Discovering a Zero
(Zero-Vector Class of Machine Learning)

Harikrishna Metta [1] **R. Venkatesh Babu** [1]

Abstract

In Machine learning, separating data into classes is a very fundamental problem. A mathematical framework around the classes is presented in this work to deepen the understanding of classes. The classes are defined as vectors in a Vector Space, where addition corresponds to the union of classes, and scalar multiplication resembles set complement of classes. The Zero-Vector in the vector space corresponds to a class referred to as the Metta-Class. This discovery enables numerous applications. One such application, termed 'clear learning' in this work, focuses on learning the true nature (manifold) of the data instead of merely learning a boundary sufficient for classification. Another application, called 'unary class learning', involves learning a single class in isolation rather than learning by comparing two or more classes. Additionally, 'set operations on classes' is another application highlighted in this work. Furthermore, Continual Learning of classes is facilitated by smaller networks. The Metta-Class enables neural networks to learn only the data manifold; therefore, it can also be used for generation of new data. Results for the key applications are shown using the MNIST dataset. To further strengthen the claims, some results are also produced using the CIFAR-10 and ImageNet-1k embeddings. The code supporting these applications is publicly available at: `github.com/hm-4/Metta-Class`.

[1]Vision and AI Lab (val.cds.iisc.ac.in), Department of Computational and Data Sciences, Indian Institute of Science, Bangalore, INDIA. Correspondence to: Harikrishna Metta <hm.iisc.iitb@gmail.com>, R. Venkatesh Babu <venky@iisc.ac.in>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

1. Introduction

There are many techniques in Machine learning that allow data classification, one such contraption is the Neural Network. A general technique used in the data classification is the following. Find a separating hyper-surface between classes such that functional value of data points belong to same class when substituted in the hyper-surface equations resulting in a 'similar' sets of values. In Neural Networks those set of values are called logits. The general technique, with respect to Neural Networks, can be understood by thinking the logit equations of a Neural Network as hyper-surface equations, those logit values are passed though the Sigmoid or Softmax layer. The similarity of the logits of data points belonging to the same class is enforced by the cross entropy loss.

An example of decision boundaries of a neural network is shown in the far-left subfigure of Figure 1. The feature space in that subfigure is divided into two decision regions: the lower half is classified as Class-2, indicated by the green region, while the upper half is classified as Class-1, represented by the red region. However, this dichotomization of the entire feature space into these two classes is not accurate, as only a small portion of the space genuinely belongs to Class-1 or Class-2. Therefore, the decision regions depicted in the far-left subfigure of Figure 1 do not accurately represent the true nature of the classes. A more realistic representation of the decision regions can be seen in the mid-subfigure of Figure 1.

If a neural network exhibits the decision regions as shown in the mid-subfigure of Figure 1, these regions have particularly notable set properties. One such property is the union of the regions: the union of the decision regions accurately represents the true nature of the combined class's decision region. Another property is the complement of the decision region: as illustrated in the far-right subfigure of Figure 1, the complement of a decision region accurately represents the true complement of the class. These properties are essential, as they reflect the true characteristics of the classes. Consequently, neural networks with such well-defined decision regions for each class are desirable.

In this work, classes are characterized in terms of the logit

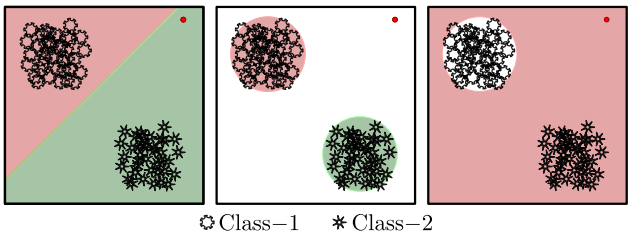


Figure 1. The far-left subfigure represents the decision regions of a neural network, learned by separating the feature space. The middle subfigure shows the actual regions corresponding to the classes, while the far-right subfigure illustrates the complement region of Class-1.

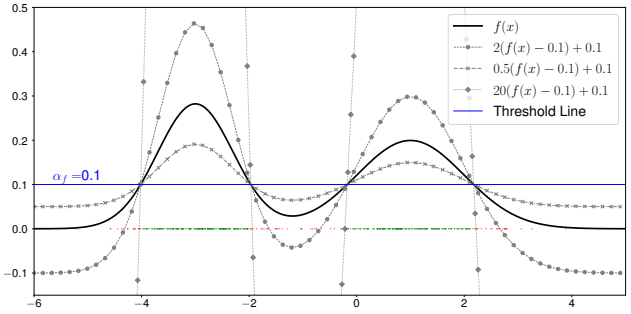


Figure 2. A set of functions that classify Class-f ($f(x)$) with the exact same decision regions for a threshold of $\alpha_f = 0.1$.

equations, and logit equations are viewed as vectors. Addition and scalar multiplication on such vectors is defined such that a vector space is created. As a consequence, it allows to use all vector operations on logits hence enables set operations on classes. The relevant proofs and meanings of the operations are provided, without going into very rigorous math. Sample PyTorch implementations and results are attached supporting the theory when needed.

2. Characterization of Logits

If the probability density function (PDF) of the data, $f_n(x)$, is known, a given data point can be assessed for membership in the distribution by comparing its probability to a predefined threshold, α_{f_n}. The threshold α_{f_n} should be chosen as a non-negative real number, ensuring that data points with high relative probability are recognized as belonging to the class. As illustrated in Figure 2, data points with probabilities ($f_n(x)$) greater than or equal to $\alpha_{f_n} = 0.1$ are classified as belonging to Class-f, while those with lower probabilities are classified as not belonging to the class. The PDF function $f_n(x)$ itself is used for this classification along with the threshold α_{f_n}.

In cases where only the unnormalized PDF $f(x)$ is known, it is still possible to classify data points by comparing $f(x)$ to the scaled threshold $\alpha_f A_f$, where A_f represents the normalization constant of $f(x)$. A diagram illustrating this mechanism can be seen in Figure 2, where $A_f = 1$.

The value $f(x_0)$ for a given data point x_0 is referred to as the 'Logit.' The function $f(x)$ is defined as the 'Logit Function' or 'Logit Equation.' Any Logit Equation that correctly classifies data with respect to a predefined threshold α_f is termed a 'Valid Logit Equation (VLE)' or 'Valid Logit Function.' These definitions will be used consistently throughout this paper.

All Valid Logit Functions can be grouped into a set, as shown in equation 1. Any function $h(x)$ that belongs to the set $[f(x)]$ in equation 1 can classify data sampled from the unnormalized distribution $f(x)$. Notice that functions of the form $h(x) + k$, where $h(x) \in [f(x)]$ and $k \in \mathbb{R}$, with a threshold $\alpha_f + k$ can perform the same classification task as $h(x)$ and α_f. Therefore, to accommodate equations of the form $h(x) + k$ and thresholds of the form $\alpha_f + k$, as well as any other valid pair of functions and thresholds, a separate set S is introduced in equation 1.

$$[f(x)] = \left\{ \beta\big(f(x) - \alpha_f A_f\big) + \alpha_f \;\middle|\; \beta \in \mathbb{R}^+ \right\} \cup S \quad (1)$$

Here, $f(x)$ represents the unnormalized PDF of the class, $\alpha_f \in \mathbb{R}$ is the predefined threshold for $f(x)$, A_f represents the normalization constant of $f(x)$, and S denotes the set of unknown (does not need to be an unnormalized or a normalized PDF) Valid Logit Functions associated with that class.

The set $[f(x)]$ in equation 1 is an equivalence class with respect to the classification of the data, as it satisfies the properties of an equivalence relation. Specifically, any equation within this set classifies the data in exactly the same way as others in the set for a predefined threshold. Consequently, the set $[f(x)]$ is referred to as the 'Valid Equivalence Set' of $f(x)$. This terminology will be consistently used throughout this paper.

The following subsection explores the resulting Valid Equivalence Set when two classes are combined.

2.1. Merging two Classes

Now consider another class, Class-2, whose unnormalized PDF is $g(x)$. Figure 3 illustrates some Valid Logit Functions for Class$-g$, similar to those shown for Class$-f$ in Figure 2. Any function $k(x)$ belonging to the set $[g(x)]$, as defined in Equation (2), is a Valid Logit Function capable of classifying data sampled from the unnormalized distribution $g(x)$, in the same way as was previously done for Class$-f$.

$$[g(x)] = \left\{ \beta\big(g(x) - \alpha_g A_g\big) + \alpha_g \;\middle|\; \beta \in \mathbb{R}^+ \right\} \cup T \quad (2)$$

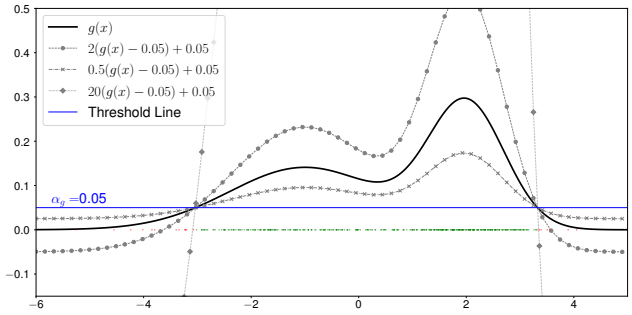


Figure 3. A set of functions that classify Class-g ($g(x)$) with the exact same decision regions for a threshold of $\alpha_g = 0.05$.

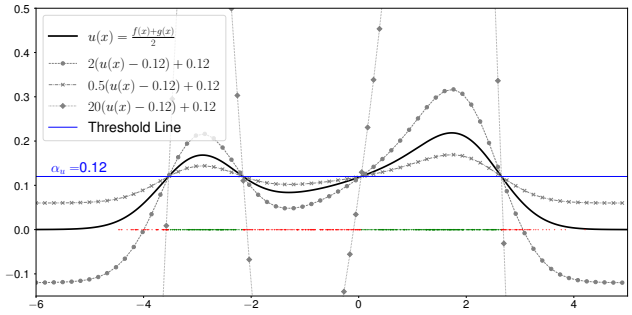


Figure 4. A set of functions that classify combined Class-u (Class$-u$ = Class$-f \cup$ Class$-g$) with the exact same decision regions for a threshold of $\alpha_u = 0.12$.

Here, $g(x)$ represents the unnormalized PDF of Class$-g$, $\alpha_g \in \mathbb{R}$ is the predefined threshold for $g(x)$, A_g represents the normalization constant of $g(x)$, and T denotes the set of unknown (does not need to be an unnormalized or a normalized PDF) Valid Logit Functions associated with Class$-g$.

Suppose there is a need to merge the data of Class$-g$ with the data of Class$-f$. The resulting PDF of the combined class, denoted as $u(x)$, is illustrated in Figure 4. Assuming an equal number of data points in Class$-f$ and Class$-g$, the equation for $u(x)$ in terms of PDFs $f_n(x)$ and $g_n(x)$, the normalized distributions of $f(x)$ and $g(x)$ respectively, is given in Equation (3). The proof of this equation is provided in Section A of the Appendix.

$$u(x) = \frac{f_n(x) + g_n(x)}{2} \tag{3}$$

Similar to what was observed for Class$-f$ and Class$-g$, there can be multiple Valid Logit Functions corresponding to the combined PDF $u(x)$, as illustrated in Figure 4. The set $[u(x)]$, as defined in Equation (4), represents all Valid Logit Functions associated with the PDF $u(x)$.

$$[u(x)] = \left\{ \beta\big(u(x) - \alpha_u A_u\big) + \alpha_u \;\middle|\; \beta \in \mathbb{R}^+ \right\} \cup Z \tag{4}$$

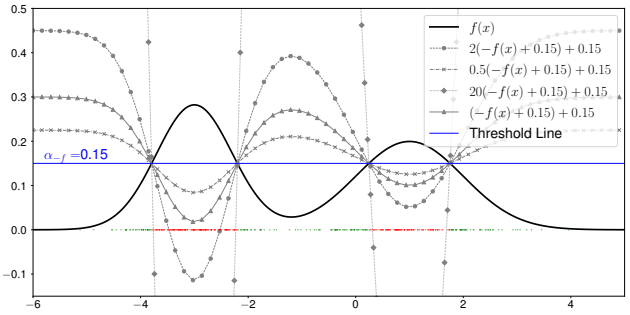


Figure 5. A set of functions that classify complement of Class-f with the exact same decision regions for a threshold of $\alpha_{-f} = 0.15$.

Here, $u(x)$ represents the PDF of the combined distribution, $\alpha_u \in \mathbb{R}$ is a predefined threshold for PDF $u(x)$, A_u represents the normalization constant of $u(x)$, and Z denotes the set of unknown (does not need to be an unnormalized or a normalized PDF) Valid Logit Functions associated with the combined distribution $u(x)$.

An important question arises: given the logit equations $h(x) \in [f(x)]$ and $k(x) \in [g(x)]$, is it possible to construct a new logit equation $v(x) \in [u(x)]$ for the combined distribution? If achievable, $v(x)$ could be utilized for classification of the combined distribution with PDF $u(x)$. However, this is not feasible, as numerous counterexamples can be identified. A related question then follows: can the sets $[f(x)]$ and $[g(x)]$ themselves be combined to derive a new set $[u(x)]$, as shown in Equation (5).

$$[f(x)] \cup [g(x)] = [u(x)] \tag{5}$$

$$[f(x)] + [g(x)] = \left[\frac{f_n(x) + g_n(x)}{2}\right] \tag{6}$$

$$[f(x)] + [g(x)] = [f_n(x) + g_n(x)] \tag{7}$$

Notice that the sets $[f_n(x) + g_n(x)]$ and $\left[\frac{f_n(x)+g_n(x)}{2}\right]$ are exactly the same. The '$+$' operator in the left-hand side of Equation (6) is used as a symbol to represent union. However, the '$+$' operator on the right-hand side of the same equation denotes the standard addition operator. This choice is intentional, as the equation is later adopted as a definition of addition in the following section.

2.2. Complement of a Class

There is an observation on the Valid Equivalence Set $[-f(x)]$: it represents the complement of Class$-f$, as illustrated in Figure 5 for the same predefined threshold α_f.

Mathematically, this is expressed as in Equation (8).

$$[f(x)]^c = [-f(x)] \tag{8}$$
$$-[f(x)] = [-f(x)] \tag{9}$$

The '$-$' operator on the left-hand side of Equation (9) is used as a symbol to represent the complement operation. However, the '$-$' operator on the right-hand side of the same equation denote standard subtraction operation. This choice is intentional, as the equation is later adopted as a definition in the following section.

3. The Vector Space

As seen in the previous chapter, the set of Valid Logit Equations of a class, whose PDF is $f(x)$, is called the Equivalence set and represented by the $[f(x)]$. Therefore the function space is divided into set of these equivalence sets each representing a class.

In this chapter, all the properties of a vector space (Axler, 2015) are verified for that set of equivalence sets, using the definitions of addition 3.2.1 and scalar multiplication 3.2.2. During this process a zero is discovered for this vector space, this zero represents an equivalence set. As previously seen each such equivalence set represents a data class. The data class represented by the zero element of the vector space is referred to as the Metta-Class (also called the Zero-Vector Class).

3.1. Definition of set V

Let V be the set of equivalence sets. Each equivalence set also known as Valid Logit Function set represents a data class. If the PDF of a data-class is $f(x)$ then the corresponding equivalence set is denoted by $[f(x)]$. Therefore the set V can be written as

$$V = \{\ [f(x)],\ \ [g(x)],\ \ [h(x)],\ \ \ldots.\ \} \tag{10}$$

Throughout this report, whenever the vector space V is mentioned, it refers to this vector space.

3.2. Definition of Addition and Scalar multiplication

As understood in the previous chapter, the equations for union (7) and complement (9) are taken as definitions here with slight modifications so that they are suitable for defining a vector space. Specifically,

3.2.1. Addition

For all $[f(x)],\ \ [g(x)] \in V$, the sum $[f(x)] + [g(x)]$ is defined as

$$[f(x)] + [g(x)] := [f(x) + g(x)] \tag{11}$$

3.2.2. Scalar multiplication

For all $\lambda \in \mathbb{R}$ and for all $[f(x)] \in V$ the product $\lambda[f(x)]$ is defined as

$$\lambda\ [f(x)] :=\ [\lambda\ f(x)] \tag{12}$$

3.3. Verification of the Properties of vector space on set V, along with addition and scalar multiplication

3.3.1. Commutativity

For all $[f(x)],\ [g(x)] \in V$ using the property of addition

$$[f(x)] + [g(x)] = [f(x) + g(x)] = [g(x)] + [f(x)]$$

Hence the commutativity property of vector space holds on set V.

3.3.2. Associativity

For all $[f(x)],\ [g(x)],\ [h(x)] \in V$, using the property of addition

$$\begin{aligned}\Big([f(x)] + [g(x)]\Big) + [h(x)]\ &= \Big([f(x) + g(x)]\Big) + [h(x)] \\ &= [f(x)] + \Big([g(x) + h(x)]\Big)\end{aligned}$$

let $a,\ b \in \mathbb{R}$, then using the property of scalar multiplication

$$\begin{aligned}(ab)[f(x)]\ &= a\Big(b[f(x)]\Big) \\ &= a\Big([bf(x)]\Big)\end{aligned}$$

Hence the associativity property of vector space holds on set V.

3.3.3. Additive Identity (The Zero)

For all $[f(x)] \in V$ there exists 0 such that

$$\begin{aligned}[f(x)] + [0] &= [f(x) + 0] \\ &= [f(x)]\end{aligned}$$

Hence there exists an additive identity on set V.

3.3.4. Additive Inverse

For every $[f(x)] \in V$ there exists $[-f(x)]$ such that

$$\begin{aligned}[f(x)] + [-f(x)] &= [f(x) + -f(x)] \\ &= [0]\end{aligned}$$

Hence there exists additive inverse on set V.

3.3.5. Multiplicative Identity

For every $[f(x)] \in V$, $1[f(x)] = [f(x)]$, hence the multiplicative identity exists.

3.3.6. DISTRIBUTIVE PROPERTIES

For all $a,\ b \in \mathbb{R}$ and for all $[f(x)],\ [g(x)] \in V$

$$\begin{aligned} a([f(x)] + [g(x)]) &= a[f(x)] + a[g(x)] \\ &= [af(x)] + [ag(x)] \\ (a+b)[f(x)] &= (a[f(x)] + b[f(x)]) \\ &= [af(x)] + [bf(x)] \end{aligned}$$

Hence the distributive properties of vector space holds on set V.

Therefore, the defined set V forms a vector space under the specified operations of addition and scalar multiplication. The equivalence sets are represented as vectors in this space and are referred to as **class-vectors**, with each vector corresponding to a distinct data class. In particular, the zero vector ($[0]$) in this space might also represents a data class, referred to as the **Zero-Vector Class**.

To avoid ambiguity between the concept of the Zero-Vector Class and the label "zero" used for other classes, this class is hereafter referred to as the **Metta-Class**.

4. Metta-Class and its Applications

As established in the previous chapter, the set V defined in Section 3.1, together with the addition and scalar multiplication operations described in Subsections 3.2.1 and 3.2.2, forms a vector space. Each vector in the vector space V represents a data class. The additive identity vector $[0]$ of this space (see Subsection 3.3.3) must therefore also represent a data class. In the previous chapter, the class corresponding to this additive identity vector was named the **Metta-Class**.

Using the equation 1, the set $[0]$ can be written as

$$[0] = \left\{ \beta\big(0 - \alpha_0 A_0\big) + \alpha_0 \;\middle|\; \beta \in \mathbb{R}^+ \right\} \tag{13}$$

Here, α_0 is a threshold for PDF of the class $[0]$, A_0 represents the normalization constant of the PDF corresponding to set $[0]$.

if we define normalizing constant A_f of unnormalized PDF $f(x)$ as

$$A_f := \lim_{\mathcal{V}\to[-\infty,\infty]^d} \int_{\mathcal{V}} f(x)\, dx \tag{14}$$

Simple substitution $f(x) = 0$ in Equation (14) gives $A_0 = 0$ hence

$$[0] = \{\alpha_0\} \tag{15}$$

where $\alpha_0 \in \mathbb{R}$. Therefore, the logit equation of the Metta-Class is a constant valued function.

If $f(x) = 0$ is treated as a constant going to zero in the limit i.e., $f(x) = \lim_{k\to 0} k$ then equation 14 becomes

$$A_0 = \lim_{\mathcal{V}\to[-\infty,\infty]^d} \int_{\mathcal{V}} \lim_{k\to 0} k\, dx \tag{16}$$

$$= \lim_{\mathcal{V}\to[-\infty,\infty]^d} \lim_{k\to 0} \int_{\mathcal{V}} k\, dx \tag{17}$$

$$= \lim_{\mathcal{V}\to[-\infty,\infty]^d} \lim_{k\to 0} \mathcal{V}k\, dx \tag{18}$$

Therefore PDF corresponding to $[0]$ can be written as:

$$\text{PDF}([0]) = \lim_{\mathcal{V}\to[-\infty,\infty]^d} \lim_{k\to 0} \frac{k}{\mathcal{V}k} \tag{19}$$

$$= \lim_{\mathcal{V}\to[-\infty,\infty]^d} \frac{1}{\mathcal{V}} \tag{20}$$

Substituting Equation (18) into set (13) results in a set of constant values. From Equations (15) and (20), the PDF of the Metta-Class appears to follow a uniform distribution. This observation serves as an indication rather than a rigorous proof. If we consider the PDF of the Zero-Vector as a uniform distribution, and since it also satisfies the additive identity property, then in a peculiar sense, having data uniformly distributed across the entire space is equivalent to having no data at all. For a limited volume $\mathcal{V}$, PDF corresponding to $[0]$ is a uniform distribution with magnitude $\frac{1}{\mathcal{V}}$. Intuitively, if a threshold α_f is used for the PDF $f(x)$, then for $f(x) + \frac{1}{\mathcal{V}}$, using a threshold of $\alpha_f + \frac{1}{\mathcal{V}}$ results in the exact same decision regions; therefore, the uniform distribution can be seen as the additive identity of the vector space. For the remainder of the report, the PDF of $[0]$ is considered as uniform distribution. For an alternative explanation, see Section F in the Appendix.

4.1. Generating Metta-Class data in finite volume of the feature space.

The data of any natural class generally spans a small volume in the feature space, as shown in the top-left plot of Figure 6. Within this small volume, data for the Metta-Class can be generated from a uniform distribution spanning the same volume. The generated data for the Metta-Class is illustrated in the top-right plot of Figure 6.

4.2. Applications of Metta-Class

4.2.1. CLEAR LEARNING

A neural network called Zero-Inclusive-Network is trained on the data shown in top-right subfigure of the figure 6, which includes the Metta-Class data. For comparison, another neural network called Zero-Exclusive-Network is trained on the data excluding the Metta-Class data i.e., the data in the top-left plot of the Figure 6. All front layers in both networks are identical except for the last layer. In the last layer of Zero-Inclusive-Network, an additional node is used for the Metta-Class.

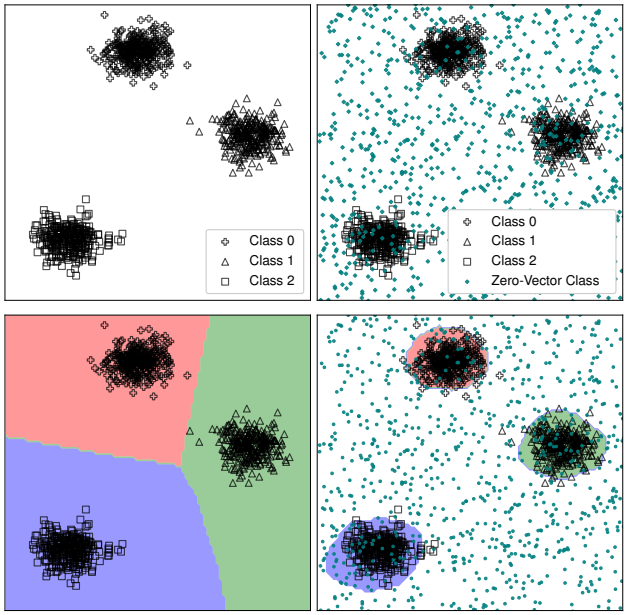


Figure 6. The top-left plot shows the data of three different classes, while the bottom-left plot depicts the decision regions learned by a Neural Network for these classes. The top-right plot includes the three classes along with the Metta-Class, and the bottom-right plot illustrates the decision regions learned by a Neural Network for these four classes.

The differences in the learning outcomes of both networks are illustrated in the bottom plots of Figure 6. The bottom-left plot corresponds to the Zero-Exclusive-Network, while the bottom-right plot represents the Zero-Inclusive-Network. It can be observed that the boundaries learned by the Zero-Inclusive-Network are superior and clearer. By 'clear,' it is meant that decision regions of the Zero-Inclusive-Network overlaps exactly with the data. For instance, in the bottom-left plot of Figure 6, which illustrates the learning process of the Zero-Exclusive-Network, a significant portion of the empty space is misclassified as Class-0, Class-1, or Class-2.

4.2.2. Unary/Uni-class classification

Imagine a scenario where all the collected data belongs to a single class. Is it still possible to train a neural network in such a case? Moreover, can the trained network determine whether a new data point belongs to this class or not? In neural networks, generally, the learning is done by comparing one class against another class(es). If the data belongs to only one class, say Class-0, the learning does not make much sense. This is where the Metta-Class is helpful. The data for the Metta-Class is a set of samples from uniform distribution. Now the original class-0 can be classified against the Metta-Class data with BCE loss using a neural network. The learning of the Neural Network trained on the combined data can be seen in the right subfigure of Figure 7.

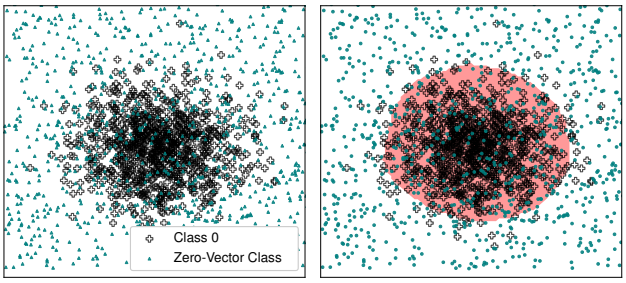


Figure 7. The left plot shows the data of Class-0 and the Metta-Class, while the right plot depicts the decision region of a Neural Network trained on these two classes using BCE loss. This illustrates how a Neural Network can learn a single class.

Therefore learning is possible even with only one class data, which is desirable since it is preferable to use thousands of smaller neural networks for training rather than training one giant neural network for a thousand classes. Results for MNIST, CIFAR10 and ImageNet data are added in the Results Section E.1.2 of Appendix.

4.2.3. Set operations on the classes

Upon closer examination, bottom-right plot of the Figure 6 exhibits resemblance to Venn Diagrams. This resemblance is no coincidence, as the addition 3.2.1 and scalar multiplication (with scalar value '-1') 3.2.2 defined on the Vector Space have the characteristics of set union and set complement respectively. As the set union (boolean addition) and set complement (boolean not) are the fundamental operations on sets (boolean logic), any composite set operation (composite boolean operation) can be written in terms of set union (boolean addition) and set complement (boolean not). The table 1 lists a few of the set operations on classes.

Consider two classes, Class-1 and Class-2, trained on two different networks (uni-class classifiers) as described in Subsection 4.2.2. Let I_{c_1} and I_{c_2} denote the indicators (network outputs) for Class-1 and Class-2, respectively. The indicator values are 1 if the data belongs to the class they represent and 0 otherwise. Table 1 illustrates the logical operations on these classes. Furthermore, any boolean expression involving the classes can be evaluated using these logical operations. These logical operations can also be performed on multiple outputs of a single network. For instance, logical operations can be applied to the classes of the classifier corresponding to the bottom-right plot of Figure 6.

4.2.4. Continual learning

In general, when the Metta-Class is not used to train neural network, every time a new class is added to the network, the network must be retrained or adopt some complicated techniques. For example say network-1 classifies all fe-

Table 1. Boolean operations between Class-c_1 and Class-c_2, where I_{c_1} and I_{c_2} are indicators derived from classifiers trained including the Metta-Class. $I_{c_1} = 1$ implies the data point belongs to Class-c_1, while $I_{c_1} = 0$ implies it does not. For example, the fifth column represents the intersection operation, indicating whether a data point belongs to both Class-c_1 and Class-c_2, demonstrating that a separate classifier is not needed to identify data belonging to both classes.

I_{c_1}	I_{c_2}	$I_{\bar{c}_2}$	$I_{c_1 \cup c_2}$	$I_{c_1 \cap c_2}$	$I_{c_1 - c_2}$	$I_{c_1 \oplus c_2}$	$I_{c_1 \odot c_2}$
0	0	1	0	0	0	0	1
0	1	0	1	0	0	1	0
1	0	1	1	0	1	1	0
1	1	0	1	1	0	0	1

lines (cats, lions, tigers, cheetahs, jaguars etc) and if there is a need to classify pets (cats, dogs, parrots etc) a new neural network must be trained. The information about cats learned on the network-1 is not useful for pets classification. This kind of split brains among neural network does not integrate the learning. But if the Metta-Class is used in training, the knowledge can be integrated. Which means the information learned on old networks can be used for newer classifications. Over the time a set of networks will act as a repository, providing information for any kind of classification.

The reason is that the empty space (regions with no available data) is not classified as belonging to any class, as seen in the bottom-right plot of Figure 6. Consequently, the class boundaries learned do not create confusion with any new class, whose data may spread into the empty space of the old classifier's classes in the future. See Section B in the Appendix for an example.

Results on MNIST and CIFAR are presented in the Figure 8 and in Subsection E.3 of the Appendix. The classifiers in the Figure 8 are learned separately without knowledge of the different classes, and the figure particularly shows accuracies when such networks are combined as a repository working as single classifier.

4.2.5. Using classifier for generation of new synthetic data

Neural network when trained properly with the Metta-Class, learns only the data manifold as shown in the Figure 7. It means that the logit values of the data from the class attain positive values only inside the manifold and negative value anywhere outside. If one hypothesizes that logit values attain peak inside the manifold, where the mode of the data lies, and gradually decreases when going away from the mode, then it is possible to reach the mode following the gradient of the logit hence generating a new synthetic datapoint in the process. Images in the Figure 9 are generated using simple gradient descent as shown in the equation 21,

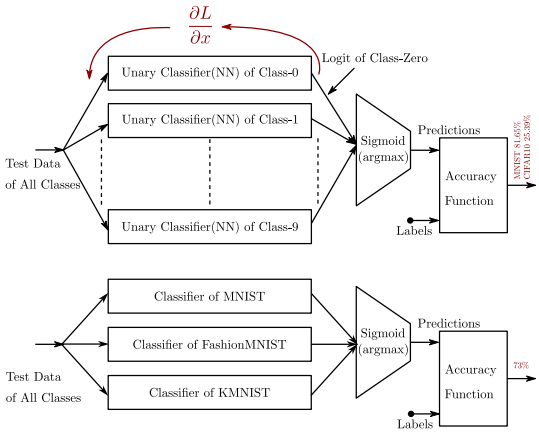


Figure 8. In the top figure each individual class of MNIST/CIFAR10 learned on separate network, and all the individual networks are kept together to act as a single network. It can be seen that for higher-dimensional data this method doesn't scale well. In the bottom figure, different set of MNIST classes are learned on different networks, and all the networks are kept together to act as single network.

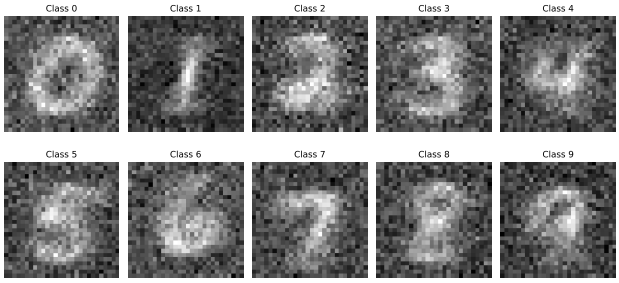


Figure 9. Each image is generated from a classifier trained on a single class (e.g., the bottom-far-left image is generated from a classifier trained solely on digit-5 MNIST data against the Metta-Class).

initialized with random noise image (x_0).

$$x_k = x_{k-1} + \alpha * \frac{\partial L}{\partial x}\bigg|_{x=x_{k-1}} \tag{21}$$

where L is the logit of a particular class, as shown in the Figure 8

4.2.6. Equation discovery

Coefficients of Taylor series equation of a class can be discovered by treating Taylor series Equation (22) (23) as Logit equation of that class, with Cross Entropy loss on

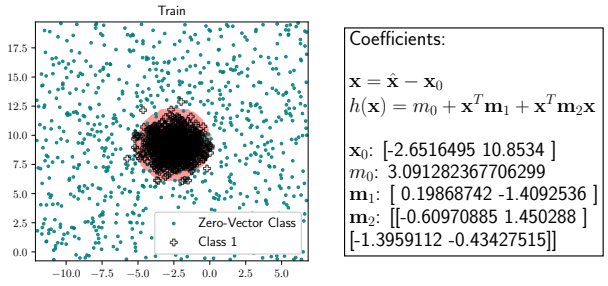


Figure 10. Left plot illustrates Class-1 data, while the right subfigure shows the coefficients of a function representing this Class-1, learned via backpropagation.

whether the Logit value is positive or not.

$$h(x) = b + \sum_{j_1=1}^{d} \lambda_{j_1} x_{j_1} + \sum_{j_1=1}^{d}\sum_{j_2=1}^{d} \lambda_{j_1 j_2} x_{j_1} x_{j_2} + \sum_{j_1=1}^{d}\sum_{j_2=1}^{d}\sum_{j_3=1}^{d} \lambda_{j_1 j_2 j_3} x_{j_1} x_{j_2} x_{j_3} + \dots \quad (22)$$

The equivalent Tensor equation for the Equation (22) can be written as below.

$$h(\vec{x}) = m_0 + m_{1i}\, x_i + m_{2ij}\, x_i x_j + m_{3ijk}\, x_i x_j x_k + ... \quad (23)$$

where $m_0,\ m_{1i},\ m_{2ij},\ m_{3ijk}, ...$ are unknown scalars.

These unknown scalars are learned through backpropagation, as is done in neural networks. An empirical result illustrating this is shown in Figure 10. This can be used for discovering equations of lower dimensional physical phenomenons.

Metrics

To Identify which classifier is superior with respect to learning true boundary, we used the following two metrics:

$$\text{Occupancy Factor} = \frac{n_{[0]:C_1} + n_{[0]:C_2} + n_{[0]:C_3} + ...}{N_{[0]}} \quad (24)$$

where $N_{[0]}$ is total data points of the Metta-Class used in training, and $n_{[0]:C_k}$ is the data points of the Metta-Class recognized as class$-C_k$.

$$\text{Purity Factor Class} - C_k = \frac{n_{(pred=C_k \wedge label=C_k)}}{n_{(pred=C_k)}} \quad (25)$$

where $n_{(pred=C_k \wedge label=C_k)}$ is the total training examples that are predicted as class$-C_k$ and also labeled as class$-C_k$, and $n_{(pred=C_k)}$ is the total training examples predicted as class$-C_k$. Further details on these metrics are provided in Section D of the Appendix.

The applications discussed in this section are empirically evaluated in the Appendix. Detailed results and analysis can be found in Section E.

5. Time Complexity

To analyze the complexity clearly, we divide the computational requirements for any c-class classifier into two distinct parts: (1) Pre-softmax logit computation, and (2) Softmax computation.

Let a *Zero-Exclusive Network* be a classifier that does not use the Metta-Class during training. Suppose there are n training points, and let the computation needed to compute pre-softmax logits be $O(f(n))$—for instance, $f(n)$ might represent millions of computations. Computing the softmax function then takes $O(n \cdot \text{softmax}(c))$ computations, where $\text{softmax}(c)$ represents the complexity of softmax over c classes.

Now consider modifying the Zero-Exclusive Network by adding an extra node at the output to classify the Metta-Class; we call this the *Zero-Inclusive Network*. In this scenario, we incorporate the Metta-Class in training. The addition of the new node introduces extra calculations dependent on the number of nodes in the preceding layer. Letting L denote the number of nodes in the layer preceding this new node, the additional computation required is $O(nL^2)$.

The computational complexities during training and inference are summarized in the Table 2.

Typically, $O(f(n))$ dominates $O(nL^2)$ since L corresponds only to the final internal layer. There is a trade-off between computational overhead and purity improvement. This trade-off depends significantly on the intended applications leveraging the full potential of the Zero-Vector framework.

6. Related Work

Theorem 1 of Gutmann and Hyvärinen's pioneering work on Noise-Contrastive Estimation (2010) establishes that, under its conditions, the logit of a neural network trained to distinguish data sampled from a density $p_d(.)$ and a noise distribution $p_n(.)$ is given by equation 26

$$G(u;\theta) = \ln p_d(u) - \ln p_n(u) \quad (26)$$

In our work, the noise distribution $p_n(.)$ is modeled as the Metta-Class, which follows a uniform distribution, i.e., $p_n(.) = k$, where k is a known constant. Substituting this into equation 26, we obtain:

$$G(u;\theta) = \ln p_d(u) - k \quad (27)$$

If the expression $\ln p_d(x) - k$ is used as a representation in place of Valid Equivalence Sets (e.g., $\ln f(x) - k$ for $[f(x)]$,

Table 2. Computational complexity comparison between a classifier that incorporates the Metta-Class data during training (referred to as a Zero-Inclusive Network) and one that does not (referred to as a Zero-Exclusive Network). Let the computation required to calculate pre-softmax logits during training for n data points be $O(f(n))$ in the Zero-Exclusive Network, where $f(n)$ may represent millions of operations. During inference, the corresponding pre-softmax logit computation for m data points is denoted by $O(g(m))$.

Classifier	**Training Cost**	**Inference Cost**
Zero-Exclusive Network	$O(f(n)) + O(n \cdot \text{softmax}(c)) \approx O(f(n))$	$O(g(m)) + O(m \cdot \text{softmax}(c)) \approx O(g(m))$
Zero-Inclusive Network	$O(f(n+k) + (n+k)L^2) + O((n+k) \cdot \text{softmax}(c+1)) \approx O(f(n+k))$	$O(g(m) + mL^2) + O(m \cdot \text{softmax}(c+1)) \approx O(g(m))$

$\ln g(x) - k$ for $[g(x)]$, $\ln u(x) - k$ for $[u(x)]$, etc.) of the Vector Space in section (3), the same conclusion can be reached—that is, the Metta-Class has a constant logit value. Consequently, the data in the Metta-Class can be interpreted as uniformly distributed.

Based on equation 27, the density can be expressed as:

$$p_d(u) = e^{G(u;\theta)+k} \tag{28}$$

and the score as:

$$\nabla \ln p_d(u) = \nabla G(u;\theta) \tag{29}$$

The score function derived from equation 29 has demonstrated its utility in generating new data using Langevin dynamics (Song & Ermon, 2019). This approach provides an explanation for the successful generation of MNIST data, as outlined in Subsection 4.2.5.

Furthermore, notable related works include advancements in Energy-Based Models, such as (LeCun et al., 2006) and (Grathwohl et al., 2019), which employ strategies to increase the energy of noise samples while decreasing the energy of true data samples.

Additional significant contributions in this domain include (Hinton, 2002), (van den Oord et al., 2018), and (Chen et al., 2020), which explore various methodologies for self-supervised learning and contrastive representation learning.

7. Conclusion

This work presents a mathematical framework to represent classes as vectors in a Vector Space, where Addition corresponds to the union of classes, and scalar multiplication closely resembles the set complement of classes. Notably, it was discovered that the Zero-Vector of the vector space is uniformly distributed in the feature space.

This framework opens up possibilities for developing several novel applications, many of which are not achievable with standard neural network training techniques. Additionally, this training methodology complements Noise Contrastive Estimation and Energy-Based Models, highlighting its utility in enriching existing paradigms.

However, this approach currently demonstrates limitations in scalability, as observed on CIFAR10 datasets. Addressing these challenges offers opportunities to extend this framework to handle higher-dimensional data effectively, paving the way for broader applicability and impact.

Acknowledgements

Harikrishna Metta would like to express his sincere gratitude to his advisor, Prof. R. Venkatesh Babu, for his continuous guidance and support throughout this project. He is also thankful to the examiners, Prof. Aditya Gopalan and Prof. Rajiv Soundararajan, for their valuable time, constructive feedback, and encouragement during the evaluation.

He deeply appreciates the support and collaboration of his lab-mates at VAL, IISc — Priyam Dey, Rishubh Parihar, Badrinath Singhal, Ankit Dhiman, and Abhipsa Basu — for their valuable technical discussions and suggestions.

He would also like to give special thanks to Mohd Shadab Ansari and Shatakshi Gupta for their consistent encouragement, helpful comments, corrections, and moral support throughout the project.

We would also like to thank the anonymous reviewers for their constructive comments and insightful suggestions that helped improve the quality of this work.

Impact Statement

This work aims to advance Machine Learning by introducing a more effective training method for existing models. The proposed approach enables applications that extend beyond the capabilities of traditional neural network training. While we do not anticipate specific ethical concerns, the broader societal impact includes improved accessibility to advanced ML techniques. We encourage responsible and lawful use of this work, with careful consideration of the ethical implications in downstream applications.

References

Axler, S. *Linear algebra done right (eBook)*. Springer, Cham, 3rd ed. edition, 2015. URL https://doi.org/10.1007/978-3-319-11080-6.

Ceylan, C. and Gutmann, M. U. Conditional noise-contrastive estimation of unnormalised models, 2018. URL https://arxiv.org/abs/1806.03664.

Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), July 2009. ISSN 0360-0300. doi: 10.1145/1541880.1541882. URL https://doi.org/10.1145/1541880.1541882.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. E. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020. URL https://arxiv.org/abs/2002.05709.

Cortes, C. and Vapnik, V. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. ISSN 1573-0565. doi: 10.1007/BF00994018. URL https://doi.org/10.1007/BF00994018.

Du, Y. and Mordatch, I. Implicit generation and modeling with energy based models. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/378a063b8fdb1db941e34f4bde584c7d-Paper.pdf.

Elsken, T., Metzen, J. H., and Hutter, F. Neural architecture search: A survey, 2019. URL https://arxiv.org/abs/1808.05377.

Grathwohl, W., Wang, K., Jacobsen, J., Duvenaud, D., Norouzi, M., and Swersky, K. Your classifier is secretly an energy based model and you should treat it like one. *CoRR*, abs/1912.03263, 2019. URL http://arxiv.org/abs/1912.03263.

Grinfeld, P. *Introduction to Tensor Analysis and the Calculus of Moving Surfaces*. Springer New York, NY, 1 edition, 2013. ISBN 978-1-4614-7866-9. doi: 10.1007/978-1-4614-7867-6. URL https://doi.org/10.1007/978-1-4614-7867-6. Published: 24 September 2013, 33 b/w illustrations, 4 illustrations in colour.

Gutmann, M. and Hyvärinen, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Teh, Y. W. and Titterington, M. (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 297–304, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/gutmann10a.html.

Hinton, G. The forward-forward algorithm: Some preliminary investigations, 2022. URL https://arxiv.org/abs/2212.13345.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14 (8):1771–1800, August 2002. ISSN 0899-7667. doi: 10.1162/089976602760128018. URL https://doi.org/10.1162/089976602760128018.

Johnson, J. M. and Khoshgoftaar, T. M. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):27, 2019. ISSN 2196-1115. doi: 10.1186/s40537-019-0192-5. URL https://doi.org/10.1186/s40537-019-0192-5.

LeCun, Y., Chopra, S., and Hadsell, R. A tutorial on energy-based learning. *Tutorial at the International Conference on Machine Learning (ICML)*, 2006.

Makke, N. and Chawla, S. Interpretable scientific discovery with symbolic regression: A review. *Artificial Intelligence Review*, 57(1):2, 2024. ISSN 1573-7462. doi: 10.1007/s10462-023-10622-0. URL https://doi.org/10.1007/s10462-023-10622-0.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space, 2013. URL https://arxiv.org/abs/1301.3781.

Ren, P., Xiao, Y., Chang, X., Huang, P.-y., Li, Z., Chen, X., and Wang, X. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Comput. Surv.*, 54(4), May 2021. ISSN 0360-0300. doi: 10.1145/3447582. URL https://doi.org/10.1145/3447582.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *CoRR*, abs/1907.05600, 2019. URL http://arxiv.org/abs/1907.05600.

Udrescu, S.-M. and Tegmark, M. Ai feynman: a physics-inspired method for symbolic regression, 2020. URL https://arxiv.org/abs/1905.11481.

van den Oord, A., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

A. PDF of resultant class when two different classes are merged.

Suppose there is a need to merge the data of Class$-g$ with the data of Class$-f$. The resulting PDF of the combined class, denoted as $u(x)$, can be expressed in terms of the PDFs $f_n(x)$ and $g_n(x)$, which are the normalized distributions of $f(x)$ and $g(x)$ respectively. The combined distribution can be determined by finding the number of data points that belong to Class$-u$ in an arbitrary interval Δx, as shown in Figure 11.

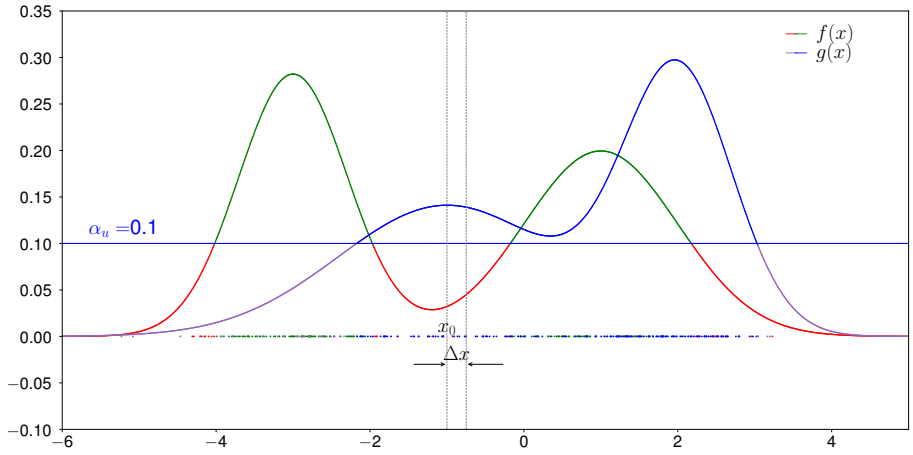


Figure 11. Figure shows data of two distributions combined to form a bigger class.

Let N_f and Ng be the total number of data points belonging to class$-f$ and class$-g$ respectively. Then the number of points in the interval Δx that belong to class$-u$ are

$$N_u = f_n(x_0)\Delta x\ N_f + g_n(x_0)\Delta x\ N_g \tag{30}$$

the density of number of data points at x_0 of class$-u$, say $u(x)$, can be written as below

$$u(x_0)\Delta x\ (N_f + N_g) \approx N_u = f_n(x_0)\Delta x\ N_f + g_n(x_0)\Delta x\ N_g \tag{31}$$

Therefore the probability density $u(x)$ of the class$-u$ is

$$u(x) = \frac{f_n(x)\ N_f + g_n(x)\ N_g}{N_f + N_g} \tag{32}$$

If $N_f = N_g$, then

$$u(x) = \frac{f_n(x) + g_n(x)}{2} \tag{33}$$

B. Continual Learning

In general, when the Metta-Class is not used to train neural network, every time a new class is added to the network, the network must be retrained. And say network-1 classifies all felines (cats, lions, tigers, cheetahs, jaguars etc) and if there is a need to classify pets (cats, dogs, parrots etc) a new neural network must be trained. The information about cats learned on the network-1 is not useful for pets classification. This kind of split brains among neural network does not integrate the learning. But if the Metta-Class is used in training, the knowledge can be integrated. That means the information learned on old networks can be used for newer classifications. Over the time a set of networks will act as a repository, providing information for any kind of classification. An example is shown in Figures 12 and 13.

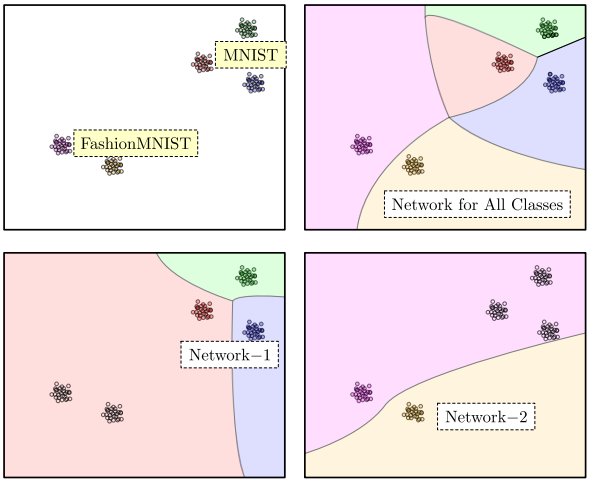


Figure 12. This figure represents how classifiers typically learn when the Metta-Class is not included during training. The top-left subfigure shows the data (a representation of MNIST data), while the top-right subfigure illustrates the boundaries learned by a neural network trained on all the classes. The bottom-left and bottom-right subfigures depict the boundaries learned by two different neural networks trained on MNIST and FashionMNIST data, respectively. The knowledge learned by Network−1 is not compatible or transferable to the knowledge learned by Network−2.

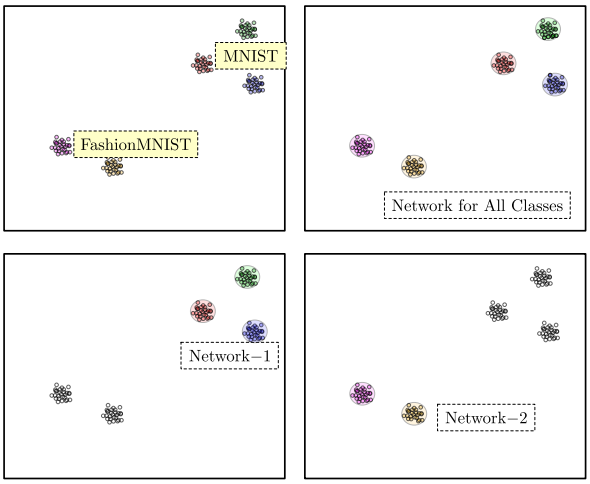


Figure 13. This figure represents how classifiers learn when the Metta-Class data is included during training. The top-left subfigure shows the data (a representation of MNIST data), while the top-right subfigure illustrates the boundaries learned by a neural network trained on all the classes along with the Metta-Class. The bottom-left and bottom-right subfigures depict the boundaries learned by two different neural networks trained on MNIST and FashionMNIST data, respectively, along with the Metta-Class. Unlike the boundaries learned by Network−1 and Network−2 in Figure 12, the boundaries learned here are distinct and can be merged to produce boundaries similar to those of a network trained on all the classes and the Metta-Class, as shown in the top-right subfigure.

C. Taylor-Series Based Network

Any vector of a vector space can be written as linear a combination of a list of linearly independent vectors spanning the vector space, these spanning and independent vectors are called the basis vectors. The following is one such list of linearly independent vectors for the Vector Space V.

C.1. A Basis

A Basis of the vector space V is a list of class-vectors in V that are linearly independent and spans the vector space V.

CLAIM

The set of class-vectors $\{\ [x],\ [x^2],\ [x^3],\ [x^4],\ ....\ \}$ is independent. Since the only way zero-class-vector can be written as linear combination of the set is by making all the coefficients $\lambda_1,\ \lambda_2,\ \lambda_3,\ ....$ zero, it can be seen in the following equation.

$$[0] = \sum_{k=1}^{...} \lambda_k\ [x^k] \tag{34}$$

if k goes to infinity it may span the vector space V as long as the feature space is one dimensional, but it is not going be to verified here. If the feature space is not one dimensional then the data point x is a vector. The following Basis is for a more generalized setting.

If the features space is $d-$dimensional, then x has a vector representation. And let x_i represent the i^{th} element of the vector x. Then the zero-class-vector can be expressed as shown in the Equation (35) if and only if all the coefficients $\lambda_{j_1},\ \lambda_{j_1 j_2},\ \lambda_{j_1 j_2 j_3}...$ are zero, it is not proved here but it is easy to understand looking at the equation.

$$[0] = \sum_{j_1=1}^{d} \lambda_{j_1}[x_{j_1}] + \sum_{j_1=1}^{d}\sum_{j_2=1}^{d} \lambda_{j_1 j_2}[x_{j_1} x_{j_2}] + \sum_{j_1=1}^{d}\sum_{j_2=1}^{d}\sum_{j_3=1}^{d} \lambda_{j_1 j_2 j_3}[x_{j_1} x_{j_2} x_{j_3}] + ... \tag{35}$$

Therefore the set $I_V = \left\{ [x_{j_1}],\ [x_{j_1} x_{j_2}],\ [x_{j_1} x_{j_2} x_{j_3}], ... \middle| j_k \in \{1, ..., d\} \right\}$ is independent. Hence any class-vector $[f(x)]$ from the vector space V can be written as a linear combination of these vectors, assuming they span the Vector Space V, as in the Equation (36).

$$[f(x)] = \sum_{j_1=1}^{d} \lambda_{j_1}[x_{j_1}] + \sum_{j_1=1}^{d}\sum_{j_2=1}^{d} \lambda_{j_1 j_2}[x_{j_1} x_{j_2}] + \sum_{j_1=1}^{d}\sum_{j_2=1}^{d}\sum_{j_3=1}^{d} \lambda_{j_1 j_2 j_3}[x_{j_1} x_{j_2} x_{j_3}] + ... \tag{36}$$

for some $\lambda_{j_1},\ \lambda_{j_1 j_2},\ \lambda_{j_1 j_2 j_3},\ ... \in \mathbb{R}$.

Using addition and scalar multiplication definitions the equation can be written in the following way.

$$[f(x)] = \left[\sum_{j_1=1}^{d} \lambda_{j_1} x_{j_1} + \sum_{j_1=1}^{d}\sum_{j_2=1}^{d} \lambda_{j_1 j_2} x_{j_1} x_{j_2} + \sum_{j_1=1}^{d}\sum_{j_2=1}^{d}\sum_{j_3=1}^{d} \lambda_{j_1 j_2 j_3} x_{j_1} x_{j_2} x_{j_3} + ... \right] \tag{37}$$

Let $hl(x) \in [f(x)]$ is a logit equation. Therefore

$$h(x) = b + \sum_{j_1=1}^{d} \lambda_{j_1} x_{j_1} + \sum_{j_1=1}^{d}\sum_{j_2=1}^{d} \lambda_{j_1 j_2} x_{j_1} x_{j_2} + \sum_{j_1=1}^{d}\sum_{j_2=1}^{d}\sum_{j_3=1}^{d} \lambda_{j_1 j_2 j_3} x_{j_1} x_{j_2} x_{j_3} + ... \tag{38}$$

The Equation (38) is a multidimensional Taylor series equation. All this means is that the logit equation of any class is a multidimensional Taylor series equation. Note that the verification of whether the set I_V spans the Vector Space V has not been conducted. Now, this verification is unnecessary since the Equation (38) is a Taylor series. The logit equations are written as taylor series, and focus here is to learn at least one Valid Logit equation for every class. Furthermore, the set $I_P = \left\{ 1,\ x_{j_1},\ x_{j_1} x_{j_2},\ x_{j_1} x_{j_2} x_{j_3}, ... \middle| j_k \in \{1, ..., d\} \right\}$ is independent and spans polynomial space, proof is not required.

The equivalent Tensor equation for the Equation (38) can be written as below.

$$h(x) = m_0 + m_{1i}\ x_i + m_{2ij}\ x_i x_j + m_{3ijk}\ x_i x_j x_k + ... \tag{39}$$

where $m_0,\ m_{1i},\ m_{2ij},\ m_{3ijk}, ...$ are unknown scalars.

These unknown scalars are learned using the back propagation as in neural networks. This learning of a tensor equation using the back propagation is named as Taylor-Series Based Network.

D. Metrics

As seen in Figure 6, two-dimensional data can be displayed on paper to visualize how the decision boundaries are formed when the Metta-Class data is used to train a network. This displaying of the results helps in comparing how the learning is different when the Metta-Class data is incorporated into the training process versus when it is not.

However, for higher dimensions, visualizing the data to see the decision boundaries becomes challenging. There are dimensionality reduction techniques, such as Principal Component Analysis (PCA) or t-Distributed Stochastic Neighbor Embedding (t-SNE), which can reduce the dimensionality of data for visualization purposes. These techniques can effectively display the data in two or three dimensions, making it easier to understand the distribution and clustering of the data points.

Despite their usefulness in visualizing the data, these dimensionality reduction techniques fall short when it comes to displaying the decision boundaries of classification models. An example is given in the Figure 14. As it can be seen in the third and fourth subfigure of Figure 14 the data points can be clustered and displayed in the lower dimensions, but the boundaries cannot be accurately displayed in the reduced dimensions.

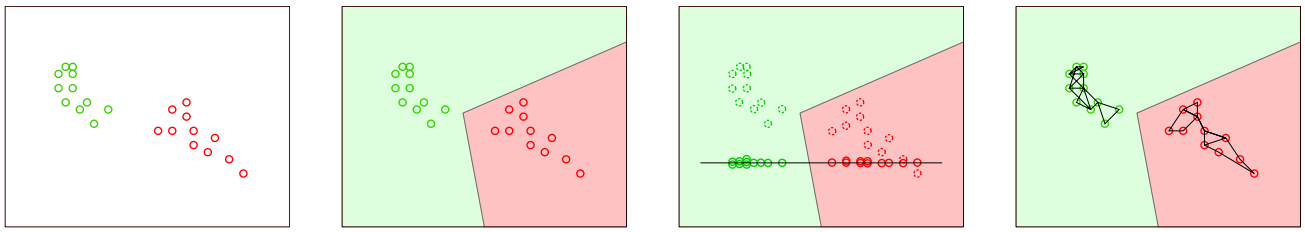


Figure 14. Diagram illustrating the limitations of PCA and t-SNE for visualizing decision boundaries. The top-left subfigure shows example data, while the top-right subfigure displays the decision regions learned by a neural network. The bottom-left subfigure demonstrates PCA applied to reduce dimensionality, highlighting that it is ineffective for visualizing decision boundaries. The bottom-right subfigure shows t-SNE applied for clustering in lower dimensions, which, while effective for grouping data, does not provide insight into decision regions.

Another possible way to visualize higher dimensions is by slicing higher dimensional space into two dimensional or three dimensional spaces and visualizing boundaries in these lower dimensional spaces. A good analogy is that of a CT scan. However, this method is not efficient because the data naturally is very sparse in higher dimensional space.

Therefore it is clear that new metrics are needed to measures the effect of the Metta-Class data on the decision boundaries.

D.1. Metric-1

In Figure 15, figure$-a$ is the learning of two different classifiers. The decision boundaries of Classifier$-R$ are better since the classifier does not classify any given data as one of the three classes(RED, GREEN, BLUE). So there must be a metric to distinguish between these two types of learning. This distinction is possible by counting volume of the predicted classes on both Classifiers. Zero-Vector Data Points(Data sampled from the Metta-Class) can be used for this purpose. Counting Number of Zero-Vector points that are classified as a non-Metta-Class by both classifiers and comparing them, it is possible to distinguish right classifier from the left. Therefore, the following metric called Occupancy Factor defined below to reflect this property.

D.1.1. Definition: Occupancy factor

It is the ratio of the Zero-Vector points that are classified as non-Metta-Class to the total Zero-Vector points.

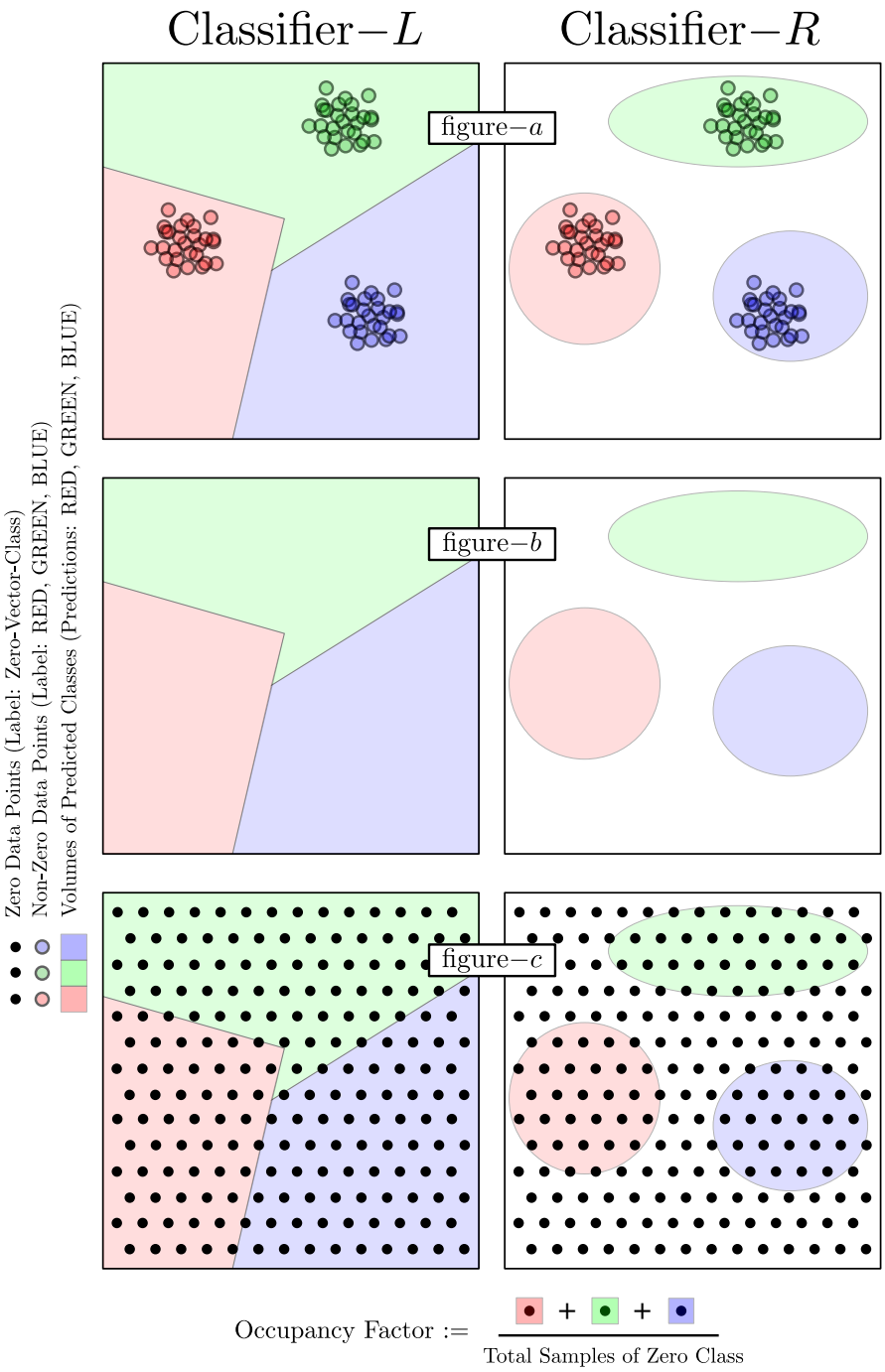


Figure 15. Metric-1 (Occupancy Factor): Figure$-a$ illustrates the learning process of two different classifiers. The decision boundaries of Classifier-R are superior compared to those of Classifier-L, as it avoids classifying any data point as belonging to one of the three explicit classes (RED, GREEN, BLUE). The Occupancy Factor serves as a metric to quantify this difference. Specifically, it measures the volume of decision regions corresponding to the defined classes relative to the volume of the Metta-Class decision region.

