Deep Learning based Anomaly Detection using Ensembles with Leave-out classes

Libin Kutty
Faculty of Computer Science, DKE
Otto-von-Guericke University
Magdeburg, Germany
libin.kutty@st.ovgu.de

Ritu Gahir
Faculty of Computer Science, DKE
Otto-von-Guericke University
Magdeburg, Germany
ritu.gahir@st.ovgu.de

Viju Sudhi
Faculty of Computer Science, DE
Otto-von-Guericke University
Magdeburg, Germany
viju.sudhi@st.ovgu.de

Abstract

Anomalies are the out-of-distribution data that needs to be recognized and removed before it affects the system. Modern neural networks are capable to classify data with high accuracy based on the knowledge it gained from the training data but are unable to identify if the new sample belongs to in-distribution or is simply an anomaly. To address this issue, various state-of-the-art methods exploit the knowledge of the softmax distribution of a sample by a classifier and determine if the sample is an anomaly or not. In our work, to improve the anomaly detection performance, we exploit the strength of ensemble learning and combine the knowledge gained by each classifier trained on different samples of data by leaving one class out at a time.

We propose two methods in this work. The first method is a statistical comparison with the help of In- and Out-of-Distribution (OOD) reference vectors and utilizing them to determine anomalies. The second method focuses on training a binary classifier on the dataset generated by combining the output softmax distribution of a sample by each classifier in the ensemble and subsequently classify a given instance as an anomaly or not.

Our work includes experiments and evaluation of ensemble method with leave-out classes to detect anomalies on image datasets (MNIST and CIFAR-10) and text datasets (20 Newsgroups). Our implementation outperforms the baseline method on the MNIST dataset.

Index Terms

Anomaly detection, Ensemble learning, Leave-out class, Softmax distribution, Euclidean distance, Entropy, Decision rule, Binary Classifier

I. Introduction

In recent years, supervised training of neural networks has helped in the much-sought classification tasks with reasonably high accuracy, in various fields of application such as fraud detection, intrusion detection, fault detection, and system health monitoring. But even such ”highly accurate” networks are still unreliable when it comes to recognizing an instance - the sort of which was not present while training. Such instances termed as ’anomalies’ or ’outliers’ or ’out-of-distribution data’ - used interchangeably in literature - are capable of adversely affecting the systems in use, increasing the need to detect them well before deployment. This is very much inline with the definition of anomaly by Hawkins as “an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism” [1]. The extent of compliance or deviance of features to call an instance an outlier or otherwise is, however, subjective and depends significantly on the application [2].

From the probability distribution from classifiers performing well on training and validation data, the softmax value of the predicted class label tends to be higher for those instances the model is so sure of. However, depending on the train and validation accuracies, the model may or may not give less softmax values for those instances it was not trained on. In a sensitive application area, such as healthcare systems, one would not want this uncertainty from an otherwise well-performing model. This is where anomaly detection finds its role, demanding any network to initially assign low softmax values to anomalous instances and thereby, reliably distinguishing in- from out-of-distribution data.

Seeking the aid of the predictions from a group of classifiers trained with different training data distributions and different sets of hyper-parameters on different model architectures could possibly give a better performance than a single classifier trained on the same classification task. This is the motivation behind ensemble learning, where a set of classifiers, trained with different settings, provides different perspectives on a given sample, helps to achieve higher classification accuracy. Ensembles try to eliminate the gap between algorithms and applications by making use of the best from multiple algorithms. Ensemble approaches are also used to address the bias-variance trade-off, as discussed in [3].

In our implementation [1], we train an ensemble of classifiers on different data distributions generated by leaving one class and propose two methods for anomaly detection. The first method involves statistical comparison with reference vectors for IN and OOD data and using them differently to determine the anomalies. The second method focuses on learning the softmax

[1]https://github.com/libinjohn26/Anomaly_Detection

probability distribution pattern of data, by training a binary classifier on a dataset generated by considering the softmax distribution predicted by each classifier in the ensemble on a sample.

In the context of anomaly detection using ensembles, we try to answer the following questions:

- **RQ-1:** Can ensemble learning aided with different data distributions help in distinguishing anomalous data from the ones the model was trained on?
- **RQ-2:** Can the knowledge gained from leaving out classes in the classifiers of an ensemble be exploited to learn more about the anomalous data?

