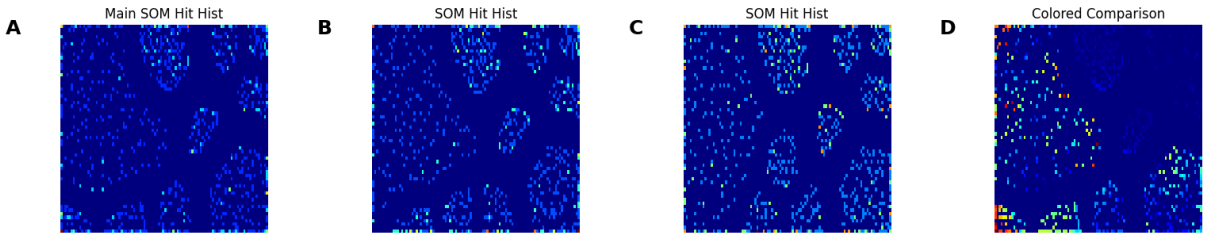
Large SOM Chainlink Dataset - SOMToolbox comparison

In [19]:
```
image_path = os.path.join(chainlink_base_folder, 'compare', 'compare_100000_to_1100
display(Markdown(f"Java SOMToolbox Comparison"))
display(Image(filename=image_path, width=300, height=300))
```

Java SOMToolbox Comparison

Large SOM 10-clusters Dataset

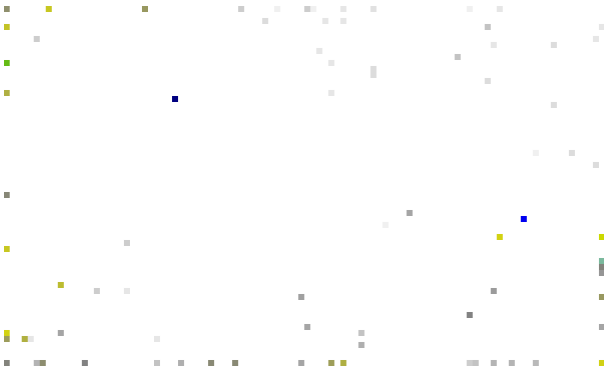
In [20]:
```
show_comparison_histo(
    base_folder=cluster_base_folder,
    _idata=cluster_idata,
    main="10clusters_L_100000.wgt.gz",
    compare=["10clusters_L_100100.wgt.gz",
             "10clusters_L_110000.wgt.gz"]
)
```



Large SOM 10-clusters Dataset - SOMToolbox comparison

In [21]:
```
image_path = os.path.join(cluster_base_folder, 'compare', 'compare_100000_to_100100
display(Markdown(f"Java SOMToolbox Comparison"))
display(Image(filename=image_path, width=300, height=300))
```

Java SOMToolbox Comparison

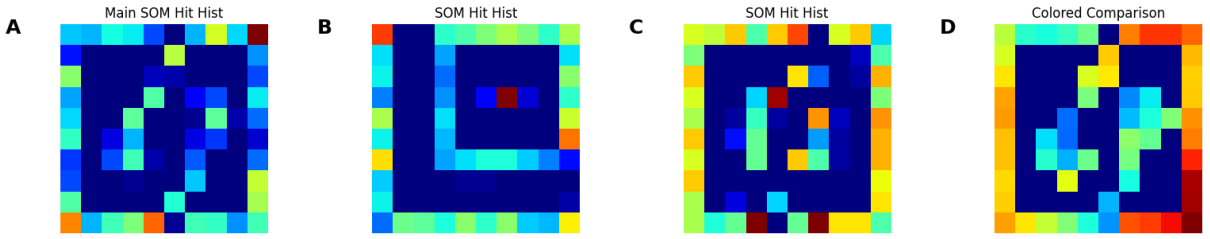


Comparing SOMs from far apart training iterations

The main small SOM is trained in 1000 iterations and compared to the SOMs after 3000 and 10000 iterations. The main large SOM is trained in 100,000 iterations and compared to the SOMs after 1000 and 10,000 iterations. All SOMs are trained with the same seed (7) to ensure deterministic training. Unsurprisingly, the comparison shows there is large differences between SOMs of far apart training iterations.

