ART-1 MODEL:

The ART-based clustering algorithm adaptively and continually generates prototype nodes corresponding to given data, and the generated nodes are used as classifiers. The label probability computation independently counts the number of label appearances for each class and calculates the Bayesian probabilities. Thus, the label probability computation can cope with an increase in the number of labels. Experimental results with synthetic and real-world multi-label datasets show that the proposed algorithm has competitive classification performance to other well-known algorithms while realizing continual learning.  
Adaptive Resonance Theory (ART) is considered as an effective approach for realizing continual learning thanks to its ability to handle the plasticity-stability dilemma. In general, however, the clustering performance of ART-based algorithms strongly depends on the specification of a similarity threshold, i.e., a vigilance parameter, which is data-dependent and specified by hand. This paper proposes an ART-based topological clustering algorithm with a mechanism that automatically estimates a similarity threshold from the distribution of data points. In addition, for improving information extraction performance, a divisive hierarchical clustering algorithm capable of continual learning is proposed by introducing a hierarchical structure to the proposed algorithm. Experimental results demonstrate that the proposed algorithm has high clustering performance comparable with recently proposed state-of-the-art hierarchical clustering algorithms.

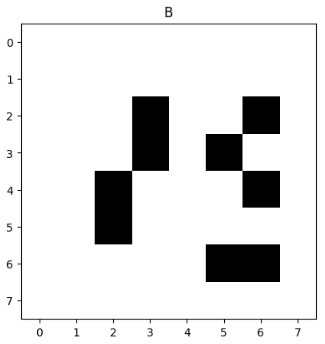
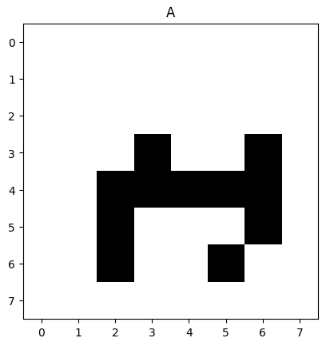
For more information you can visit this site: <https://arxiv.org/pdf/2103.01511v3.pdf>

Problems:

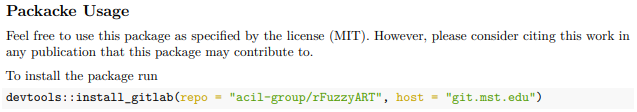
I used patterns of alphabets 28x28 instead of 8×8 as I tried making the patterns of 8×8 but downscaling them to 8×8 almost destroyed the patterns and made training unable.

Sample of 8\*8:

All the images lost data like these ones, so we stick with the lowest dimension we can use which are 28\*28.

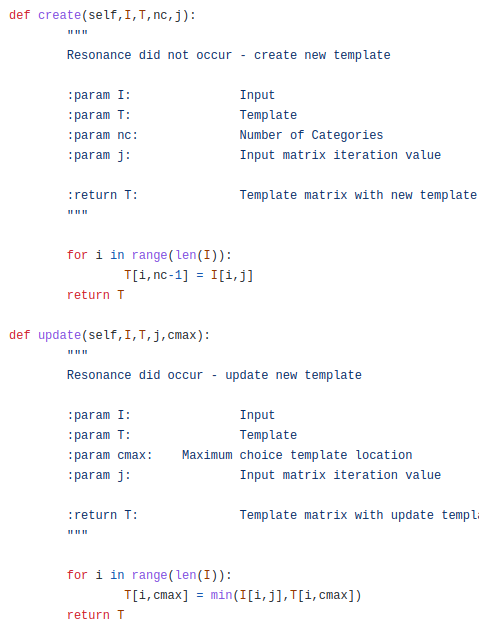


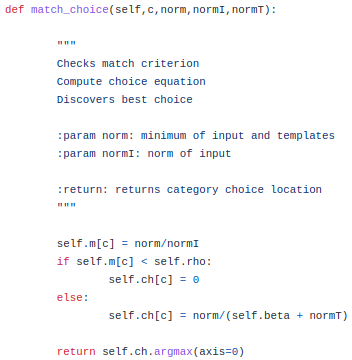
HELPING CODE MATERIAL:

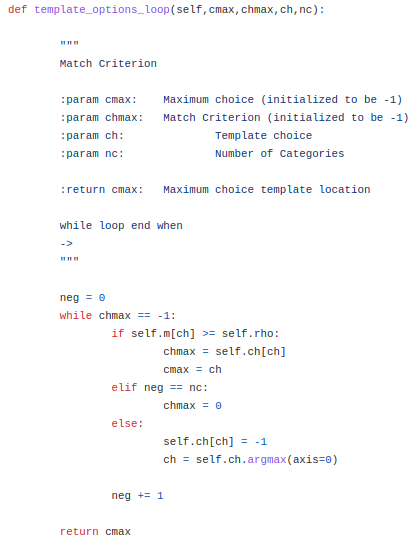


The package or helping code that we used to build, train, and test our **ART Model.**

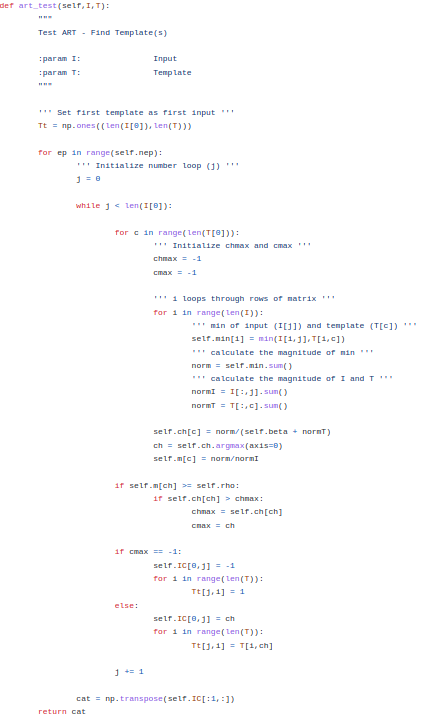
Training:

.





Testing:



**Algorithm:**

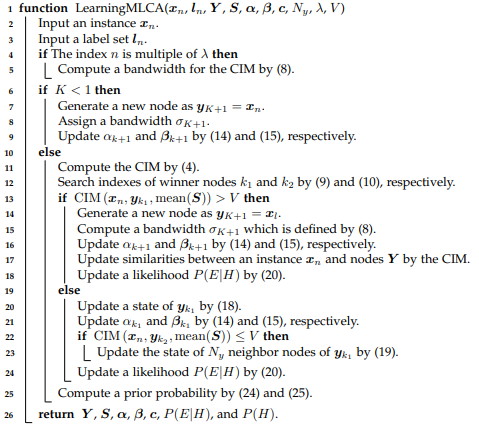
Clustering:

Cluster analysis is one of the widely applied approaches to extract hidden relation from data. Typical types of clustering algorithms are the Gaussian mixture model, k-means, and Self-Organizing Map (SOM). Although the above algorithms are quite simple and highly adaptable, the number of classes and network architectures are specified in advance. Growing Neural Gas (GNG) and Self Organizing Incremental Neural Network (SOINN) are well-known growing self-organizing clustering algorithms that can overcome the drawbacks of the typical types of clustering algorithms. GNG and SOINN can adaptively generate topological networks corresponding to given data. However, since these algorithms permanently insert new nodes into their networks for memorizing new knowledge, they have a potential to forget learned knowledge (i.e., catastrophic forgetting). This trade-off is called the plasticity-stability dilemma. A variant of GNG, called Grow When Required (GWR) can avoid the plasticity stability dilemma by adding nodes whenever the state of the current network does not sufficiently match the instance. One problem of GWR is that as the number of nodes in the network increases, the cost of calculating a threshold for each node increases, and thus the learning efficiency decreases.

Multi-label Classification:

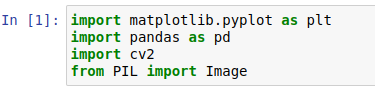
The multi-label classification algorithms are categorized into two approaches, namely, a problem transformation approach and an algorithm adaptation approach. The problem transformation approach transforms a multi-label classification problem into multiple single-label classification problems. The problem transformation approach is further divided into two methods, namely, the Binary Relevance (BR) and the Label Powerset (LP). The BR transforms a multi-label classification problem into multiple binary classification problems by decomposing multi-labels into multiple single labels. The LP transforms a multilabel classification problem into a multi-class classification problem by merging multi-labels into a single label. Various single-label classification algorithms have been used in the problem transformation approach thanks to their simplicity and applicability.

ART-1 in ACTION:



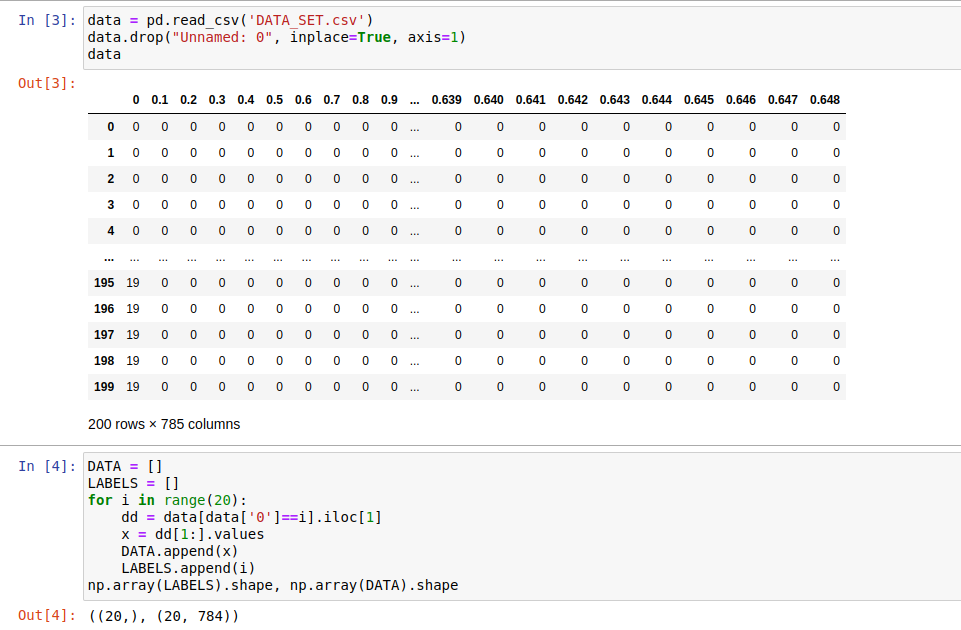
CODE EXPLAINATION:

Libraries:



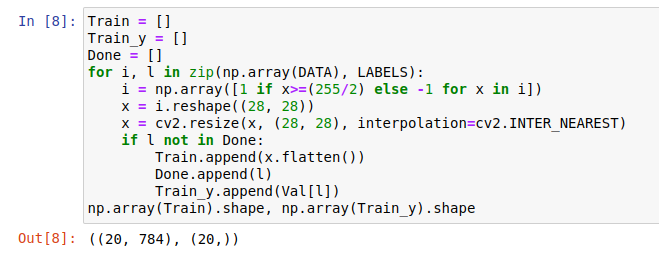
matplotlib.pyplot is used for creating data visualizations in Python. Pandas to read data and save it as data frames and series. CV2 is used for reshaping and resizing images of the DATASET for image processing, such as image filtering, object detection, and image segmentation.

Dataset:



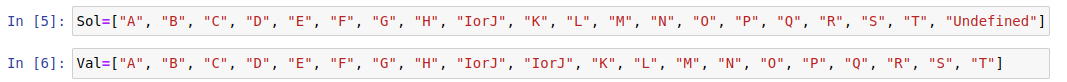
Dataset is downloaded from “Kaggle,” and it is filtered to get only one image of the alphabets from “A-T.”

Data and Lables:



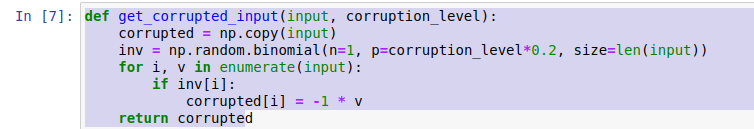
This code is preparing training data. The main loops pass over each value in DATA and LABELS. Then converts each pixel value into **1 or -1** based on a threshold of **255/2**. This is an effortless way to normalize the input values and convert them to a binary format. After that reshapes the 1D array of pixel values into a 2D array with dimensions 28x28 to make it an Image. Every single data is saved, and their Labels are also saved to evaluate the model.

VALUES:



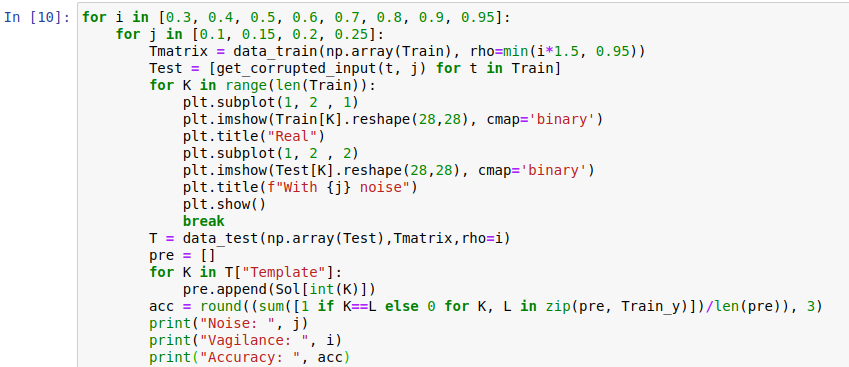
Here the values of Training and Testing are saved. The model can predict every value that was given to it but also another value “undefined” represented as “-1” so we have “undefined” as last value in the solution. The “Val” represents the values of the original Label.

Noise Function:



This is a function that takes an input array and a corruption level and returns a corrupted version of the input array. This function creates a copy of the input to ensure that the original input array is not modified and generates a binary mask of the same length as the input array. Each element in the mask has a probability of being 1, which is determined by the **corruption level** parameter. If the binary mask has a 1 value, then the function sets the value of the current element in the corrupted array to its negative value. This flips the sign of the element, effectively corrupting it.

MAIN FUNCTION:

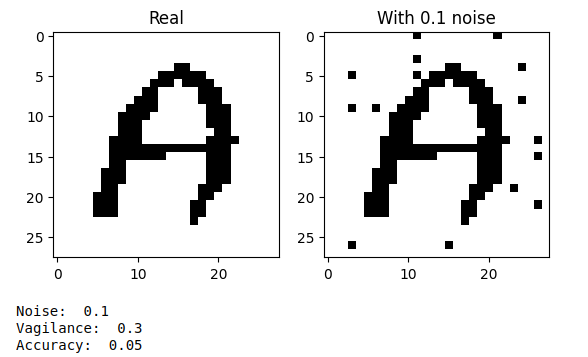


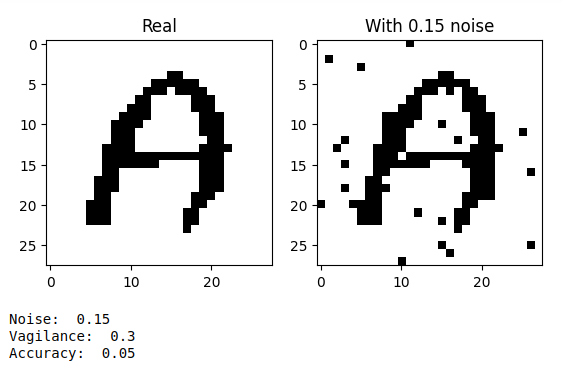
In this function we nested loops to set values of “**rho**” and **“noise”** parameter and prints the value of the accuracy and the noise level. Firstly, generate a transformation matrix by training on data with the given **rho** parameter. This matrix is used to transform the input data into a new representation that is more amenable to clustering. Then get the corrupted inputs by applying the “**get\_corrupted\_input”** function to each input in the training set with the given **noise level**. After that Display, **original** vs **corrupted** input.

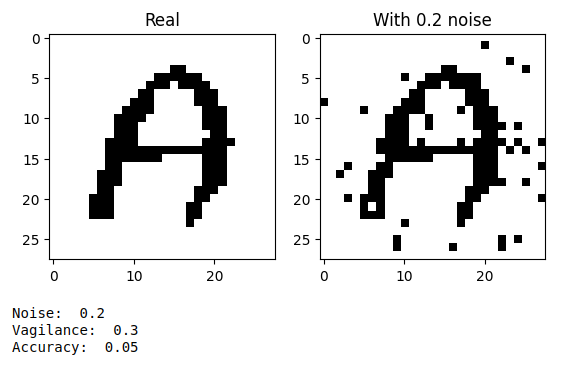
**Classify** each **noisy image then** get their value by checking the return labels by **prediction** function into **“pre” by “Sol.”**

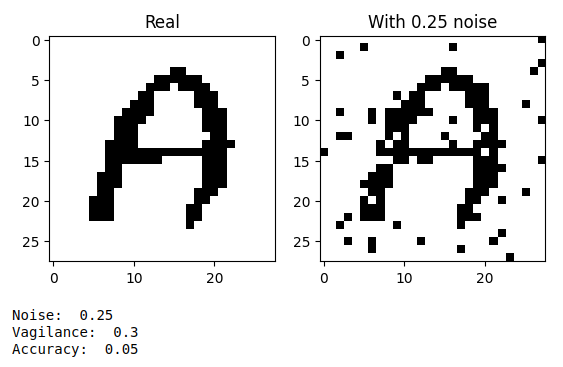
Lastly, compute the **accuracy** of the **classification** by **comparing the predicted labels to the true labels.**

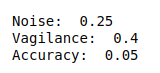
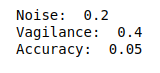
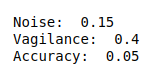
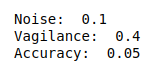
OUTPUT:

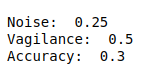
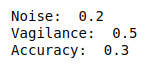
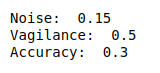
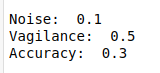


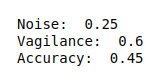
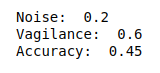
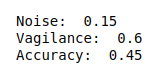
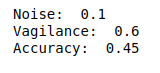


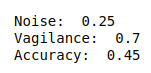
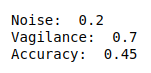
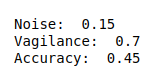
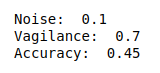


















This was the output of **main Function.**

References:

**The Training and Testing helping code was gathered from “GitHub.”**

**References:**

1. [**https://github.com/cbirkj/art-python.git**](https://github.com/cbirkj/art-python.git)
2. [**https://github.com/masuyama-lab/mlca**](https://github.com/masuyama-lab/mlca)
3. [**https://arxiv.org/pdf/2103.01511v3.pdf**](https://arxiv.org/pdf/2103.01511v3.pdf)

**This repo is like an ART network library.**