Let N be the total Zero-Vector points, n_1 be number of Zero-Vector points classified as RED class, n_2 be the number of Zero-Vector points that are classified as GREEN and n_3 as BLUE, for the Figure 15, then

$$\text{Occupancy Factor} = \frac{n_1 + n_2 + n_3}{N} \tag{40}$$

Occupation Factor is 1 for any classifier that is trained without the Metta-Class data incorporated into the train data since $N = n_1 + n_2 + n_3$. Similarly, Occupation Factor of Classifier$-L$ is 1, and Occupation Factor for Classifier$-R$ is less than 1.

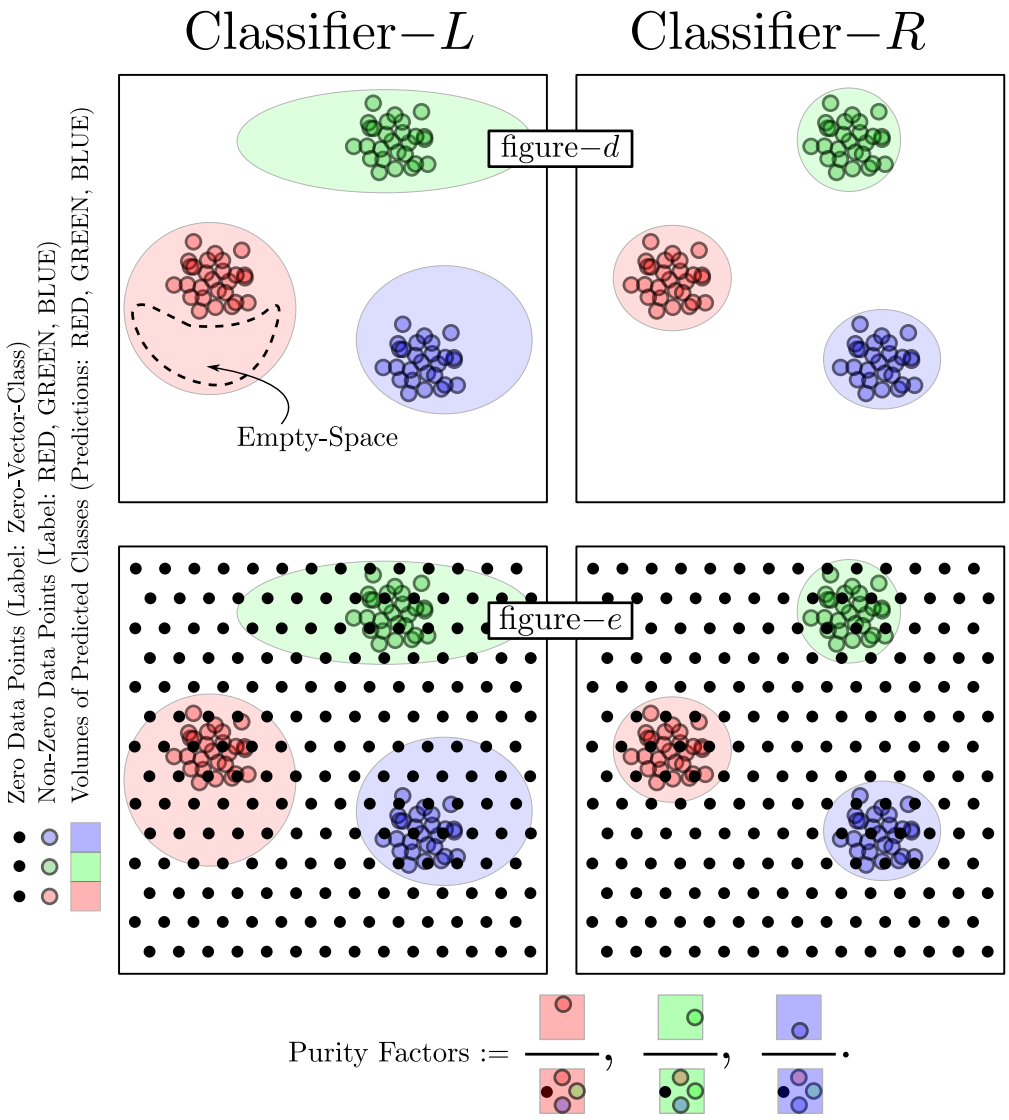


Figure 16. Metric-2 (Purity Factors): Figure$-a$ illustrates the learning process of two different classifiers. Boundaries of the Classifier$-R$ are superior compared with boundaries of the Classifier$-L$ as they represent the true nature of the classes. For example, the 'Empty-Space' in the RED class of Classifier$-L$ also classified as RED class. The Purity Factor serves as a metric to quantify this difference. Specifically, it measures volume of true class with respect to volume of decision region of the class.

D.2. Metric-2

In the next Figure 16, let figure$-a$ is the learning of two different classifiers. Boundaries of the Classifier$-R$ are superior compared with boundaries of the Classifier$-L$ as they represent the true nature of the classes. For example, the 'Empty-Space' in the RED class of Classifier$-L$ also classified as RED class. The Occupancy Factor can be used to distinguish these two classifiers. However, it is possible to have data of one class in the boundary of a different class; for instance, there can be blue data in the Red Volume. Therefore, another metric is defined below to address this issue.

D.2.1. DEFINITION: PURITY FACTORS

Purity Factor is defined for one class. It is the accuracy of the class. For class$-c$, it is defined as the ratio of the total predictions of class$-c$ that are labeled as class$-c$ to the total predictions of class$-c$.

$$\text{Purity Factor of class}-c = \frac{n_{(\text{Pred}=c \,\wedge\, \text{Label}=c)}}{n_{(\text{Pred}=c)}} \tag{41}$$

where

- $n_{(\text{Pred}=c \,\wedge\, \text{Label}=c)}$ is number of data points that are predicted as class$-c$ and also labeled as class$-c$.
- $n_{(\text{Pred}=c)}$ is the number of data points predicted as class$-c$.

Using this definition the Purity Factors for the classifier$-L$ are low compared with the classifiers$-R$. Therefore the Purity Factors tell if the learned boundaries are true boundaries representing true nature of the class or not. Where as the Occupancy Factor tells total volume of predicted Classes in the space.

D.2.2. CAUTION

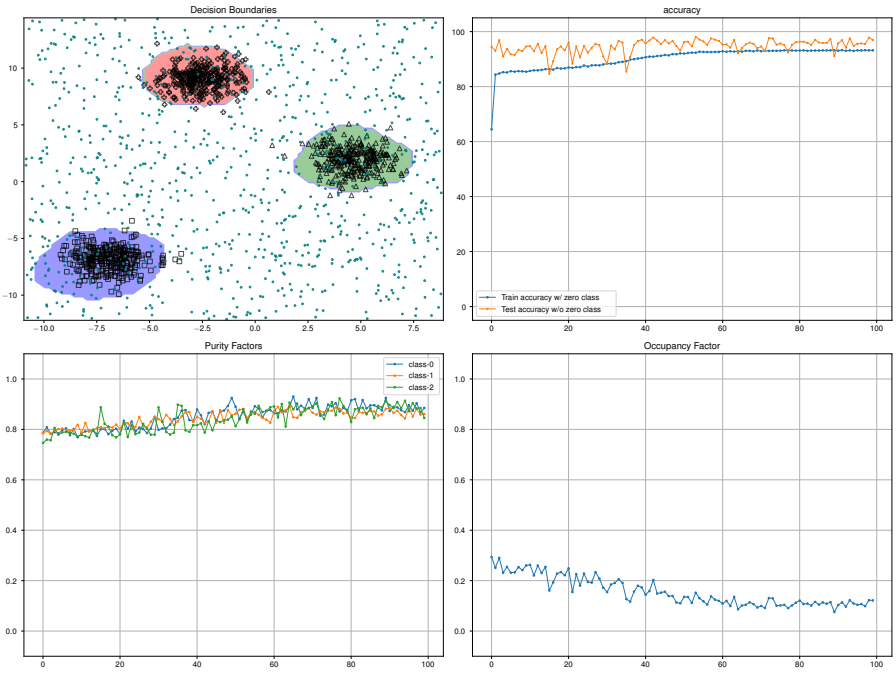
To compare any two classifiers using the Occupancy Factor and the Purity Factors, keep the density of the Zero-Vector-points the same for both classifiers during testing. Therefore, the numbers for Occupancy Factor and Purity Factors are valid only if the number of Zero-Vector data points along with the range of the space in which they are generated, is presented.

D.3. Understanding the Metrics through an Example

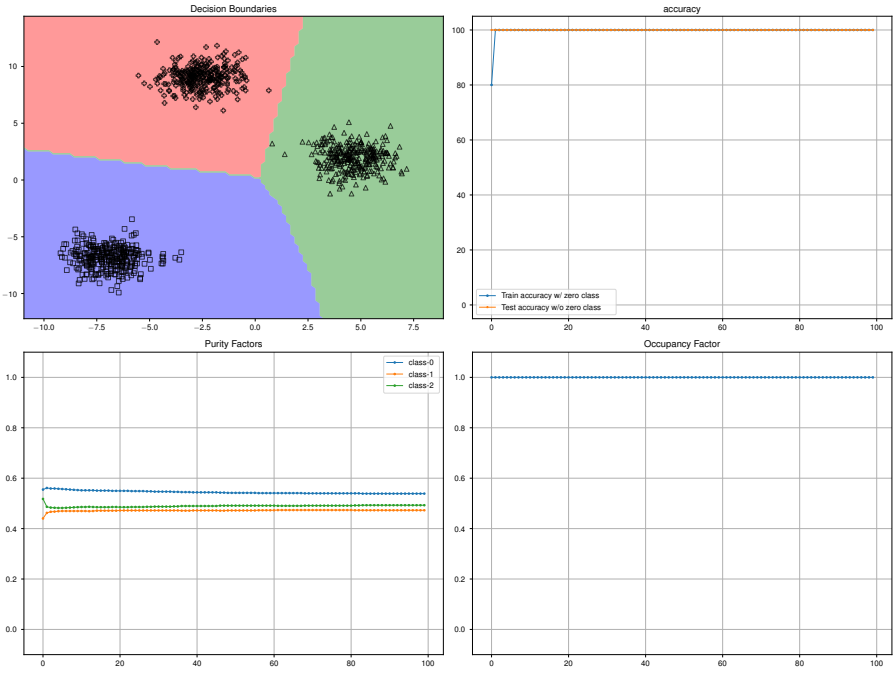
The three class data in the Figure 17 is used to to understand the behavior of the Purity Factors and Occupancy Factor.

The data is used for training two networks, the Metta-Class data is used to train only one of the networks. The results are presented in the following Figure 17. In subfigure 17a the boundaries learned are true boundaries representing true natures of the data. The Occupation factor is reaching zero, indicating that the volume occupied by the Predicted Class is small. Similarly the Purity Factors are high, indicating the empty space in the each class is low.

In contrast, in subfigure 17b, results of classifier trained without incorporating the Metta-Class data, the boundaries do not represent the true nature of the classes. They merely are separating surfaces. Hence, the Occupation Factor is always one. and purity factors are low due to the unwanted empty space present in the each class.



(a) The Metta-Class is incorporated during training. The top-left subfigure illustrates the learned true boundaries. The top-right subfigure depicts the accuracy plot over 100 epochs. The bottom-left subfigure presents the purity factors, all of which approach one, indicating that the data within the decision boundaries becomes increasingly pure as the training progresses. The bottom-right subfigure shows the occupancy factor converging towards zero, implying that the decision regions are progressively enclosing the classes more tightly.



(b) The Metta-Class is not incorporated during training. The top-left subfigure shows the decision regions after 100 epochs, which fail to represent the true regions. These decision regions merely partition the space arbitrarily. Consequently, the occupancy factor remains one throughout the training, and the purity factors are approximately 0.5, as the number of Zero-Vector data points and class data points in each region are nearly equal.

Figure 17. Example to understand the Metrics.

E. Results

E.1. MNIST

Standard MNIST data set is used for the following results.

E.1.1. One Classifier for ALL MNIST classes

The MNIST data is used for training two separate classifiers: one with incorporating the Metta-Class data (Zero-Inclusive-Network) and one without incorporating the Metta-Class (Zero-Exclusive-Network). Results are shown in Figure 18 (Purity factors, Occupancy factors and Accuracies) of both classifiers side by side.

E.1.2. One Classifier for EACH MNIST class

A Unary classifier is trained on each MNIST class data including a Metta-Class data for that class. Results are sown in figures 19, 20, 21, 22, 23, 24, 25, 26, 27, 28. That is each MNIST class data is separated and a Metta-Class data is appended, the resultant data is divided into train and test data, this train data is used to train a classifier, that classifier is called unary classifier of that particular class. At the end these individual classes are used as repository as an experiment. The repository accuracy is calculated at the end as shown in the Figure 41.

E.2. CIFAR10

Standard CIFAR10 data set is used for the following results.

E.2.1. One Classifier for ALL CIFAR10 classes

The CIFAR10 data is used for training two separate classifiers: one with incorporating the Metta-Class data (Zero-Inclusive-Network) and one without incorporating the Metta-Class (Zero-Exclusive-Network). The results are shown in Figure 29 (Purity factors, Occupancy factors and Accuracies) of both classifiers side by side.

E.2.2. One Classifier for EACH CIFAR10 class

A Unary classifier is trained on each CIFAR10 class data including a suitable Metta-Class data for that class. Results are sown in figures 30, 31, 32, 33, 34, 35, 36, 37, 38, 39. That is each CIFAR10 class data is separated and a Metta-Class data is appended, the resultant data is divided into train and test data, this train data is used to train a classifier, that classifier is called unary classifier of that particular class. At the end these individual classifiers are used as repository as an experiment. The repository accuracy is calculated at the end as shown in the Figure 42.

E.3. Rover on an MNIST Planet

The Section 4.2.4 mentions an example of a rover visiting an alien planet. Here it is implemented for the three classes as shown in the following Figure 43.

The following graphs are the results of the training of individual networks. The experiment conducted is as follows, it has been assumed that the rover got to know MNIST data alone on that planet, so a network is trained for MNIST data alone and added to the repository of the MNIST. Later the Rover comes across the FashionMNIST data, a new network is trained on for the FashionMNIST data alone without disturbing the previously add Network in the repository, and added the newly trained network to the repository. At this point the accuracy of the repository is calculated. At last the rover reaches the KMNIST data, a new network is trained on that KMNIST data and added the new network to the repository. Again the repository accuracy calculated here. The following are the corresponding diagrams.