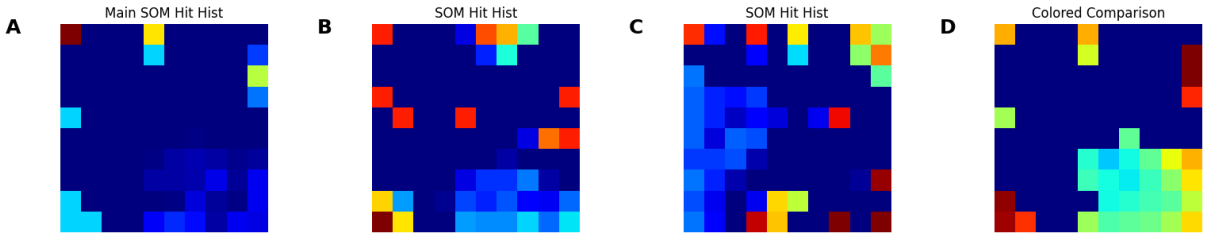
Small SOM Chainlink Dataset

In [22]:
```
show_comparison_histo(
    main="chainlink_1000.wgt.gz",
    compare=["chainlink_3000.wgt.gz",
             "chainlink_10000.wgt.gz"]
)
```



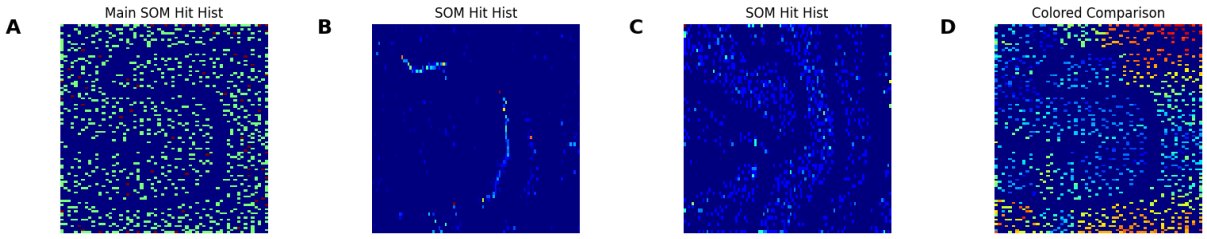
Small SOM 10-clusters Dataset

In [23]:
```
show_comparison_histo(
    base_folder=cluster_base_folder,
    _idata=cluster_idata,
    main="10clusters_1000.wgt.gz",
    compare=["10clusters_3000.wgt.gz",
             "10clusters_10000.wgt.gz"]
)
```



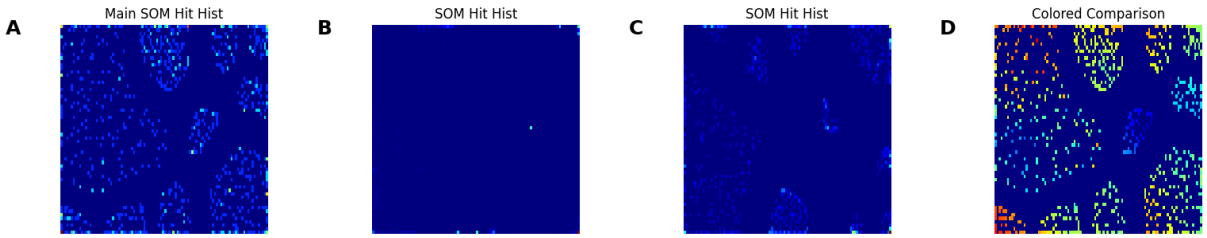
Large SOM Chainlink Dataset

In [24]:
```
show_comparison_histo(
    main="chainlink_L_100000.wgt.gz",
    compare=["chainlink_L_1000.wgt.gz",
             "chainlink_L_10000.wgt.gz"]
)
```



Large SOM 10-clusters Dataset

In [25]:
```
show_comparison_histo(
    base_folder=cluster_base_folder,
    _idata=cluster_idata,
    main="10clusters_L_100000.wgt.gz",
    compare=["10clusters_L_1000.wgt.gz",
             "10clusters_L_10000.wgt.gz"]
)
```



In [ ]: