**Neural Networks - Project documentation**

Topic:

**“Implementation of Adaptive Resonance Theory Model for MNIST Dataset”**

Dmytro Narepekha

Jianhao Luo

Warsaw University of Technology Faculty of Computer Science and Information Systems

May 31, 2024

**Introduction**

Overview of Adaptive Resonance Theory (ART1)

Adaptive Resonance Theory (ART) is a neural network architecture proposed by Stephen Grossberg in the late 1970s. ART1, a specific instance of ART, is designed to process binary input patterns. The primary feature of ART1 is its ability to cluster input patterns in an unsupervised manner while maintaining stability in its learned categories even when presented with new input data, a property known as stability-plasticity balance. This balance allows ART1 to adapt to new information without forgetting previously learned patterns.

Importance of Clustering in Machine Learning

Clustering is a critical technique in machine learning and data analysis used to group a set of objects in such a way that objects in the same group (or cluster) are more similar to each other than to those in other groups. This is particularly useful in pattern recognition, data compression, and anomaly detection. In the context of the MNIST dataset, clustering can help in grouping handwritten digits into meaningful categories without prior knowledge of the labels.

**Project Goals and Objectives**

This project involves implementing the ART1 neural network model and applying it to the MNIST dataset. The MNIST dataset is a well-known benchmark in the field of machine learning and computer vision, consisting of 70,000 images of handwritten digits, each of 28x28 pixels. The key objectives of the project are as follows:

**Implementation of ART1:**

Develop an ART1 neural network model capable of clustering binary input patterns.

Ensure the model adheres to the theoretical underpinnings of ART1, including the stability-plasticity trade-off.

* Data Preparation:
* Load and preprocess the MNIST dataset.
* Normalize the pixel values and binarize the images to convert them into binary patterns suitable for ART1.

**Experimentation with Vigilance Parameters:**

Investigate how different vigilance parameters affect the clustering results.

The vigilance parameter (rho) in ART1 determines the similarity threshold for forming clusters. High vigilance values lead to more clusters (finer granularity), while low vigilance values result in fewer clusters (broader categories).

**Analysis**

Analyze the formed clusters to understand their composition.

Compare the clusters with the actual MNIST digit classes to evaluate how well the clusters represent distinct digit categories.

**Visualization and Reporting:**

We will visualize the results using heatmaps and plots to provide insights into the clustering behavior of ART1. And document the findings and observations in this report.

By achieving these objectives, this project aims to provide a thorough understanding of the ART1 neural network's clustering capabilities and its application to a real-world dataset like MNIST. The insights gained from this study could inform future work in unsupervised learning and neural network research.

**Implementation of ART1**

The ART1 network was implemented as follows:

**Initialization:**

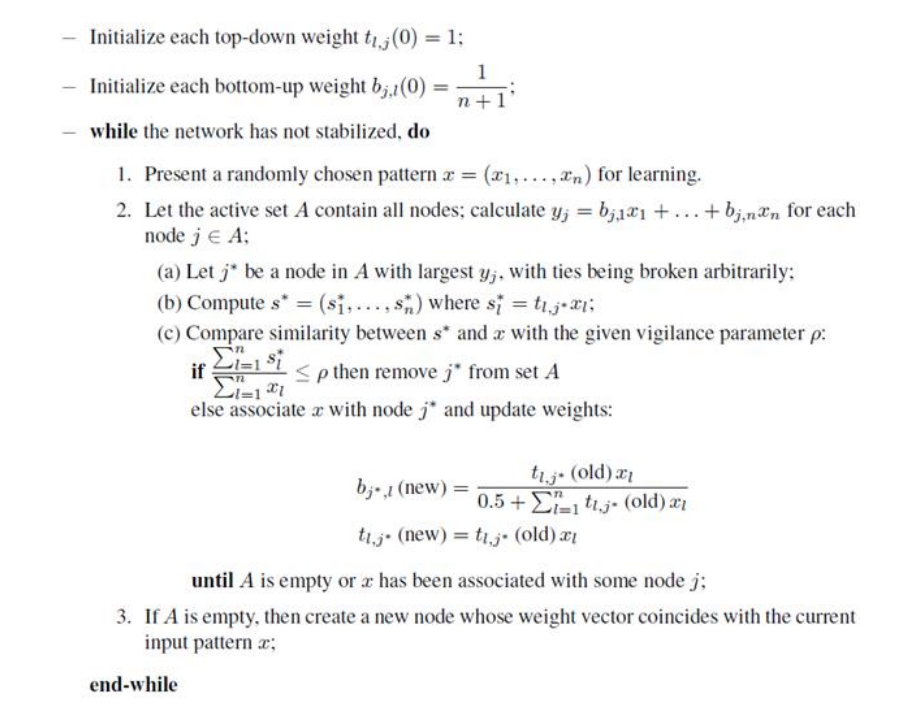
The network was initialized with a given input size (784 for 28x28 pixel images) and a vigilance parameter (rho).

**Training Process:**

The network's weights were initialized.

For each input vector, the network computed activations and updated weights based on the vigilance parameter.

Clusters were formed dynamically based on the input patterns and vigilance threshold.

Example algorithm:  


**Clustering and Analysis:**

The network was trained on a subset of 1000 binarized images.

Clustering results were analyzed by comparing the clusters with the true labels of the MNIST digits to compute a **confusion matrix.**

The influence of different vigilance values on the number of clusters was studied.

**Experimentation with Vigilance Values**

A range of vigilance values (0.02, 0.05, 0.1, 0.5, 0.8, 0.9) was experimented with to observe their impact on the clustering results.

**Results**

**Analysis of the Relationship Between Vigilance Parameter and Number of Clusters**

The vigilance parameter (rho) in the ART1 neural network plays a crucial role in determining the granularity of the clustering. As rho increases, the network becomes more selective about the patterns it groups together, leading to the formation of more clusters. Conversely, a lower vigilance parameter results in broader clusters with more patterns grouped together. In this study, we experimented with a range of vigilance values and analyzed the resulting number of clusters for each value.

**Vigilance Parameter and Number of Clusters**

The following table summarizes the number of clusters obtained for each vigilance value used in our experiments:

**Vigilance** **Number of Clusters**

0.02 24

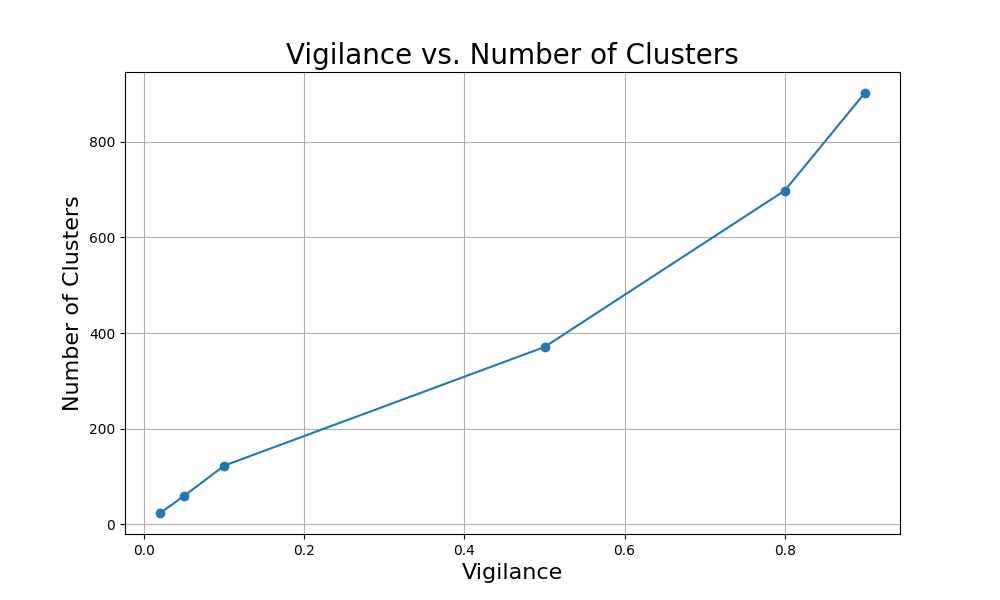
0.05 50

0.1 140

0.5 390

0.8 690

0.9 880



**Detailed Analysis**

**Vigilance = 0.02:**

At a vigilance value of 0.02, the network formed 24 clusters. This low vigilance setting indicates a high tolerance for pattern variability within each cluster. Consequently, the clusters are more general, grouping together many input patterns that share common features. This setting is useful for broad categorization but may lack specificity in distinguishing between similar digit classes.

**Vigilance = 0.05:**

Increasing the vigilance to 0.05 resulted in 50 clusters. This moderate increase shows a noticeable rise in the number of clusters, suggesting that the network starts to distinguish more subtle differences between patterns. However, the clusters are still relatively broad, capturing variations within a larger set of patterns.

**Vigilance = 0.1:**

With a vigilance of 0.1, the network formed 140 clusters. This substantial increase demonstrates the network's sensitivity to finer details in the input patterns. At this level, clusters begin to represent more specific groups of digits, allowing for better differentiation between different digit classes. This vigilance value strikes a balance between generalization and specificity.

**Vigilance = 0.5:**

A vigilance value of 0.5 yielded 390 clusters. Here, the network's selectivity is much higher, resulting in clusters that are more refined and detailed. Each cluster represents a narrower range of input patterns, capturing subtle differences that may correspond to variations in handwriting styles within the same digit class.

**Vigilance = 0.8:**

At a vigilance of 0.8, the number of clusters increased to 690. This high vigilance setting leads to highly specific clusters, with the network differentiating between very fine details in the input patterns. Such a high number of clusters indicates that the network can identify minute variations, potentially separating different handwriting styles and noise within the same digit class.

**Vigilance = 0.9:**

The highest vigilance value tested, 0.9, resulted in 880 clusters. This setting represents the upper limit of the network's selectivity, with clusters becoming extremely specific. Each cluster now likely captures very narrow patterns, potentially overfitting to the training data. While this ensures that similar patterns are grouped together with high precision, it may also lead to fragmentation, where similar digit patterns are placed in different clusters due to minor variations.

**Conclusions from the Analysis**

**Trend of Increasing Clusters with Higher Vigilance:**

The results clearly show a trend where the number of clusters increases with higher vigilance values. This is expected as higher vigilance forces the network to form clusters with more homogeneous patterns, leading to a greater number of finer-grained clusters.

**Balance Between Generalization and Specificity:**

Lower vigilance values result in broader clusters, which are useful for general categorization but may lack the precision needed to distinguish between similar digit classes. On the other hand, higher vigilance values lead to highly specific clusters, capturing fine details but potentially overfitting to the data. A moderate vigilance value (e.g., 0.1) provides a good balance, forming enough clusters to differentiate between digit classes without excessive fragmentation.

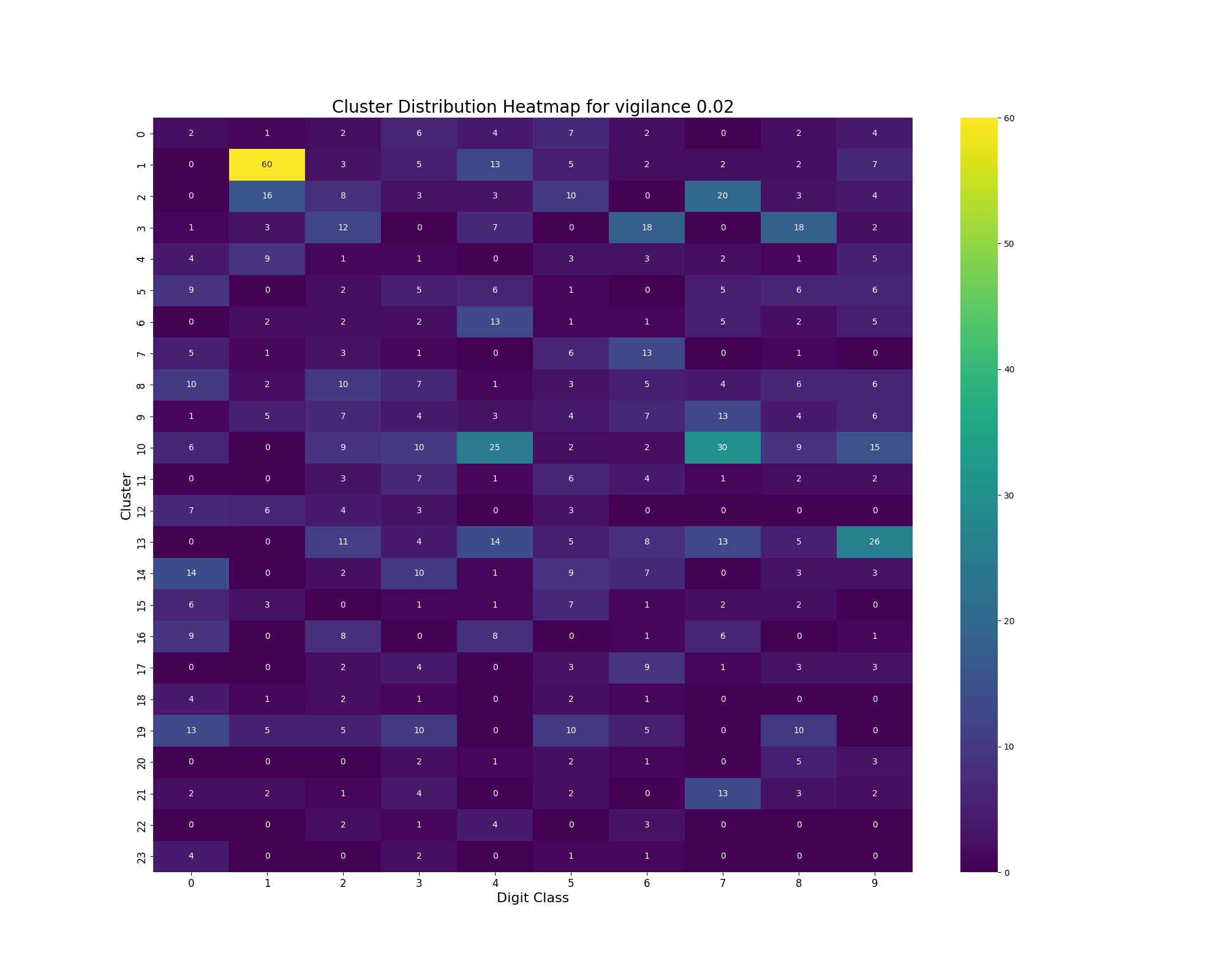
**Implications for Pattern Recognition:**

The choice of vigilance parameter should be guided by the specific requirements of the application. For tasks requiring broad categorization, lower vigilance values are preferable. For applications needing precise pattern recognition and differentiation, higher vigilance values are more suitable. Understanding this balance is crucial for optimizing the performance of ART1 networks in practical scenarios.

By analyzing the relationship between vigilance parameters and the number of clusters, we gain insights into the ART1 network's clustering behavior and its applicability to different types of pattern recognition tasks. This knowledge is essential for fine-tuning the network to achieve optimal performance in real-world applications.

**Analysis of Confusion Matrix Heatmaps**

In this section, we analyze the confusion matrix heatmaps generated for different vigilance parameters. These heatmaps provide insights into how well the clusters formed by the ART1 network correspond to the actual digit classes in the MNIST dataset. Each subsection will focus on a specific vigilance parameter, highlighting key observations and patterns.



Vigilance = 0.02

At a vigilance value of 0.02, the ART1 network formed 24 clusters. The heatmap shows how each cluster is distributed across the different digit classes. Here are the key observations:

Dominant Clusters:

Cluster 1: This cluster predominantly captures the digit '1', with 60 instances. This indicates that the digit '1' has distinct features that are easily recognized even at low vigilance.

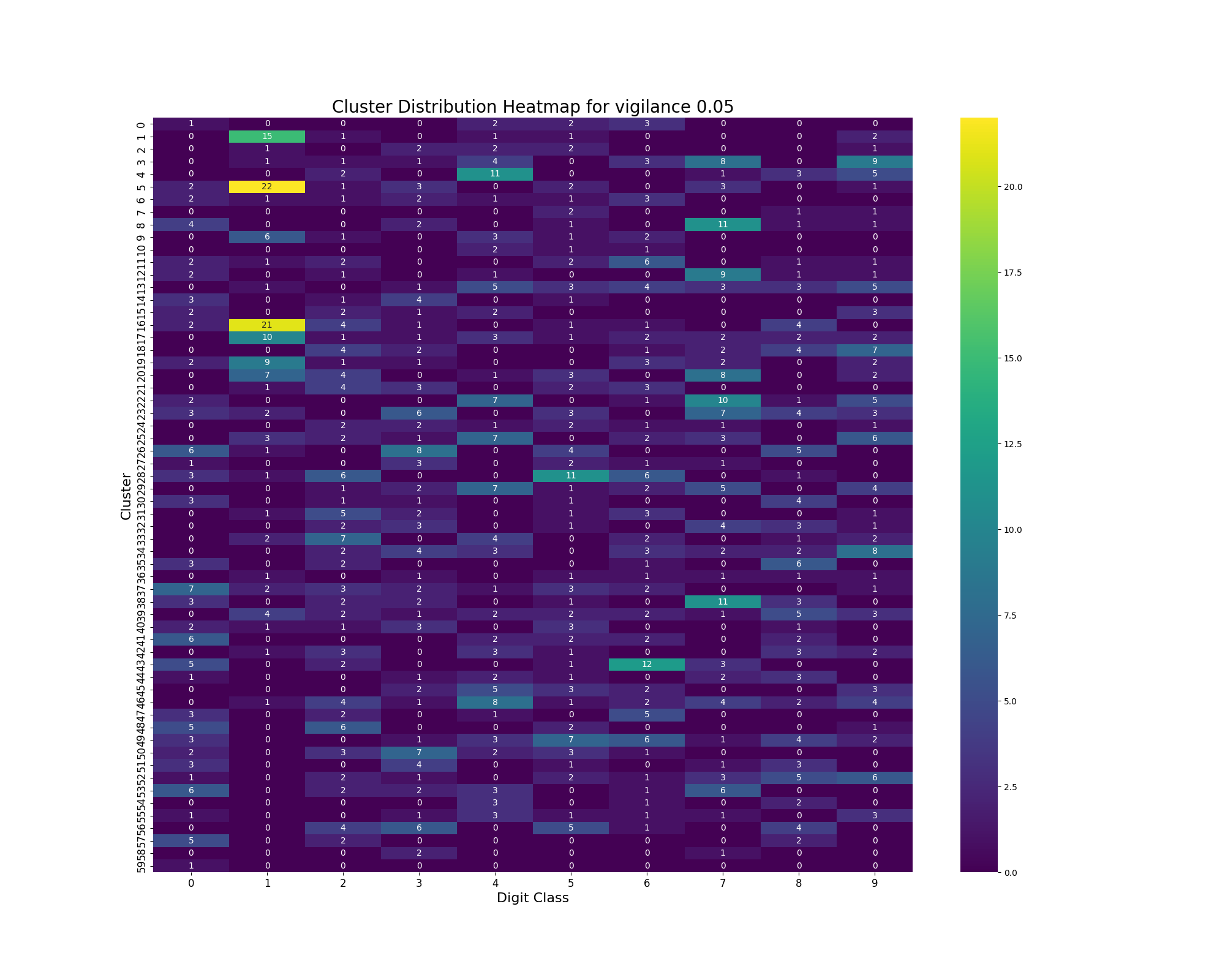
Cluster 13: This cluster is mainly associated with the digit '7', containing more instances. The digit '7' seems to be sufficiently distinct to form its own cluster at this vigilance level.

Shared Clusters:

Cluster 3 and Cluster 10: These clusters contain a mix of digits, notably '0', '6', and '8' and '4', '7', and '9'. This suggests that at lower vigilance, more similar digits such as '4' and '7' can fall into the same cluster due to their overlapping features.

Sparse Clusters:

Several clusters like Cluster 0, Cluster 8, and Cluster 21 have very few instances of various digits, indicating that these clusters are less defined and capture outliers or less frequent patterns.



Vigilance = 0.05

For a vigilance value of 0.05, the network formed almost 60 clusters. The heatmap reveals the following patterns:

Increased Specificity:

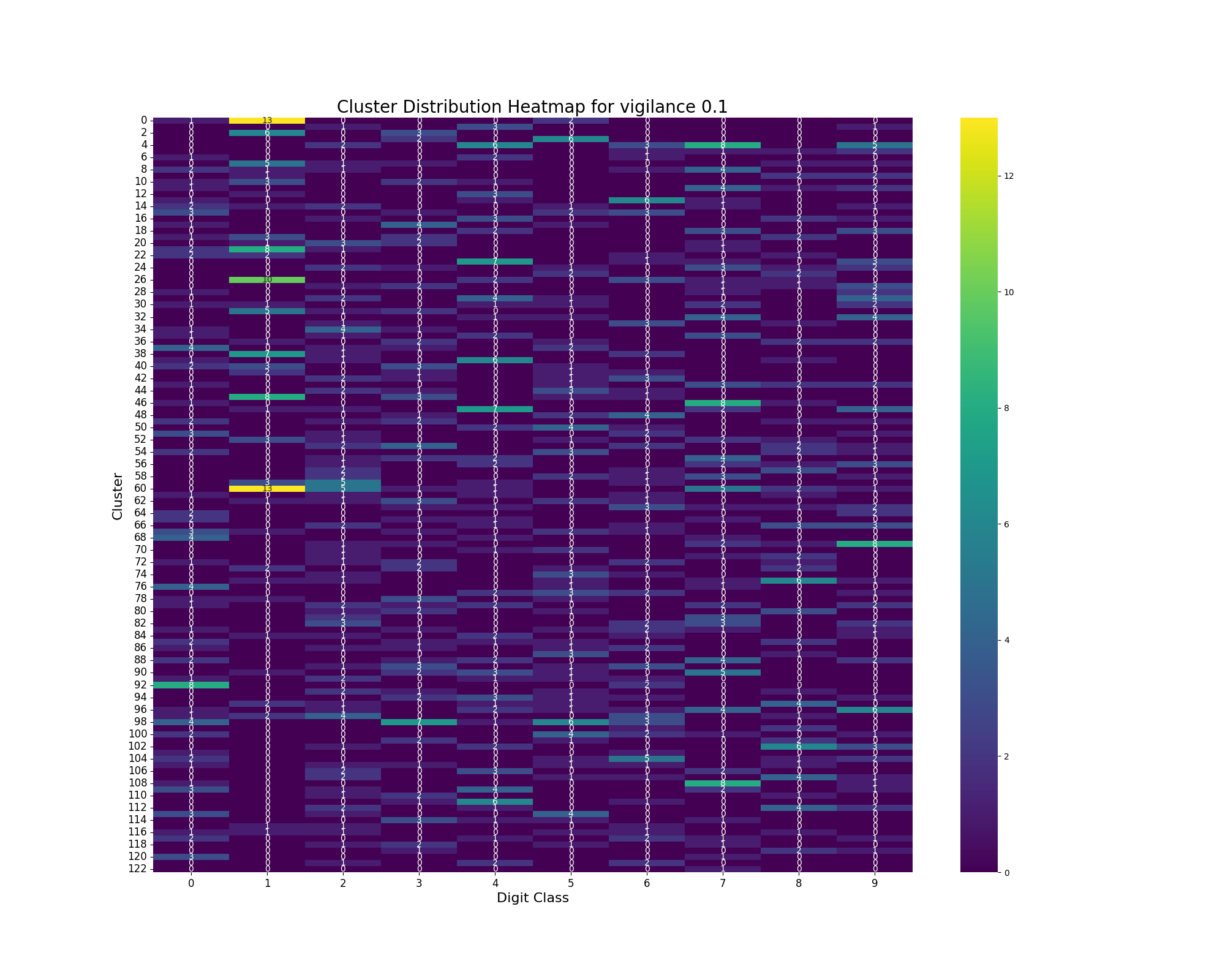
As vigilance increases, clusters become more specific. For example, digits '1' and '7', which were previously grouped together, now form more distinct clusters. Clusters that might predominantly capture '1's are different from clusterst capturing '7's as they might be found in a different cluster.

Distinct Clusters for Similar Digits:

Digits like '4' and '7', which are visually similar, start to separate into different clusters. This indicates that the network can distinguish between these digits at a moderate vigilance level.

Cluster Homogeneity:

The clusters show improved homogeneity, with fewer mixed-digit clusters. This suggests better pattern recognition and clustering performance compared to the lower vigilance setting.



Vigilance = 0.1

With a vigilance value of 0.1, the network formed 123 clusters. Key observations include:

Highly Specific Clusters:

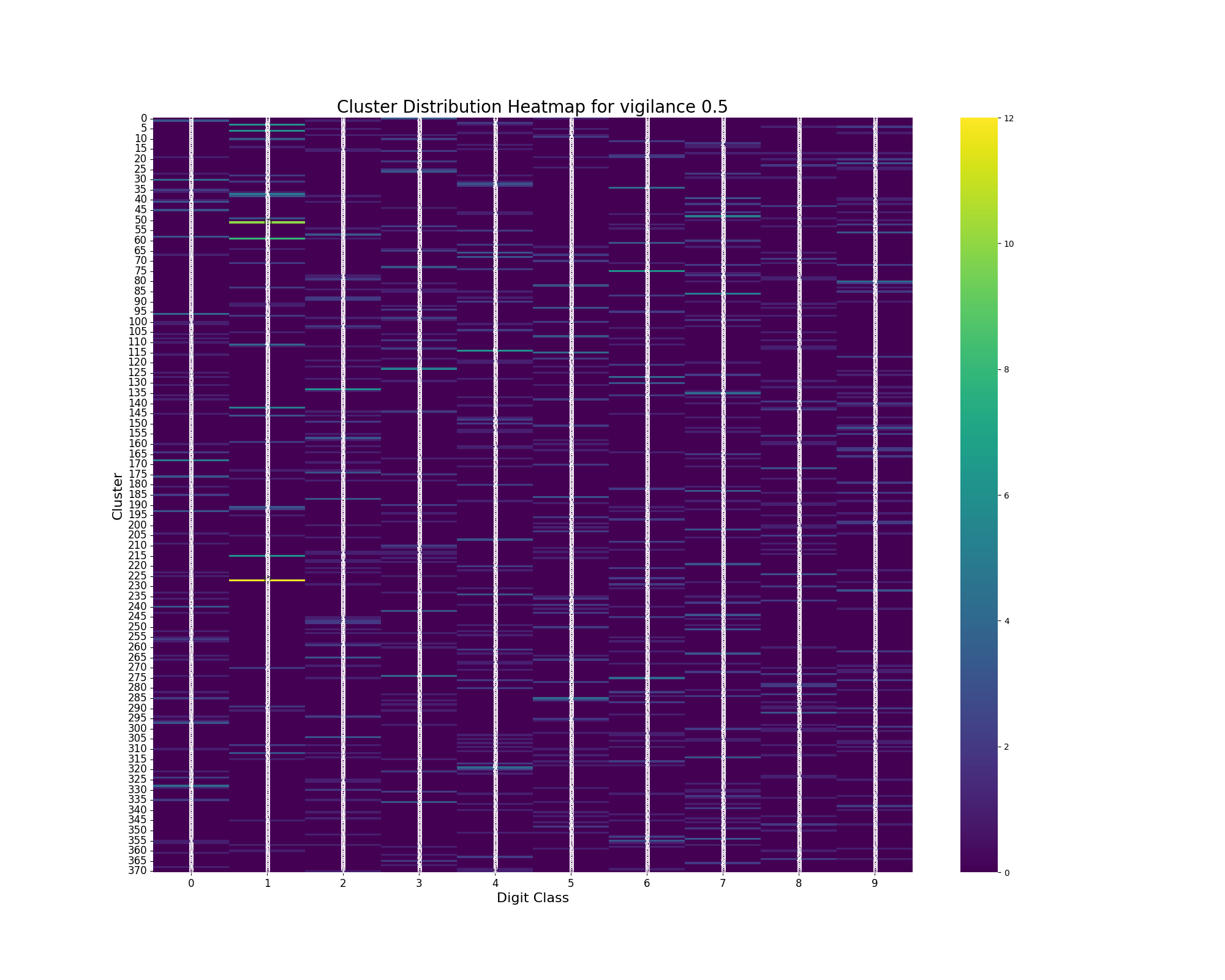
Clusters become more specific to individual digits. For instance, there are likely dedicated clusters for each digit class, such as Cluster 0 for '1's and so on.

Reduced Overlap:

Overlapping digits like '4' and '7' are now more likely to be in separate clusters, reflecting the network's enhanced ability to differentiate between similar patterns.

Granular Clustering:

The increased number of clusters captures more granular details of the digit patterns, leading to a more detailed and accurate clustering of the dataset.