The rest of the report discusses related works in the field of anomaly detection using ensemble learning and leave-out classes in Section II, followed by a detailed description of the proposed method and steps involved in detecting anomalies in Section III. Various experimental settings and results for the methods proposed are explained in Section IV for MNIST, CIFAR-10, and 20 Newsgroups datasets, and a comparative study of these results is presented in Section V. Finally, the conclusion of these experiments and the performance of ensemble in anomaly detection over a single classifier is explained in Section VI.

II. Related Work

Owing to its relevance in different fields ranging from software to healthcare, different methods have been proposed for detecting anomalies. The authors of [4] categorize the methods broadly into threshold methods, neighborhood methods, and model-based methods. Threshold methods exploit the knowledge of a range of values both the normal instances and anomalous instances give, while the neighborhood methods compare the distance with neighbors and arrive at a conclusion. Model-based methods try to fit a given dataset and make use of this knowledge to detect instances from an anomalous class.

All of the above methods for anomaly detection rely on the inherent properties of a dataset and might not work the same way with a different dataset possessing no similar features with the former. This is when, as the authors of [5] suggest, ensembles help in getting better results. However, the question lingers to the number of classifiers in an ensemble which could give satisfactory results in detecting an anomaly. This points in the direction of the 'law of diminishing returns in ensemble construction'. The experiments by [6], [7] give a rather broad conclusion that the ensemble size should be selected considering the number of class labels. They suggest, choosing the number of classifiers in an ensemble such that it is around the number of class labels could give better results than with an ensemble of larger sizes. Our study with different ensemble sizes also agrees with this finding II.

The authors of [8] discuss how ensemble methods have been explored less for anomaly detection. [9] claims to be the first research tying ensemble methods for anomaly detection. The authors tried to approach the problem by combining the different factors leading to anomaly - from different high dimensional sub-spaces, thereby creating a framework that unifies other researches in the field with regard to the causal factors considered and the combination functions used.

According to the authors of [10], the simplest way to detect anomalies is to utilize the softmax probability values from the trained model. The softmax probability values (corresponding to the predicted class label) are found to be higher for those instances the model could predict with better confidence and lesser for those instances the model could not. The latter instances can be considered anomalous. In this paper, the authors try to find if this assumption is good for different types of datasets - image, text, and speech. We have considered this method as our baseline to compare our results.

III. Concept

In our implementation, we propose two methods that work on the softmax probability distribution of each classifier in the ensemble. Before diving deep into how the individual method is implemented for each dataset, this section gives a general overview of the data distribution considered to train each classifier in an ensemble.

A. Anomalous Class

As discussed in the previous sections, an anomaly is out-of-distribution data on which the model is not trained on. In our work, we considered a single class from the given data as an anomaly and none of the classifiers are trained on this anomalous class. For example, in the MNIST dataset, class 9 is considered as an anomaly while the rest of the classes are considered as IN distribution for training.

B. Leave-out Class

For training each classifier in an ensemble, we leave out one class from IN distribution data, and this class is considered as OOD for that particular classifier. We have used the information of this OOD data to identify the actual anomalous class.

For example, in the MNIST dataset with class 9 as the anomalous class, we train classifier 0 with class 0 as OOD data, and the rest of the classes, ie. class 1 through class 8 are considered as IN distribution data. Refer Fig. [1] for the remaining distribution.