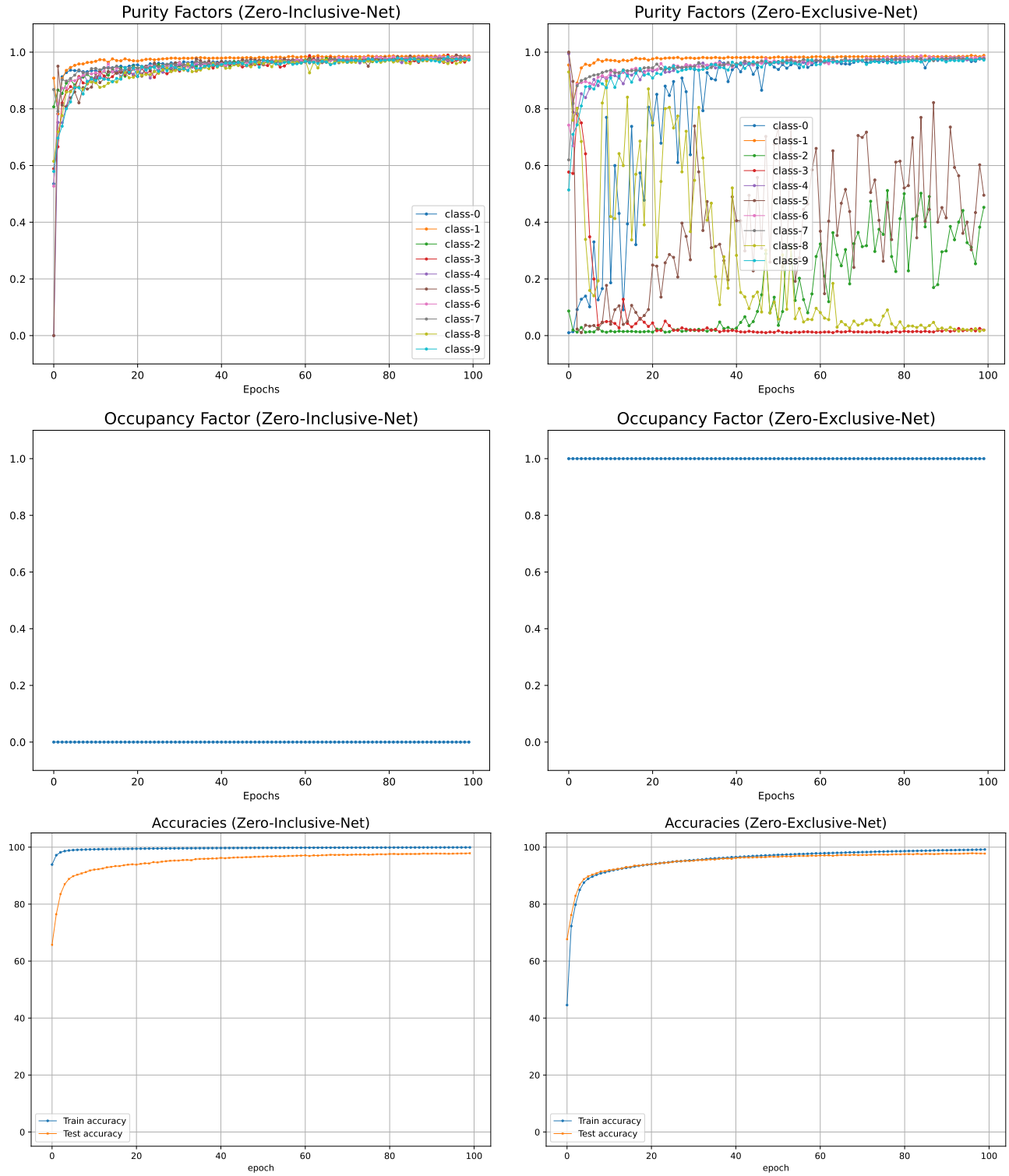


Figure 18. The MNIST data is used for training two separate classifiers: one with incorporating the Metta-Class data (Zero-Inclusive-Network) and one without incorporating the Metta-Class (Zero-Exclusive-Network). When trained with the Metta-Class the purity is high as shown in the top two plots, occupancy is low as shown in the middle plots, and test accuracy is same for both networks as shown in bottom figures. (The Metta-Class data used to calculate the train accuracy but not used in calculation of test accuracy)