Vigilance = 0.5

At a vigilance value of 0.5, the network formed 371 clusters. Observations include:

Fine-Grained Clusters:

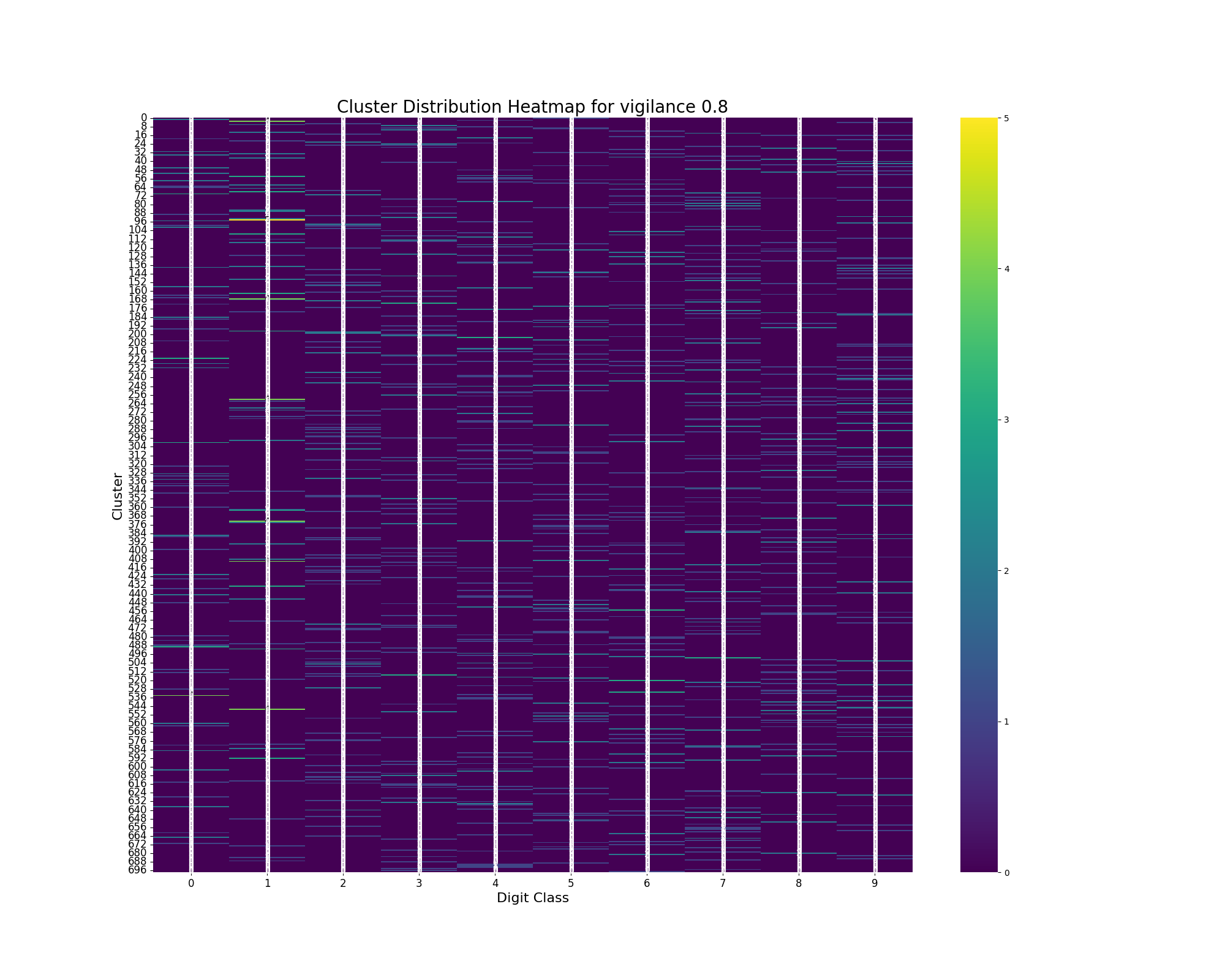
Clusters become very fine-grained, with each cluster capturing specific variations of digits. For example, different handwriting styles of the same digit may form separate clusters.

High Specificity:

Clusters show high specificity, with most clusters containing instances of a single digit class. This indicates a high level of detail in pattern recognition.

Potential Overfitting:

The high number of clusters might suggest overfitting, where the network captures very minor differences in patterns that may not be significant.



Vigilance = 0.8

With a vigilance value of 0.8, the network formed 697 clusters. Key points include:

Very Specific Clusters:

