Anomaly detection in embedding space learned with a contrastive approach

Corentin Lamboley, Tristan Bitard-Feildel, Amélie Chérubin

Anomaly detection in embedding space learned with a contrastive approach

Corentin Lamboley
IA3D
DGA-MI
Rennes, France

Tristan Bitard-Feildel *
IA3D
DGA-MI
Rennes, France

Amélie Chérubin *
IA3D
DGA-MI
Rennes, France

***Abstract*—Confidence in data is critical to learn safe and reliable AI models. In this paper, we explore the ability of embedding space learned through contrastive training to capture outliers and labeling errors. Ideally, embedding space learned with a contrastive approach should enforce proximity of data points in embedding space. This property of the embedding should facilitate the application of anomaly detection methods to detect erroneous data points. The study focus on the CIFAR-10 dataset. Once the embedding learned, we evaluated the ability of the anomaly detection methods to identify correctly erroneous data points using controlled noise through label flipping. We tested several anomaly detection methods with different hyper parameters. The data separation per class are easily observed in the embedding space and class outliers can be identified, highlighting the presence of data point outside the domain distribution of each class. The embedding space is also resilient to noisy data, and the tested anomaly detection methods can capture data points not corresponding to the classes used to learn the embedding. This preliminary work shows the potential growth of embedding space learned with unsupervised method to capture outliers and preprocess data.**

Keywords—data confidence, anomaly detection, self supervised learning, contrastive loss

I. Introduction

When performing a classification task, the maximum achievable accuracy depends on the quality and the quantity of data and on the appropriateness of the chosen learning algorithm [4]. Large datasets have enabled deep neural networks to achieve remarkable success in various domains, but this success is highly dependent on the quality of the training labels, which can be extremely very to create manually as datasets grow, especially when domain expertise is required.

Consequently, substitution labeling techniques have been developed such as crowdsourcing [1], model-assisted labeling, active-learning [2] and weak supervision [3]. Nevertheless, these methods inherently generate noisy labels decreasing the performance of deep neural networks, since they have high capacities and tend to overfit to these noisy labels, resulting in poor generalization performance.

To tackle this, methods have been developed to identify mislabeled data [4, 5] or to be robust to label noise [6, 7, 8, 9]. In particular, self-supervised contrastive learning [10] demonstrated great results in the past few years, and turned out to achieve good performance even in the presence of label noise [11]. Contrastive learning discards all the labels and learns representations by maximizing similarity between differently augmented views of the same data point via a contrastive loss in the latent space. A simple fully connected network can be trained on these representations in a supervised way, given the (potentially noisy) labels, and shows label noise robustness.

In this work, we present methods for outliers and errors detection, as well as learning with noisy labels techniques using representations from contrastive algorithms. Our contributions are two folds.

As a preliminary step, we evaluated the quality of learned representations with a Contrastive Learning approach, and their ability to provide label noise robustness. Second, we evaluated several anomaly detection methods on the learned representations. Anomaly detection algorithms are highly sensitive to the intrinsic data structure, and finding an appropriate method working with learned representations is a challenging task. Such, we investigated the relationship between incorporating noise and performances of anomaly detection algorithms.

II. Preliminaries

A. Contrastive Learning

The goal of contrastive learning methods is to learn an embedding space in which similar sample pairs stay close to each other while dissimilar ones are far apart. A positive pair consists of two different points of view of the same example, obtained using data augmentation (e.g. Random cropping or Color distortion for image data).

Given this new dataset of augmented data, an encoder extracts features and creates an embedding for each example of each pair. The ResNet model is usually the choice for the image encoder, and the final goal of the contrastive approach is to find the correct weights for the encoder to match similar pairs together. The encoder creates representations, and in practice, we add a Projection Head to map the representations h_i to a lower dimensional vector z_i before calculating the contrastive loss.

This loss function tends to maximize the similarity between embedding for positive pairs and minimize it for negative pairs. A classical choice is the Noise Contrastive Estimation (NCE) defined as [24]:

$$l_{i,j} = -log \frac{\exp\left(\frac{z_i^T z_j}{\tau}\right)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp\left(\frac{z_i^T z_k}{\tau}\right)} \quad (1)$$

for the positive pair of augmented samples (i, j), where z_i is the output of the projection head of the augmented sample i, $\mathbb{1}_{[k \neq i]}$ is the indicator function and τ the temperature parameter. We compute the final loss across all positives pairs. Fig. 1 provides an example of the full process of this algorithm.