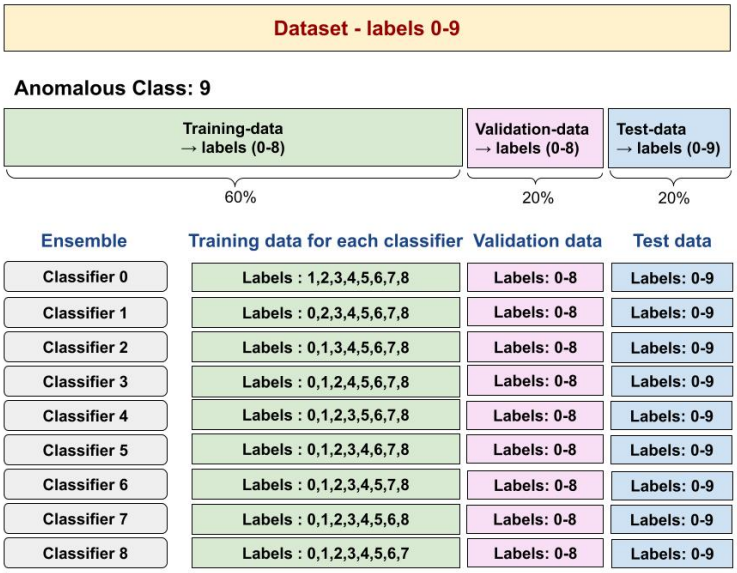

Fig. 1. Data-set distribution for anomalous class 9

C. Dataset Distribution

The dataset is divided into 3 sets - training, validation, and test sets in the ratio of 60 : 20 : 20, as shown in Fig. 1. The test set contains all the class labels including the anomalous class, while the validation set contains all the classes except for the anomalous class. The training set consists of classes except for the leave-out class of the corresponding classifier and the anomalous class.

D. Model Calibration

To make our predictions more reliable, we calibrated our model using temperature scaling. Temperature scaling helps in distributing the softmax probabilities more uniformly for those instances which the classifier is not sure about. This should give us a softmax pattern in such a way that softmax values for OOD data are distributed across class labels but for IN data, the values tend to be concentrated on a specific label.

IV. Methodology

A. Method 1 - Statistical comparison

Method 1 is a statistical comparison with the help of rules or similarity measures to determine whether the input is IN data or OOD data. To determine the similarity between vectors, we experimented with both cosine similarity and Euclidean distance. Converting the Euclidean distance into a similarity metric is done according to Eqn. 1.

$$sim(\overrightarrow{v_1}, \overrightarrow{v_2}) = \frac{1}{1 + dist(\overrightarrow{v_1}, \overrightarrow{v_2})} \tag{1}$$

For both cosine and Euclidean similarities, the closer the value to 1, the higher is the similarity between vectors. We observed in our experiments that Euclidean distance performed better compared to cosine similarity to determine the IN or OOD data. We suppose this could be because Euclidean distance considers the relative distance between vectors. However, cosine similarity takes the angle between vectors into account. As a result, even when the vectors are closer but are opposite to each other relative to the origin, they would be considered dissimilar. On the other hand for the Euclidean similarity, since the Euclidean distance between vectors is small, Euclidean similarity will be high, making them more similar. Hence, for our similarity calculations, we have used Euclidean distance, rather than cosine similarity.

1) Reference Vectors: Reference vectors, as the name suggests, are used to compare our data with reference data to determine which distribution our data belong to. We determined the reference vector for IN data and OOD data separately.

Reference vectors are calculated based on the average maximum softmax value of each classifier in our ensemble. We used validation data to determine our reference vectors. From each classifier, we considered the average maximum softmax value and form a C-dimensional vector, where C stands for the number of classifiers.

From classifier i, for each instance j we have softmax values $\overrightarrow{y_{i,j}}$ as,

$$\overrightarrow{y_{i,j}} = [y_{i,j}^{(1)}, y_{i,j}^{(2)}, .., y_{i,j}^{(K)}] \tag{2}$$

Here, K indicates the number of class labels the classifier is trained on. In our case, $K = 8$.

Now, computing maximum softmax value for the j^{th} instance, we get $(y_{i,j})_{max}$ as,

$$(y_{i,j})_{max} = \max_k y_{i,j}^{(k)} \tag{3}$$

For classifier i, taking average of maximum softmax values for all the instances in the validation data, we get r_i as,

$$r_i = \frac{1}{|D|}\left[\sum_{j=1}^{D}(y_{i,j})_{max}\right] \tag{4}$$

Here, D is the number of instances in validation data. For IN reference vector, we consider only those instances from the validation data that belong to the IN distribution, D_{IN} and likewise for the OOD reference vector using D_{OUT}.

Finally, reference vector $\overrightarrow{v}$ is obtained by appending r_i values for each of the classifiers.

$$\overrightarrow{v} = [r_1, r_2, .., r_C] \tag{5}$$

Here, C is the number of classifiers.

Since we used the concept of leave-out classifiers, we created two reference vectors - one with the help of classes on which the classifiers were trained and one with the classes on which the classifiers were not trained. Since the classes on which the classifiers were not trained are considered as OOD data for that particular classifier, it gives us an idea of how the vectors would look like in general for IN and OOD data.