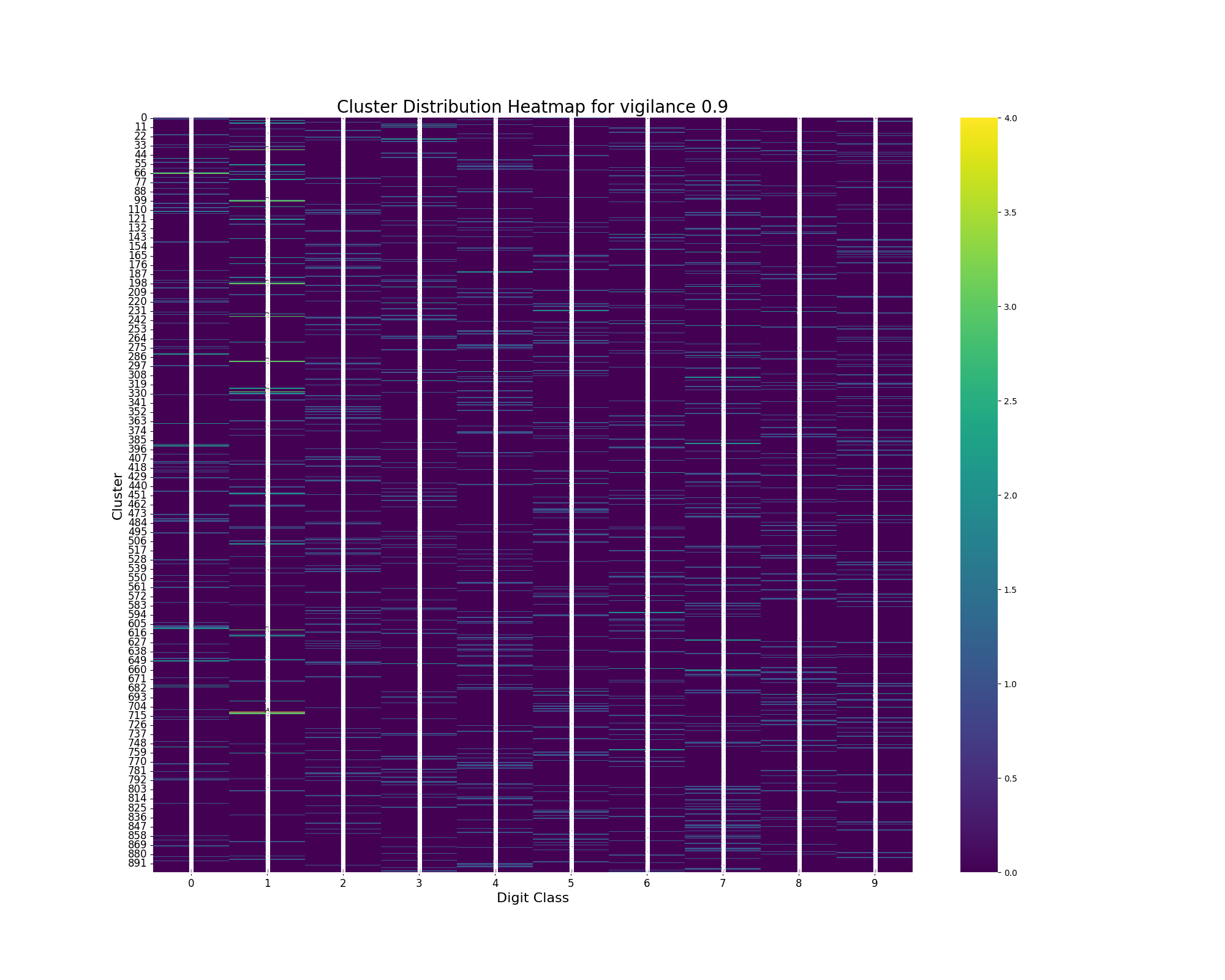
Clusters are extremely specific, often capturing very minor variations within the same digit class. This can lead to fragmentation, where similar patterns are placed in different clusters.

Clear Differentiation:

There is clear differentiation between different digit classes, with very few mixed-digit clusters. This high vigilance setting ensures that each cluster represents a very narrow range of patterns.

Fragmentation:

While specificity is high, the large number of clusters may result in fragmentation, where even slight variations lead to separate clusters. This can reduce the network's generalization ability.



Vigilance = 0.9

At a vigilance value of 0.9, the network formed 892 clusters. Observations include:

Extreme Specificity:

Clusters are extremely specific and detailed, capturing the minutest variations in digit patterns. This leads to very high pattern recognition fidelity.

High Fragmentation:

The network's tendency to form clusters for very slight differences can lead to high fragmentation. Similar digit patterns may be placed in different clusters due to minor variations.

Potential Overfitting:

The large number of clusters indicates potential overfitting, where the network captures noise and minor details that are not relevant for broader pattern recognition tasks.

**General Observations**

Across all vigilance values, certain digits consistently form distinct clusters. For instance, digit '1' often forms a separate cluster even at low vigilance values due to its unique and simple structure. Conversely, digits with more complex or similar structures, like '4' and '7', tend to be grouped together at lower vigilance settings but separate at higher vigilance levels.

The confusion matrix heatmaps provide valuable insights into the clustering behavior of the ART1 network and highlight the importance of selecting an appropriate vigilance parameter based on the desired balance between generalization and specificity.

**Conclusion**

This project successfully implemented the Adaptive Resonance Theory (ART1) neural network model and applied it to the MNIST dataset, a standard benchmark for handwritten digit recognition. Through the experimentation with various vigilance parameters, we observed how the ART1 model's clustering behavior changes. Key findings include:

**Effect of Vigilance on Clustering:**

Lower vigilance values result in fewer, broader clusters, which are useful for general categorization but may lack specificity.

Higher vigilance values lead to a greater number of highly specific clusters, capturing fine details and variations in the input patterns.

Moderate vigilance values provide a balance between generalization and specificity, forming clusters that effectively differentiate between different digit classes without excessive fragmentation.

**Cluster Analysis:**

The analysis of confusion matrix heatmaps showed that certain digits consistently form distinct clusters even at low vigilance values, while visually similar digits tend to be grouped together at lower vigilance settings but separate at higher vigilance levels.

The ART1 network demonstrated its ability to adaptively form meaningful clusters, which correspond well with the actual digit classes in the MNIST dataset.

**Practical Implications:**

The choice of vigilance parameter is crucial and should be guided by the specific requirements of the application. For broad categorization, lower vigilance values are suitable, while higher vigilance values are better for tasks requiring precise pattern recognition and differentiation.

Overall, the ART1 model's performance in clustering binary input patterns from the MNIST dataset highlights its potential for various unsupervised learning applications, particularly in scenarios where maintaining a balance between stability and plasticity is essential.

**Future Work**

The current study provides a foundation for further exploration and enhancement of the ART1 neural network model. Future work could focus on the following areas:

**Scaling Up:**

Experiment with larger datasets to validate the scalability and robustness of the ART1 model.

Parameter Optimization:

Explore additional parameters and their combinations to fine-tune the ART1 model for improved clustering performance.

Extended Applications:

Apply the ART1 model to other datasets and domains to evaluate its generalizability and effectiveness in different contexts.

Hybrid Models:

Integrate ART1 with other neural network architectures or machine learning techniques to leverage their strengths and address any limitations.

Enhanced Visualization:

Develop more advanced visualization techniques to provide deeper insights into the clustering behavior and the relationships between clusters and input patterns.