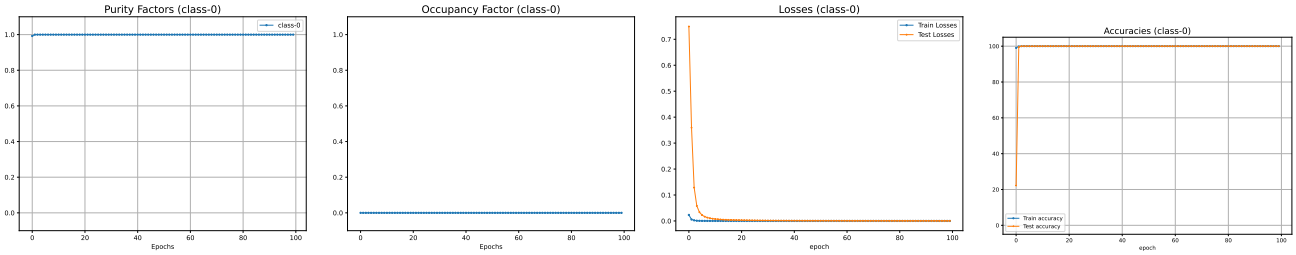


Figure 19. Training results of Unary Classifier of MNIST Class 0

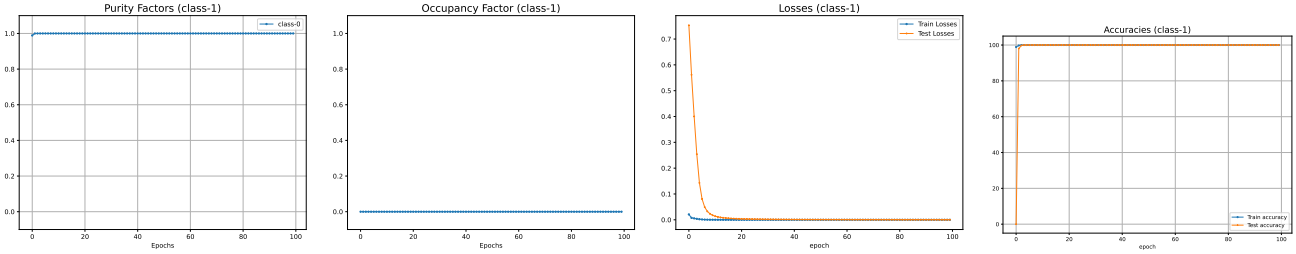


Figure 20. Training results of Unary Classifier of MNIST Class 1

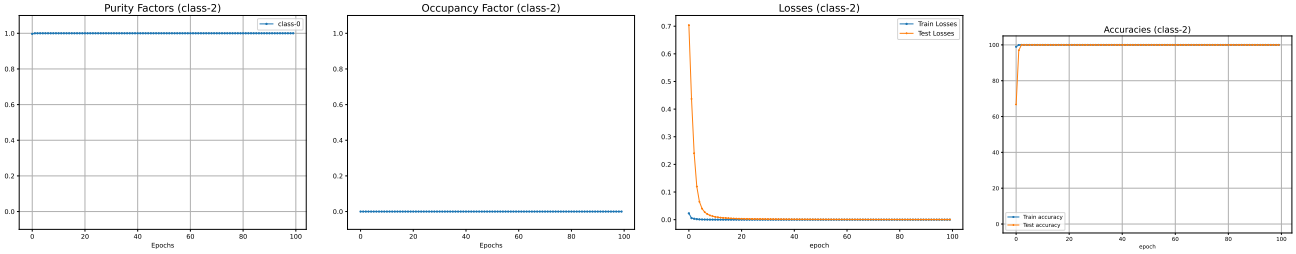


Figure 21. Training results of Unary Classifier of MNIST Class 2

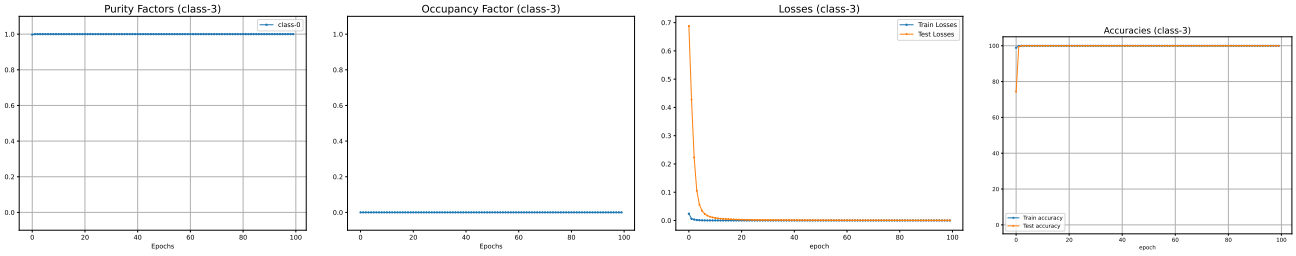


Figure 22. Training results of Unary Classifier of MNIST Class 3

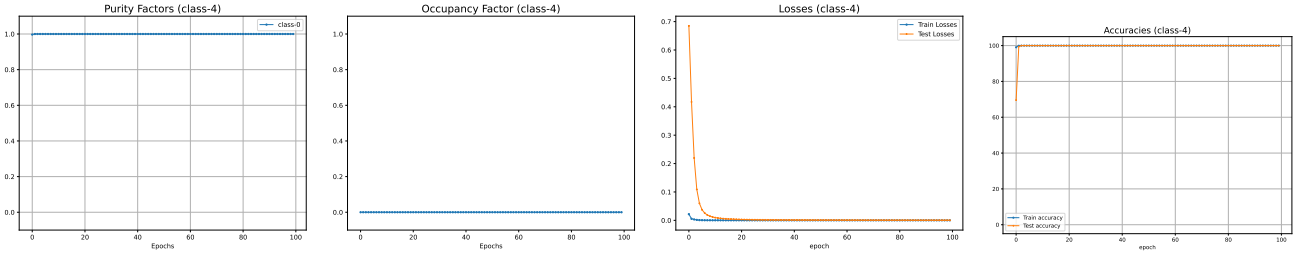


Figure 23. Training results of Unary Classifier of MNIST Class 4

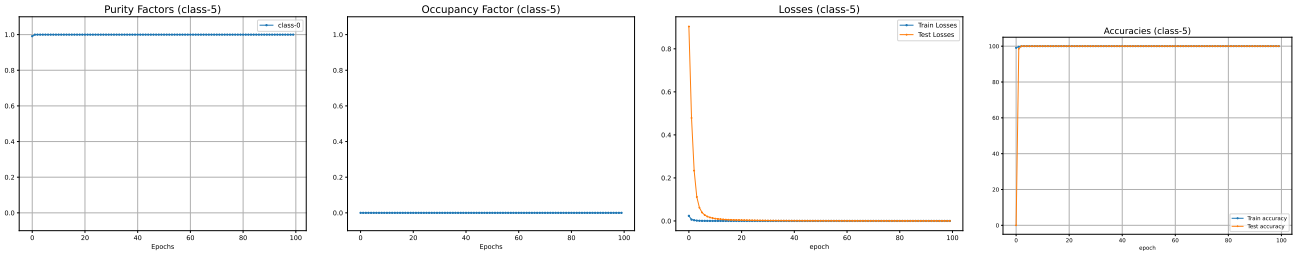


Figure 24. Training results of Unary Classifier of MNIST Class 5

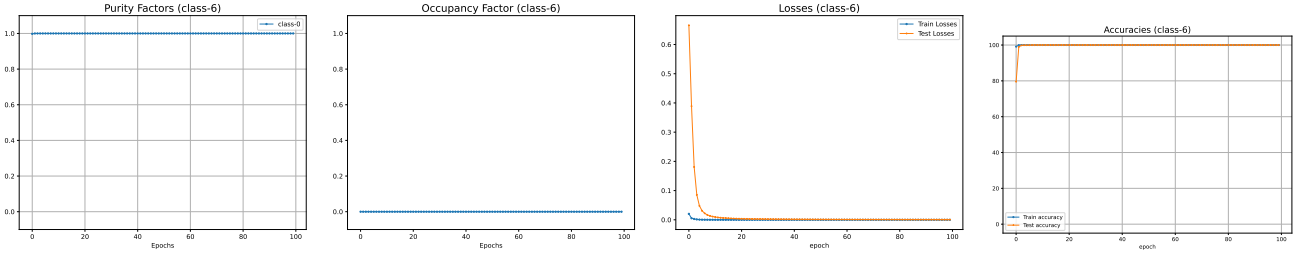


Figure 25. Training results of Unary Classifier of MNIST Class 6

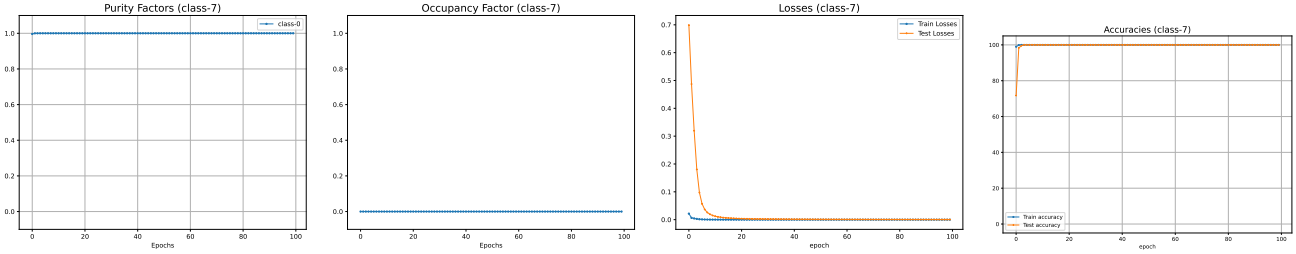


Figure 26. Training results of Unary Classifier of MNIST Class 7

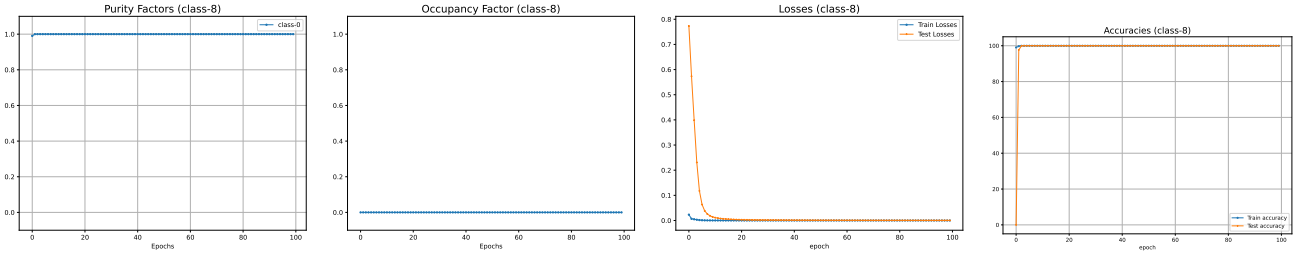


Figure 27. Training results of Unary Classifier of MNIST Class 8

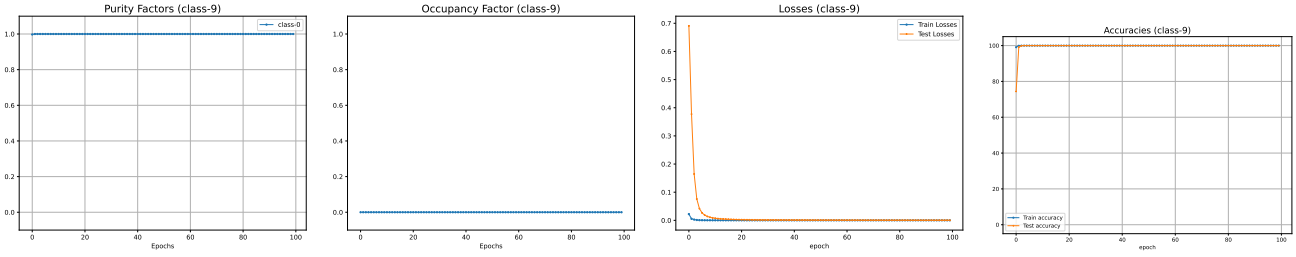


Figure 28. Training results of Unary Classifier of MNIST Class 9

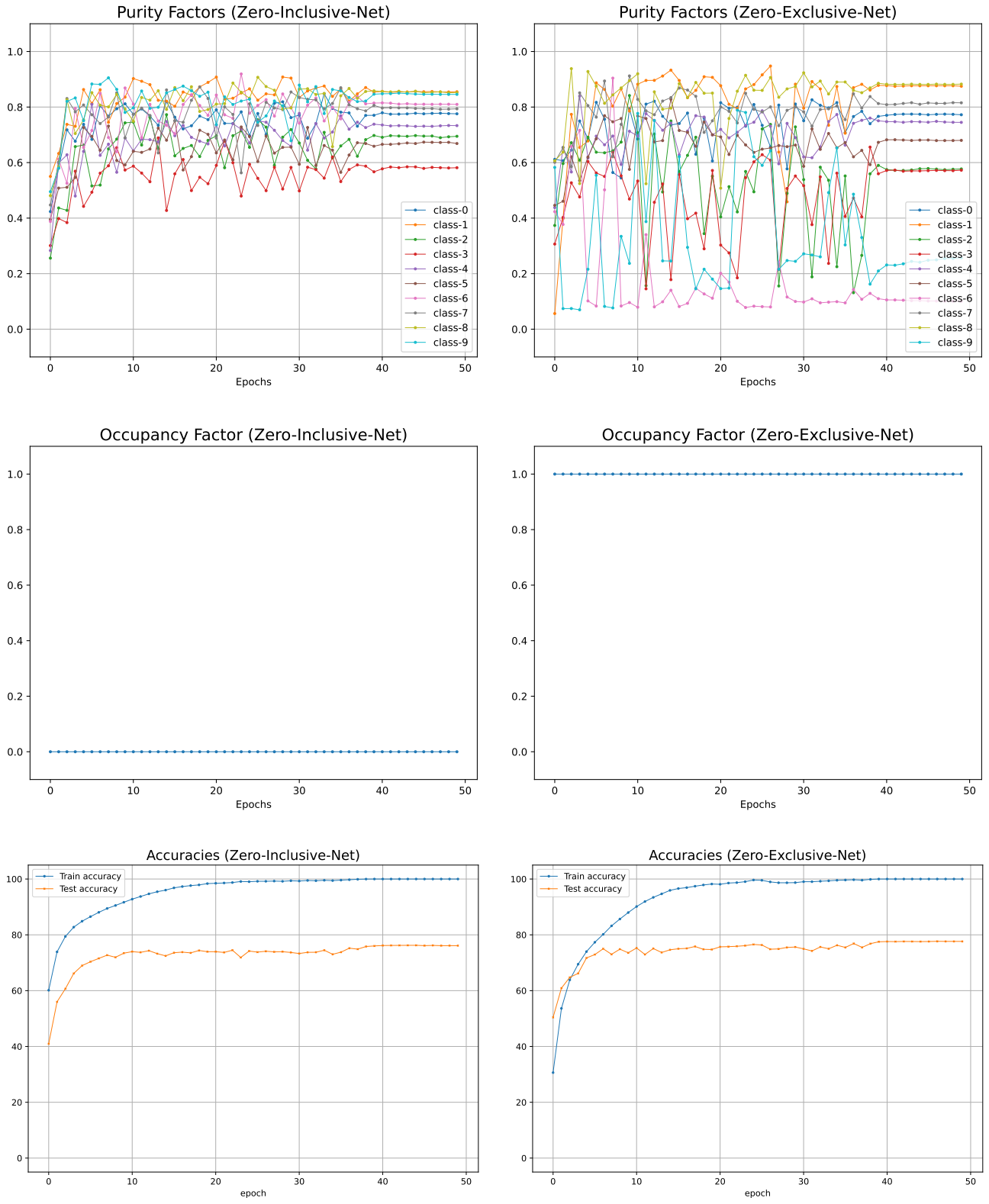


Figure 29. The CIFAR10 data is used for training two separate classifiers: one with incorporating the Metta-Class data (Zero-Inclusive-Network) and one without incorporating the Metta-Class (Zero-Exclusive-Network). When trained with the Metta-Class the purity is high as shown in the top two plots, occupancy is low as shown in the middle plots, and test accuracy is comparable with the network trained without the Metta-Class. (The Metta-Class data used to calculate the train accuracy but not used in calculation of test accuracy)