While testing, we considered the maximum softmax value of each classifier to form the C-dimensional vector, and compared this vector with the reference vectors using Euclidean similarity.

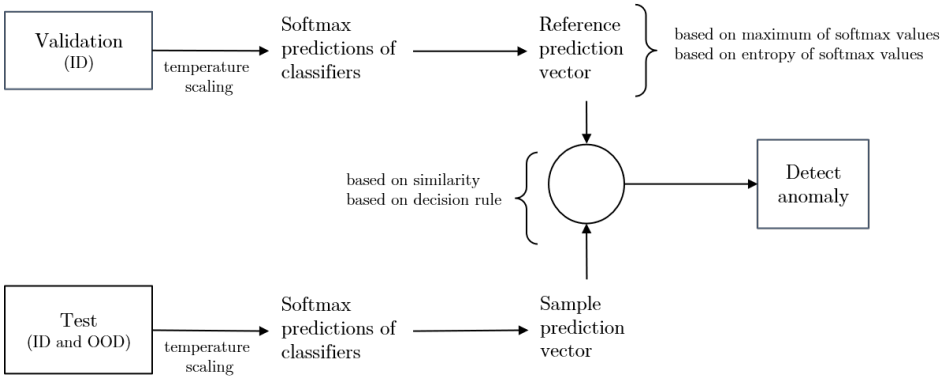

Fig. 2. Statistical comparison of softmax distribution

2) Entropy Value: Entropy determines how softmax probabilities are distributed across all the class labels. The idea behind using entropy is based on the softmax distribution for IN and OOD data. It is observed that the softmax probabilities are distributed uniformly among classes in case of OOD data, resulting in high entropy values, while for IN data the softmax probabilities are concentrated over one particular class label, resulting in lower entropy values.

We calculated reference entropy value for both IN data and OOD data based on our validation data set. We take the average entropy value of each classifier to determine our reference entropy value. We used Euclidean similarity to aid comparison.

From classifier i, for each instance j we have softmax values $\overrightarrow{y_{i,j}}$ as,

$$\overrightarrow{y_{i,j}} = [y_{i,j}^{(1)}, y_{i,j}^{(2)}, .., y_{i,j}^{(K)}] \tag{6}$$

Here, K indicates the number of class labels the model is trained on.

We then calculate entropy on these softmax values to get entropy $r_{i,j}$ for instance j as,

$$r_{i,j} = -\sum_{k=1}^{K} y_{i,j}^{(k)} log(y_{i,j}^{(k)}) \tag{7}$$

Finally, we compute the average with respect to the number of data in the validation set and then again take average with respect to the number of classifiers to get the reference entropy value e, given by Eqn. 8.

$$e = \frac{1}{|C|} \sum_{i=1}^{C} \left[\frac{1}{|D|} \sum_{j=1}^{D} r_{i,j} \right] \tag{8}$$

Here, C indicates the number of classifiers and D indicates the number of instances in the validation data.

3) Decision Rule: To determine whether the data is IN data or OOD data based on the predictions of all the classifiers in our ensemble, we have considered a simple decision rule. The rule assumes that if the majority of the classifiers predicts the same label (absolute majority count), then the data is considered to be IN data and OOD, otherwise. Since each classifier was trained leaving one class out, for each of the IN distribution instances, the majority of the predictions should be the same.

Algorithm 1 Decision Rule

$AMC \leftarrow$ Absolute majority count
$threshold \leftarrow$ 7 or 8

if $AMC(PredictedLabels) \geq threshold$ **then**
  ID
else
  OOD
end if

For example, if 8 classifiers predict the label of a sample as 1 and the 9th classifier predicts something else, we consider this data to be of IN data. On the other hand for OOD data, labels predicted by classifiers would vary, thereby reducing the absolute majority count.

4) Combination of above methods: We have tried different combinations of decision rule and similarity methods with reference vectors and with entropy reference values.