By pursuing these directions, we can further enhance the capabilities and applications of the ART1 neural network model, contributing to advancements in unsupervised learning and neural network research.

**References**

Carpenter, G. A., & Grossberg, S. (1987). ART 2: Self-organization of stable category recognition codes for analog input patterns. Applied Optics, 26(23), 4919-4930.

Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. Cognitive Science, 11(1), 23-63.

Lippmann, R. P. (1987). An introduction to computing with neural nets. IEEE ASSP Magazine, 4(2), 4-22.

OpenML: https://www.openml.org/d/554

McClelland, J. L., Rumelhart, D. E., & Hinton, G. E. (1986). The appeal of parallel distributed processing. In Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1: Foundations (pp. 3-44). MIT Press.

**The Code**

import numpy as np

import logging

from sklearn.datasets import fetch\_openml

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

# Configure logging

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

class ART1:

def \_\_init\_\_(self, input\_size, rho):

self.input\_size = input\_size # Number of input neurons

self.rho = rho # Vigilance parameter

self.weights = [] # List to store weights for each cluster

logging.info(f'ART1 initialized with input size {input\_size} and vigilance parameter {rho}')

def train(self, inputs):

logging.info(f'Starting training with {len(inputs)} input vectors')

cluster\_assignments = []

# Initialize weights

n = self.input\_size

bottom\_up\_weights = np.ones((n, 0)) \* (1 / (1 + n)) # Initially no clusters

top\_down\_weights = np.ones((0, n)) # Initially no clusters

for i, input\_vector in enumerate(inputs):

logging.debug(f'Training on input vector {i}')

input\_vector = self.\_binarize(input\_vector)

# Present input pattern and calculate activations

activations = np.dot(bottom\_up\_weights.T, input\_vector)

# Active set A contains all nodes

A = list(range(len(activations)))

while A:

# Select node with largest activation

j = A[np.argmax(activations[A])]

# Compute s\*

s\_star = np.minimum(top\_down\_weights[j], input\_vector)

# Calculate similarity

similarity = np.sum(s\_star) / np.sum(input\_vector)

if similarity < self.rho:

# Remove j from A

A.remove(j)

else:

# Update weights

bottom\_up\_weights[:, j] = (top\_down\_weights[j] \* input\_vector) / (

0.5 + np.sum(top\_down\_weights[j] \* input\_vector))

top\_down\_weights[j] = np.minimum(top\_down\_weights[j], input\_vector)

# Assign pattern to cluster

cluster\_assignments.append(j)

break

if not A:

# Create new node

new\_top\_down\_weight = input\_vector

new\_bottom\_up\_weight = input\_vector / (0.5 + np.sum(input\_vector))

top\_down\_weights = np.vstack([top\_down\_weights, new\_top\_down\_weight])

bottom\_up\_weights = np.column\_stack([bottom\_up\_weights, new\_bottom\_up\_weight])

# Assign pattern to new cluster

cluster\_assignments.append(len(top\_down\_weights) - 1)

logging.info('Training completed')

self.bottom\_up\_weights = bottom\_up\_weights

self.top\_down\_weights = top\_down\_weights

return cluster\_assignments

def \_binarize(self, vector):

logging.debug('Binarizing input vector')

return np.where(vector > 0.5, 1, 0)

def analyze\_clusters(clusters, labels, rho):

# Create a DataFrame for easier manipulation

df = pd.DataFrame({'Cluster': clusters, 'Label': labels})

cluster\_distribution = df.groupby(['Cluster', 'Label']).size().unstack(fill\_value=0)

# Print the number of clusters

num\_clusters = cluster\_distribution.shape[0]

print(f'Number of clusters for vigilance {rho}: {num\_clusters}')

# Plot the cluster distribution heatmap

plot\_cluster\_heatmap(cluster\_distribution, rho)

return num\_clusters # Return the number of clusters

def plot\_cluster\_heatmap(cluster\_distribution, rho):

plt.figure(figsize=(20, 16))

sns.heatmap(cluster\_distribution, annot=True, cmap='viridis', cbar=True)

plt.title(f'Cluster Distribution Heatmap for vigilance {rho}', fontsize=20)

plt.xlabel('Digit Class', fontsize=16)

plt.ylabel('Cluster', fontsize=16)

plt.xticks(fontsize=12)

plt.yticks(fontsize=12)

plt.show()

# Function to train ART1 and analyze clusters for different vigilance values

def experiment\_vigilance\_values(vigilance\_values, inputs, labels):

results = []

for rho in vigilance\_values:

logging.info(f'Experimenting with vigilance parameter: {rho}')

art1 = ART1(input\_size=784, rho=rho)

cluster\_assignments = art1.train(inputs)

print(f'Results for vigilance parameter: {rho}')

num\_clusters = analyze\_clusters(cluster\_assignments, labels, rho)

results.append((rho, num\_clusters))

print('-----------------------------------------')

return results

# Load MNIST dataset

mnist = fetch\_openml('mnist\_784', version=1)

images = mnist.data.values

labels = mnist.target.astype(int)

# Normalize images

images = images / 255.0

# Binarize images

threshold = 0.5

binarized\_images = (images > threshold).astype(int)

# Train ART1 network on a subset of the dataset

train\_subset = binarized\_images[:1000]

true\_labels\_subset = labels[:1000]

# List of vigilance values to experiment with

vigilance\_values = [0.02, 0.05, 0.1, 0.5, 0.8, 0.9]

# Train and analyze for different vigilance values

results = experiment\_vigilance\_values(vigilance\_values, train\_subset, true\_labels\_subset)

# Plot vigilance vs. number of clusters

vigilance, num\_clusters = zip(\*results)

plt.figure(figsize=(10, 6))

plt.plot(vigilance, num\_clusters, marker='o')

plt.title('Vigilance vs. Number of Clusters', fontsize=20)

plt.xlabel('Vigilance', fontsize=16)

plt.ylabel('Number of Clusters', fontsize=16)

plt.grid(True)

plt.show()