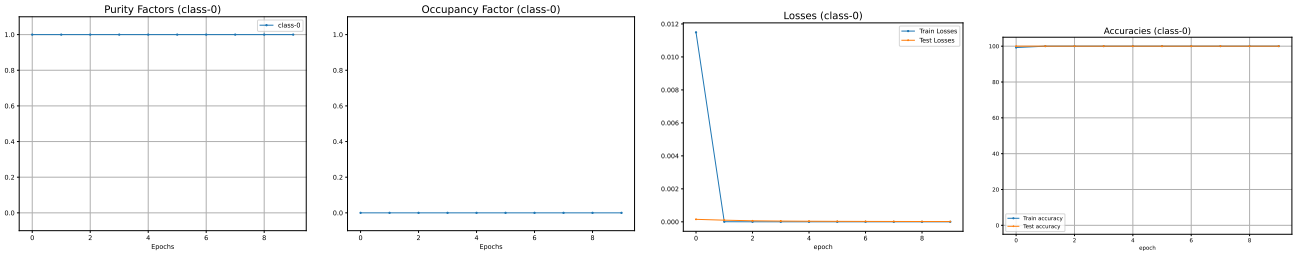


Figure 30. Training results of Unary Classifier of CIFAR10 Class 0

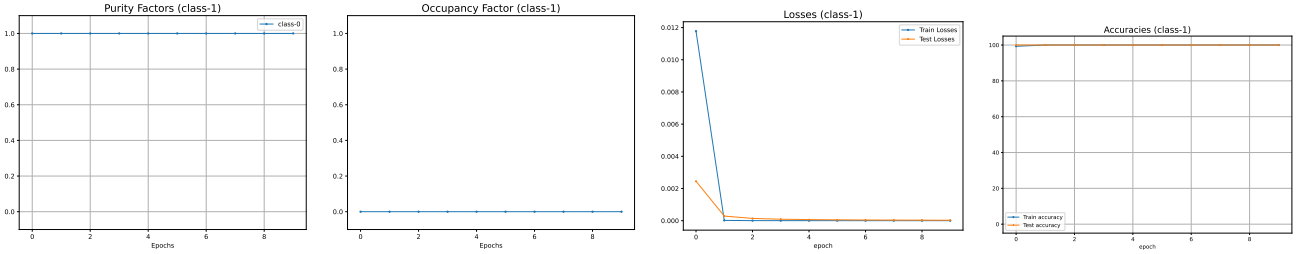


Figure 31. Training results of Unary Classifier of CIFAR10 Class 1

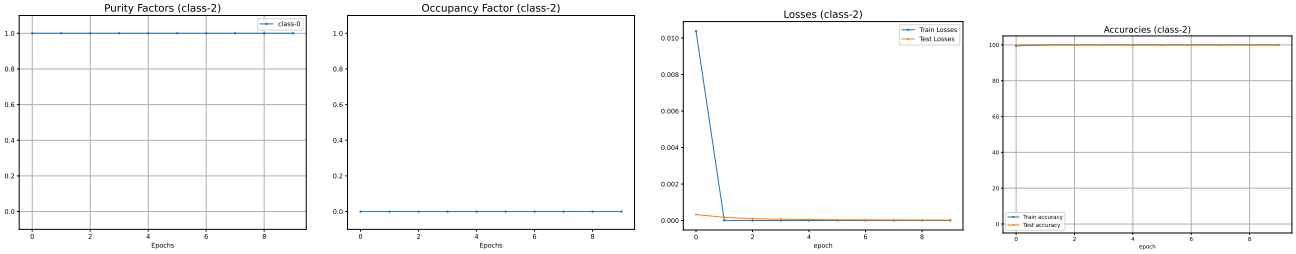


Figure 32. Training results of Unary Classifier of CIFAR10 Class 2

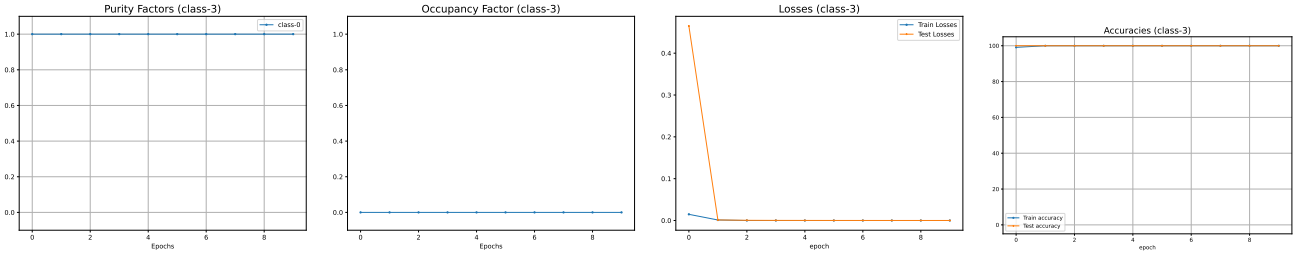


Figure 33. Training results of Unary Classifier of CIFAR10 Class 3

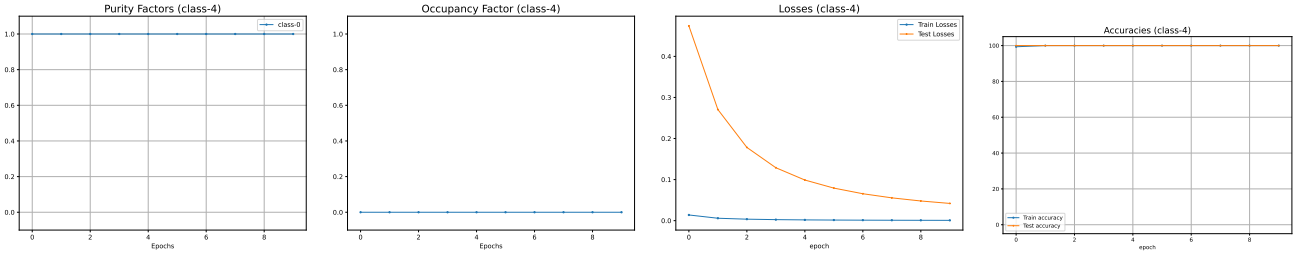


Figure 34. Training results of Unary Classifier of CIFAR10 Class 4

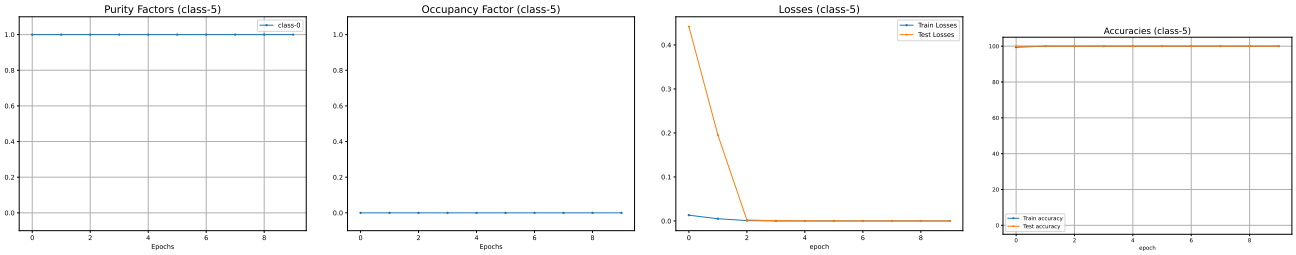


Figure 35. Training results of Unary Classifier of CIFAR10 Class 5

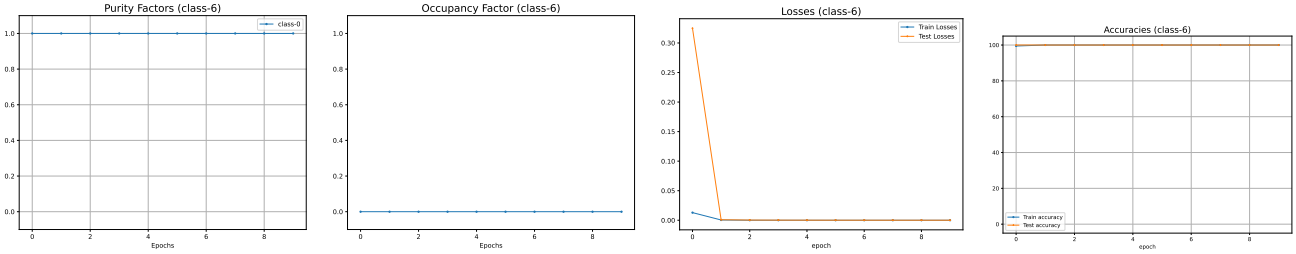


Figure 36. Training results of Unary Classifier of CIFAR10 Class 6

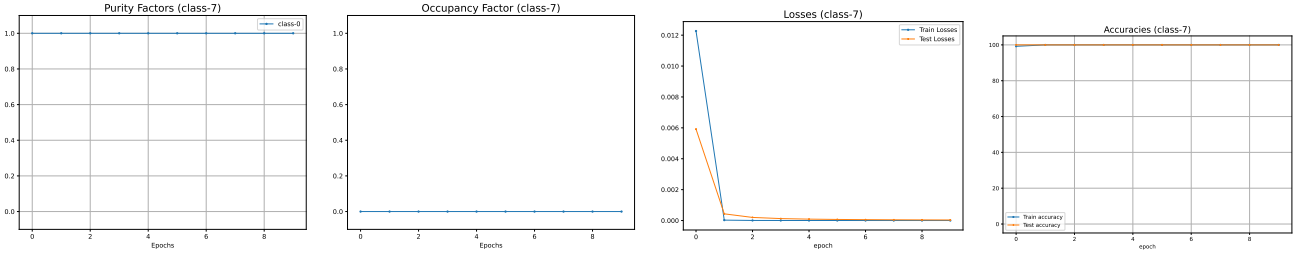


Figure 37. Training results of Unary Classifier of CIFAR10 Class 7

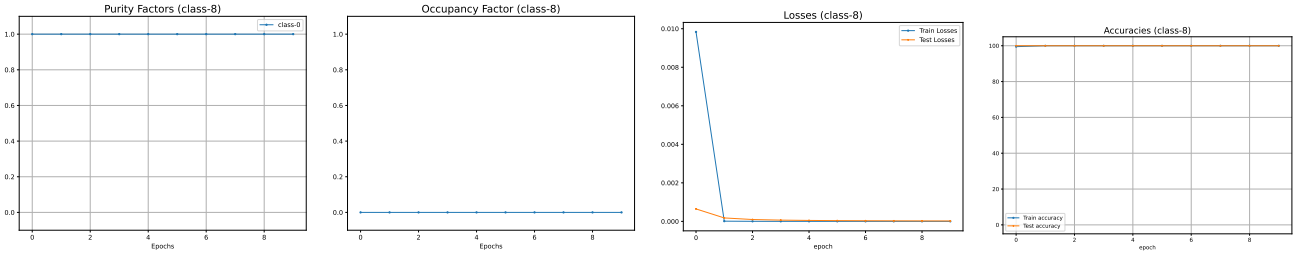


Figure 38. Training results of Unary Classifier of CIFAR10 Class 8

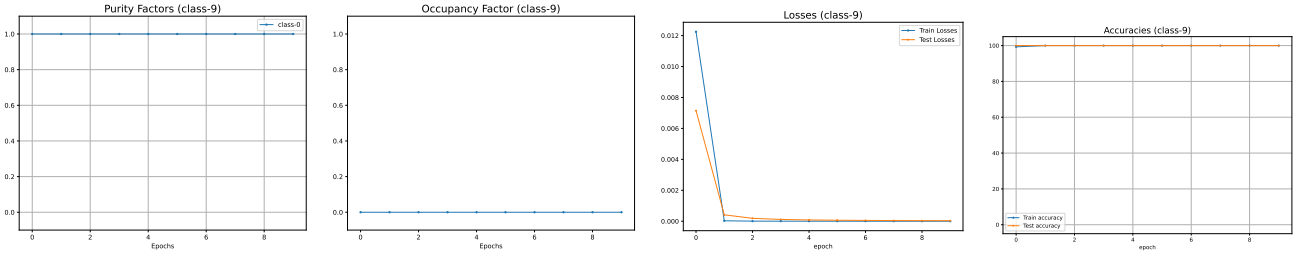


Figure 39. Training results of Unary Classifier of CIFAR10 Class 9

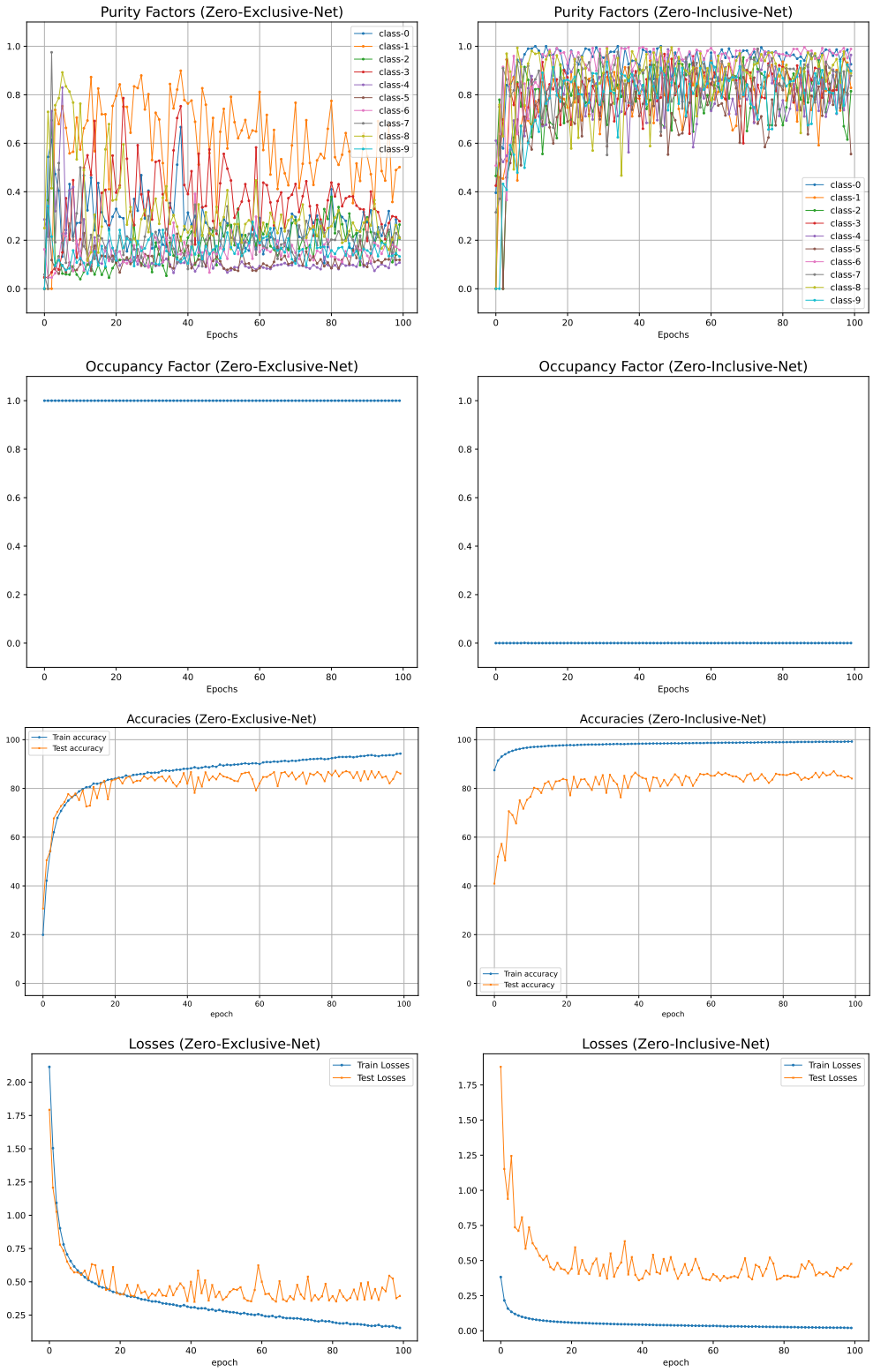


Figure 40. The **ImageNet-1k** embeddings (extracted using a Masked Autoencoder—MAE) are used for training two separate classifiers: one with incorporating the Metta-Class data (Zero-Inclusive-Network) and one without incorporating the Metta-Class (Zero-Exclusive-Network). When trained with the Metta-Class the purity is high as shown in the top two plots, occupancy is low as shown in the second row plots, and test accuracy is same for both networks as shown third row plots. (The Metta-Class data used to calculate the train accuracy but not used in calculation of test accuracy).

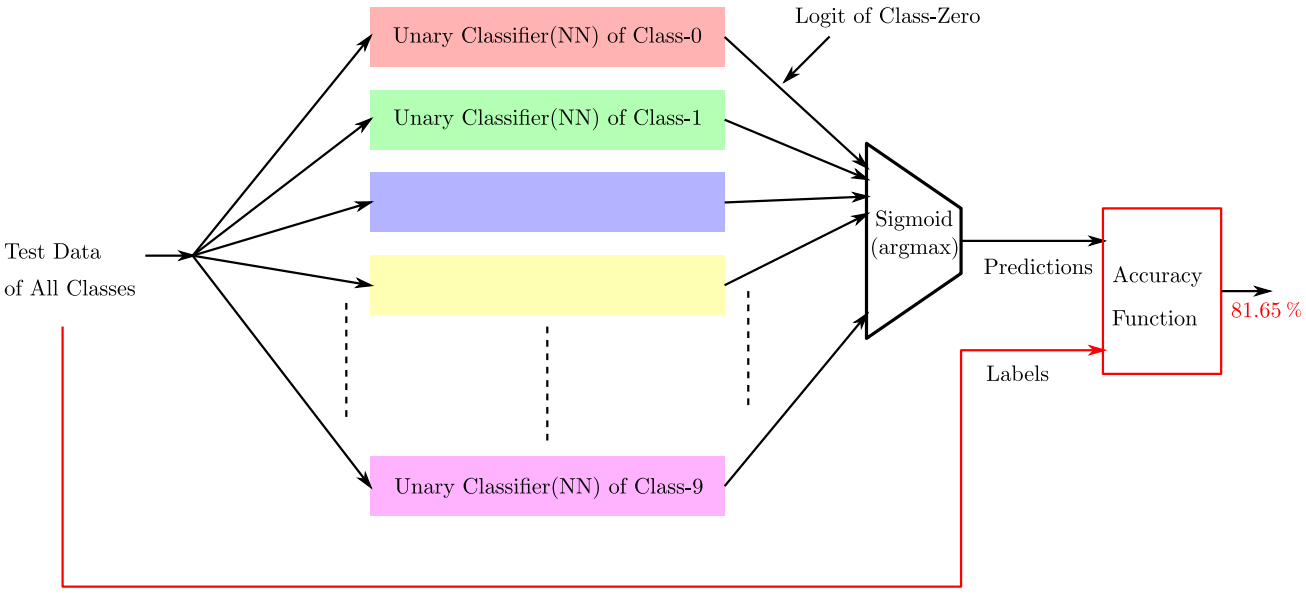


Figure 41. Accuracy of the Repository of MNIST Unary Classifiers: Figure shown Method used for calculating combined accuracy of all individual classifiers, where each classifier is trained to recognize a specific MNIST class.

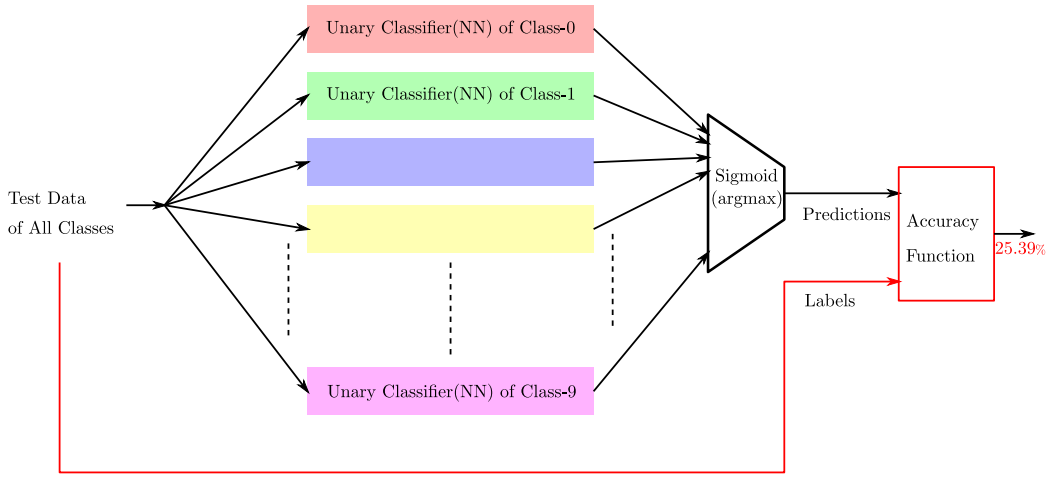


Figure 42. Accuracy of the Repository of CIFAR10 Unary Classifiers: Figure shown Method used for calculating combined accuracy of all individual classifiers, where each classifier is trained to recognize a specific CIFAR10 class.

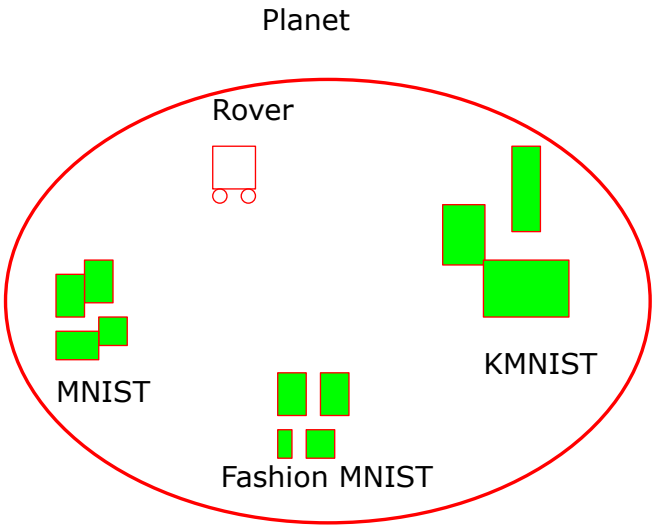


Figure 43. The figure illustrates an example of how continual learning can be implemented. Imagine a rover landing on a new planet, encountering different types of data as it explores various locations. A separate neural network can be trained for each unique type of data. By incorporating the Metta-Class during training, knowledge from different classifiers can be shared without needing to retrain the one big classifier for all classes each time it encounters new data of different class.

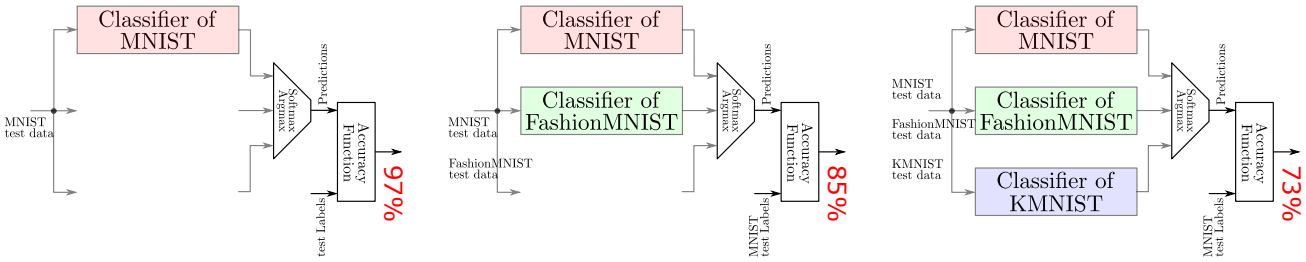


Figure 44. Figure shows continuous learning on MINST datasets. The far-left figure shows a classifier trained only on MNIST, incorporating the Metta-Class data during its training. In the middle, a new, separate classifier trained on FashionMNIST and the Metta-Class data is added to the existing classifier. The far-right figure shows another new classifier trained on KMNIST and the Metta-Class data is added on top of the existing classifiers.

F. Equivalence set as set of Tuples

As mentioned in the section 2 (Characterization of Logits) of the main paper, if the probability density function (PDF) of the data, $f(x)$, is known, a given data point can be assessed for membership in the distribution by comparing its probability to a predefined threshold, α_f. The threshold α_f should be chosen as a non-negative real number, ensuring that data points with high relative probability are recognized as belonging to the class. Therefore tuple $(f(x), \alpha_f)$ defines that class (class$-f$) of data. To generalize it further lets take $f_T(x)$ is the threshold of the PDF $f(x)$, and the tuple $(f(x), f_T(x))$ defines that class of data. Consider a black-box operator that takes a function as input and returns a threshold function suited for that PDF. In fact, an entire set of such tuples can represent the same class. For example, $(f(x) + k, f_T(x) + k)$ defines the same class-f for any real-valued constant k. All such tuples preserve the decision boundary and thus represent the same underlying class.

Let $\mathcal{M}$ be the set of monotonically non-decreasing functions. Then each tuple in the set

$$[f(x)] := \left\{(M \circ f(x),\ M \circ f_T(x)) \;\middle|\; M \in \mathcal{M}\right\} \tag{42}$$

also defines the same class-f. Here, $M \circ f(x)$ denotes $M(f(x))$, written this way to reduce bracket clutter.

Definition of set V

Let V be the set of sets of the kind shown in the equation 42. Therefore the set V can be written as

$$V = \{\ [f(x)],\ \ [g(x)],\ \ [h(x)],\ \ ....\ \} \tag{43}$$

Throughout this report, whenever the vector space V is mentioned, it refers to this vector space.

Definition of Addition and Scalar multiplication

As understood in the main paper, the equations for union and complement are taken as definitions here with slight modifications so that they are suitable for defining a vector space. Specifically,

ADDITION

For all $[f(x)],\ [g(x)] \in V$, the sum $[f(x)] + [g(x)]$ is defined as

$$[f(x)] + [g(x)] := [f(x) + g(x)] \tag{44}$$

SCALAR MULTIPLICATION

For all $\lambda \in \mathbb{R}$ and for all $[f(x)] \in V$ the product $\lambda[f(x)]$ is defined as

$$\lambda\ [f(x)] :=\ [\lambda\ f(x)] \tag{45}$$

All the properties of a vector space on the set V, including the definitions of addition and scalar multiplication, can be verified similarly to what has been done in the main paper. Here, we focus specifically on the additive identity (the zero-vector). Let $[I(x)]$ is the identity of this vector space. Then by the definition of identity for any vector $[f(x)] \in V$

$$[f(x)] + [I(x)] = [f(x)] \tag{46}$$
$$[f(x) + I(x)] = [f(x)] \tag{47}$$
$$[(f + I)(x)] = [f(x)] \tag{48}$$
$$\left\{(M \circ (f + I)(x),\ M \circ (f + I)_T(x)) \;\middle|\; M \in \mathcal{M}\right\} = \left\{(M \circ f(x),\ M \circ f_T(x)) \;\middle|\; M \in \mathcal{M}\right\} \tag{49}$$
$$\tag{50}$$

If $I \in \mathcal{M}$ then above equality can be written as,

$$\left\{(N \circ f(x),\ N \circ f_T(x)) \;\middle|\; N \in \mathcal{M}\right\} = \left\{(M \circ f(x),\ M \circ f_T(x)) \;\middle|\; M \in \mathcal{M}\right\} \tag{51}$$

Therefore, the additive identity (i.e., the zero-vector) of the vector space possesses the property of being a monotonically non-decreasing function. Since every vector in V represents a class (i.e., a probability distribution over data), there must exist a probability density function (PDF) corresponding to the zero-vector that is monotonically non-decreasing.

To understand the possible form of the zero-vector, we consider the following two extreme cases:

1. When the PDF is constant.
2. When the PDF is strictly monotonically increasing.

Case 1: PDF of the Zero-Vector is Constant

A constant PDF corresponds to a uniform distribution. Hence, if the zero-vector has a constant PDF, the data associated with the zero-vector is uniformly distributed across its support.

Case 2: PDF is Strictly Monotonically Increasing

Let A be the volume spanned by a PDF that is strictly monotonically increasing and has zero probability outside the volume V. Since a PDF must integrate to one over its support, as $A \to \infty$, the PDF appears approximately constant within any finite, localized region. Therefore, samples drawn from such a PDF will appear uniformly distributed in small local areas.

From both Case 1 and Case 2, it follows that data sampled from the zero-vector appears uniform within any finite local region. Hence, the Metta-Class (the Zero-Vector Class) effectively behaves as a uniform distribution.

NOTE:

Some technical details, such as the constraints on the set $\mathcal{M}$, have been omitted to streamline the intuition. Readers are advised not to treat this as a formal proof but rather as a conceptual hint or intuition behind viewing the Zero-Vector as a uniform distribution.