Algorithm 2 Decision Rule and Similarity

$AMC \leftarrow$ Absolute majority count
$threshold \leftarrow$ 7 or 8
$sim_{IN} \leftarrow$ Similarity With IN References
$sim_{OOD} \leftarrow$ Similarity With OOD References

if $sim_{IN} \geq sim_{OOD}$ **and** $AMC(PredictedLabels) \geq threshold$ **then**
  ID
else
  OOD
end if

B. Method 2 - Softmax Pattern Learning

The second method that we propose involves training a binary classifier on the dataset built from the softmax probability distribution. The motivation behind this method is the ability of classifiers to understand the underlying patterns in a given data. It is believed that if enough training data is provided to the classifiers, they are capable to learn from it and recognize the underlying pattern and classify/predict the test data correctly. In our implementation, as discussed in Section III, the ensemble of classifiers is trained by considering one class at a time as an anomalous class. From the validation set which contains labels from 0 to 8, each sample is given to all the classifiers in the ensemble. The softmax distribution of each classifier is stored and label 1 for IN-distribution and label 0 for OOD distribution is assigned. Since the ID and OOD label for each classifier during training is known, it is easy to label the softmax distributions.

To perform the binary classification, the Support Vector Machine (SVM) classifier is trained on 80% of the newly created dataset and its performance is tested on the remaining 20% newly created dataset, split using stratified sampling. However, the newly created dataset is highly imbalanced with IN-distribution samples 8 times more than OOD-distribution samples. In order to address this issue, instead of normal SVM, weighted SVM is used where the individual class weights are determined

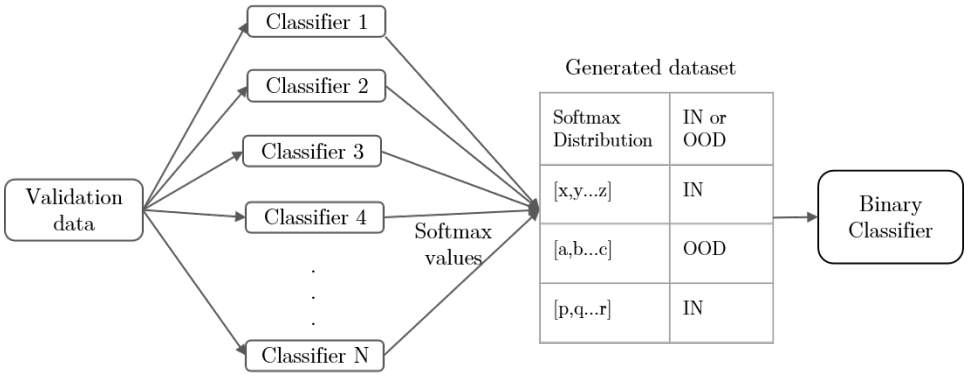

Fig. 3. Training a binary classifier

based on the label distribution in training data. While training the SVM on the training dataset when these class weights are introduced, the classifier tends to give more importance to certain classes which will encourage the classifier to get these samples right.

Once the binary classifier is trained, the samples from the test set are given to each classifier in the ensemble. This softmax distribution from each classifier is then given as an input to the SVM binary classifier and the classifier classifies it as 1 i.e the sample is IN for that classifier or as 0 i.e the sample is OOD for that classifier. Once we get these classifications from all the classifiers, the majority prediction of ID or OOD is assigned to the test sample. In order to check the performance of the proposed method, AUROC is calculated for these predicted labels against their actual labels.

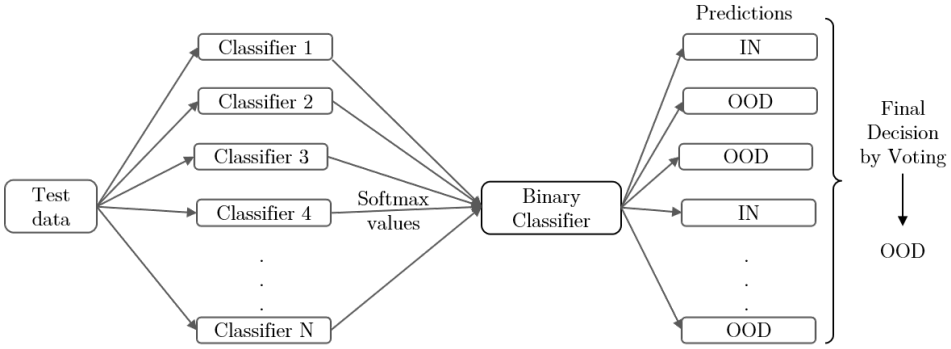

Fig. 4. Testing the binary classifier

V. Experiments

We experimented with multiple scenarios for method 1. We first calculated the Euclidean distance of the data vector with respect to reference vectors and convert them into similarity scores and then used these similarity values in different contexts.

- Comparing similarity with both IN and OOD reference vector and determine anomaly based on with which the data has the highest similarity
- Calculate similarity with just IN reference vector
- Calculate similarity with just OOD reference vector
- Comparing similarity with both IN and OOD entropy reference value and determine anomaly based on with which the data has the highest similarity
- Calculate similarity with just IN entropy reference value
- Calculate similarity with just OOD entropy reference value

We also tried different decision rules and combinations of decision rules.

- Decision rule with majority label count greater than equal to 7
- Decision rule with majority label count greater than equal to 8
- Decision rule with majority label count greater than equal to 7 and checking similarity with both reference vectors
- Decision rule with majority label count greater than equal to 8 and checking similarity with both reference vectors

- Decision rule with majority label count greater than equal to 7 and checking similarity with both entropy reference values
- Decision rule with majority label count greater than equal to 8 and checking similarity with both entropy reference values

To check the usage of our method, we have experimented on three different datasets namely MNIST, CIFAR-10, and 20 Newsgroups. For simplification, we only consider 10 classes from 20 newsgroups. We conduct the experiments considering all the classes in our dataset as anomalies individually and take the average result for each anomalous class to determine how our method works with respect to baseline.

A. MNIST and CIFAR-10 dataset

For training MNIST and CIFAR-10 datasets, we use the same deep learning architecture with the same hyperparameters. With Adam optimizer and sparse categorical cross-entropy loss. We trained our classifiers for 20 epochs with a batch size of 128.

B. 20 Newsgroups Dataset

Out of the 20 classes in the dataset, we considered only the first 10 classes for our experiments. These classes were selected randomly, as given in Table I, to make a total count of 10 which facilitated the analysis of text and image datasets with the same number of class labels throughout our experiments. We used large-BERT implementation[2] uncased model to classify the Newsgroups with sparse categorical cross-entropy loss and warm-up steps for 10 epochs with a batch size of 32.

TABLE I
CLASSES CONSIDERED FROM 20 NEWSGROUPS DATASET

Sl. No.	Class name
1	rec.motorcycles
2	rec.sport.baseball
3	comp.os.ms-windows.misc
4	talk.religion.misc
5	comp.sys.mac.hardware
6	soc.religion.christian
7	sci.crypt
8	comp.sys.ibm.pc.hardware
9	comp.windows.x
10	comp.graphics

VI. EVALUATION

This section discusses our results and inferences both on the experimental setup we chose for the three different datasets and a comparative study on our different approaches.

A. Evaluation of the experimental setup

Our work was initially focused to determine the optimal number of leave-out classes for each classifier that could yield us a good AUROC score.

TABLE II
NUMBER OF CLASS LABELS -VS- NUMBER OF CLASSIFIERS

Distribution	# leave out classes	# classes	# classifiers	AUROC score
6ID 4OOD	1	**6**	**6**	**84.15**
6ID 4OOD	2	6	15	77.91
9ID 1OOD	1	**9**	**9**	**87.73**
9ID 1OOD	2	9	36	82.39

As Table II shows, we performed experiments with the MNIST dataset considering one, and four random classes as anomaly leaving either one or two classes for training each classifier in the ensemble. It was seen when the number of classifiers is close to the number of class labels, we obtained better AUROC scores. Though the experiments were not exhaustive, from the obtained figures, we draw a rather loose conclusion which is surprisingly inline with the 'law of diminishing returns of ensemble construction'. This motivated us to perform further experiments with just one leave-out class with all the other datasets.

[2]https://github.com/tensorflow/models/tree/master/official/nlp/bert

TABLE III
AUROC SCORES (MNIST) WITH AND WITHOUT TEMPERATURE SCALING

	Without temperature scaling	With temperature scaling
Baseline	83.55	**93.00**
Method 1	87.20	**88.42**

We also wanted to check how model calibration would affect the final AUROC scores. Table III gives a solid idea that AUROC scores are sufficiently higher when the logits are temperature scaled before returning the final softmax values from the classifiers. The results are consistent regardless of the approach we used in detecting the OOD from IN distribution data.

TABLE IV
TRAINING AND VALIDATION ACCURACY

Dataset	Training accuracy	Validation accuracy
MNIST	99.51	98.22
CIFAR-10	86.86	77.77
20 Newsgroups	99.97	93.37

The above experiments and results helped us to arrive at a common experimental setup for all three datasets. We considered 9 IN distribution class labels and 1 OOD distribution class label and trained 9 classifiers in our ensemble by leaving 1 class out in each of them. This setup gave us the training and validation accuracies as detailed in Table IV. We assume that it is the less complexity of the dataset trained with a more complex architecture that helped us get the best accuracies for MNIST when comparing with the other datasets.

B. Evaluation of the proposed methods

Out of all the methods we proposed, the approaches in Method 1 which relied on IN Reference Vector and IN Reference Entropy to detect the anomaly by computing the similarity with each of these vectors showed consistently good performance for all three datasets. As inferred from Fig.5, the scores obtained with decision rules are found less than the average scores. Method 2 also gives considerably better performance than the decision rules.

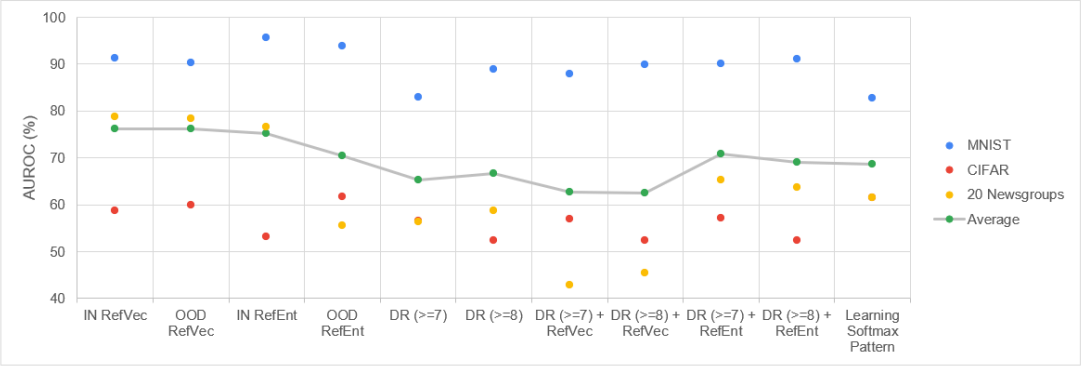

Fig. 5. Comparison of AUROC scores obtained from different approaches

The scores for IN Reference Entropy are the best with MNIST and 20 Newsgroups and the scores for OOD Reference Entropy are the best for CIFAR-10.

The Decision rule did not help any of the datasets much. Decision Rule with a count of 8 worked relatively worse for CIFAR-10 and while the one with a count of 7 was found worse for MNIST and 20 Newsgroups.

Comparing with the baseline scores, only the MNIST dataset was able to outperform the scores. Both the approaches exploiting the IN Reference Entropy and the combination of Decision Rule with Reference Entropy did better than the baseline approach by a margin of 7.4% and 2.9% respectively.

Table V represents the AUROC scores with baseline and reference vectors. It is clear from the figures that computing distance from both the reference vectors to distinguish OOD from IN distribution is not a good approach but considering the similarity with each of these vectors individually would yield better results. Though experiments with CIFAR-10 do not comply with this

TABLE V
METHOD 1 - WITH REFERENCE VECTORS AND REFERENCE ENTROPY

Dataset	Baseline	Dist RefVecs	Dist RefEnts	IN RefVec	OOD RefVec	IN RefEnt	OOD RefEnt
MNIST	88.14	84.98	88.47	91.22	90.85	**95.53**	94.56
CIFAR-10	**69.02**	62.09	62.14	58.63	59.94	53.23	61.69
20 Newsgroups	**90.98**	76.56	79.11	77.07	63.43	85.85	70.77

[1] Dist RefVecs - Distance from Reference Vectors
[2] Dist RefEnts - Distance from Reference Entropy values
[3] IN RefVec - Similarity with IN Reference Vector
[4] OOD RefVec - Similarity with OOD Reference Vector
[5] IN RefEnt - Similarity with IN Reference Entropy value
[6] OOD RefEnt - Similarity with OOD Reference Entropy value

finding, the differences in scores are negligible. It is also noteworthy that all three datasets work well with Reference Entropy values.

TABLE VI
METHOD 1 - DECISION RULE AND COMBINATIONS

Dataset	Baseline	DR count >= 7	DR count >= 8	DR + RefVec count >= 7	DR + RefVec count >= 8	DR + RefEnt count >= 7	DR + RefEnt count >= 8
MNIST	88.14	82.85	88.95	87.95	89.80	90.10	**91.04**
CIFAR-10	**69.02**	56.55	52.31	56.99	52.31	57.07	52.41
20 Newsgroups	**90.98**	61.61	67.75	62.89	69.09	76.30	76.63

[1] DR - Decision Rule
[2] RefVec - Reference Vector
[3] RefEnt - Reference Entropy

Table VI shows the AUROC scores with decision rule and combination of decision rule with Reference Vectors and Reference Entropy values. Having a hard-coded prediction count of 7 or 8 does not look feasible to either CIFAR-10 or 20 Newsgroups. However, MNIST surprisingly outperforms the baseline scores when the decision rule is coupled with Reference Entropy values.

TABLE VII
METHOD 2 - SOFTMAX PATTERN LEARNING

Dataset	Baseline	Average AUROC score
MNIST	**88.14**	82.97
CIFAR-10	**69.02**	62.41
20 Newsgroups	**90.98**	68.52

With Method 2, the trained SVMs were not able to learn or differentiate between the IN and OOD softmax patterns. This is evident from the poor AUROC scores we obtained, as shown in Table VII.

TABLE VIII
INFLUENCE OF ACCURACY ON AUROC SCORES

	Train accuracy	Validation accuracy	Baseline AUROC	Best AUROC from methods
with small-BERT	97.18	88.59	80.63	76.22
with large-BERT	99.97	93.37	90.98	86.56

Our experiments with different models for 20 Newsgroups - one with small BERT (4 transformer blocks, a hidden size of 512, and 8 attention heads) and one with the large BERT (12 transformer blocks, a hidden size of 768, and 12 attention heads), we could see how the accuracy of the trained model would affect the baseline AUROC score and the scores we obtain from our proposed methods. Table VIII shows that as the accuracy of the model increases, better are the AUROC scores obtained. This might possibly be because of the distinguishing capability the model would muster up with better validation accuracies.

VII. CONCLUSION

Though our experiments are not exhaustive mainly due to the limited computational resources, we attempt to answer the two questions we started with, with the obtained figures. From our experiments, it is observed that only the MNIST dataset

outperforms the baseline scores. This prompts us to conclude that training an ensemble of classifiers with leave-out classes need not necessarily work well for all datasets. However, all three datasets show a common trend of giving good AUROC scores when we exploited entropy values for IN and OOD data. Also, it is worth noting decision rules based on hardcoded threshold counts would not work well. Instead, a binary classifier trained on the softmax distributions should help better for distinguishing instances the model was trained on from those which were not. Nevertheless, different distribution of IN and OOD data, with a different number of leave-out classes, trained on better architectures should be experimented in order to arrive at a concrete conclusion on the potential questions this topic poses.

REFERENCES

[1] D. M. Hawkins, *Identification of outliers*. Springer, 1980, vol. 11.

[2] A. Chiang and Y.-R. Yeh, "Anomaly detection ensembles: In defense of the average," in *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, vol. 3. IEEE, 2015, pp. 207–210.

[3] L. Rokach, "Ensemble methods for classifiers," in *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 957–980.

[4] A. Chiang, E. David, Y.-J. Lee, G. Leshem, and Y.-R. Yeh, "A study on anomaly detection ensembles," *Journal of Applied Logic*, vol. 21, pp. 1–13, 2017.

[5] Z. Zhao, "Ensemble methods for anomaly detection," Ph.D. dissertation, Syracuse University, 2017.

[6] H. R. Bonab and F. Can, "A theoretical framework on the ideal number of classifiers for online ensembles in data streams," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016, pp. 2053–2056.

[7] H. Bonab and F. Can, "Less is more: a comprehensive framework for the number of components of ensemble classifiers," *arXiv preprint arXiv:1709.02925*, 2017.

[8] C. C. Aggarwal, "Outlier ensembles: position paper," *ACM SIGKDD Explorations Newsletter*, vol. 14, no. 2, pp. 49–58, 2013.

[9] Z. He, X. Xu, and S. Deng, "A unified subspace outlier ensemble framework for outlier detection in high dimensional spaces," *arXiv preprint cs/0505060*, 2005.

[10] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *arXiv preprint arXiv:1610.02136*, 2016.