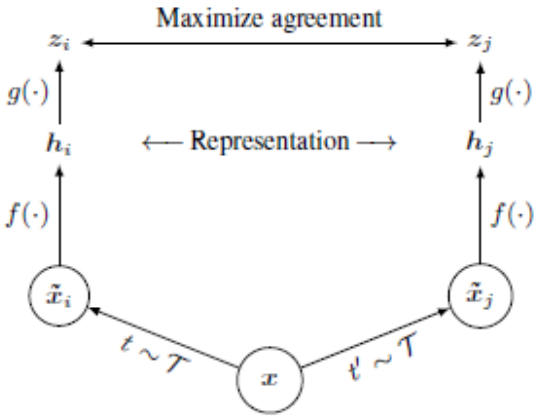

Fig. 1. Simple representation of the contrastive learning algorithm. Two different data augmentations (t and t' from the same family of augmentations) are applied to each data example to obtain two correlated views. The encoder $f(.)$ and the projection head $g(.)$ are trained to minimize the contrastive loss. After the training is completed, we discard the projection head and keep the encoder and representations h for downstream tasks.

B. *Label noise generation*

For our experiments, we use the CIFAR10 dataset and follow the same protocol as [12]. For label noise, we work with asymmetric label noise, where noisy labels are obtained by randomly flipping the true labels. This mimics real-world noise.

Let y^* be the unknown, true, uncorrupted label and $\tilde{y}$ the observed, noisy label. We generate noisy data from clean data by randomly switching some labels of training examples to different classes non-uniformly according to a randomly generated noise transition matrix $Q_{\tilde{y}|y^*} \coloneqq p(\tilde{y} = i|y^* = j)$, $\forall i, j \in [m]$ the set of unique class labels. Noise transition matrices have different traces for different noise levels, and have different sparsities. See Appendix A for the noise matrices used in our experiments.

Definition (sparsity). A statistic defined by the fraction of zeros in the off-diagonals of $Q_{\tilde{y}|y^*}$. A high sparsity quantifies non-uniformity of label noise, common to real-world datasets. Zero sparsity means that every noise rate in $Q_{\tilde{y}|y^*}$ is non-zero, while a sparsity equal to one means that there is no noise because all off-diagonals of $Q_{\tilde{y}|y^*}$ are zeros.

C. *Outliers detection*

Outliers are items in a dataset that are rare and deviate from the general data distribution. It is as if they were generated by a different mechanism. The detection of outliers is an active area of research in the field of data mining, and has various practical applications such as credit card fraud detection, telecom or medical analysis. In particular, classification algorithms mislabeled instances can be considered outliers and be removed from the training set to improve the performance of the classifier.

There are a great number of different outlier detection algorithms, and we propose to use the following ones in our future experiments:

- Auto-Encoder: A fully connected neural network that uses the reconstruction error as the outlier score [23].
- Isolation Forest: An ensemble-based method that isolates anomalies using tree classifiers [15].
- Average/Median KNN: A simple proximity-based method using the average or median of the k nearest neighbors as the outlier score [21].
- LOF: "Local Outlier Factor" An algorithm also using the k nearest neighbors to estimate density. The samples with low densities are considered outliers. [17].
- CBLOF: The cluster-based version of LOF that can use any clustering algorithm [18].
- KDE: "Kernel Density Estimation" A well-known method using kernels with a particular bandwidth to estimate the probability density function [16].
- PCA: "Principal Component Analysis" A method for outliers' detection using a Principal Component Classifier [20].
- COPOD: "Copula-based Outlier Detection" that constructs an empirical copula to predict tail probabilities to determine the level of "extremeness" of a sample [19].
- ECOD: "Empirical Cumulative Outlier Detection" using the empirical cumulative distribution per dimension of the data to predict tail probabilities across dimensions [22].

III. Experiments

A. *Exploring label noise robustness and representation's quality*

In this experiment, we first demonstrate how Contrastive Learning improves label noise robustness, even when using sub-optimal hyperparameters to generate representations in the latent space. Moreover, we evaluate the algorithm's ability to separate samples into relevant clusters, and the side effects of generating representations in an unsupervised way.

B. *Finding labelling errors using anomaly detection*

If Contrastive Learning provides a certain robustness against label noise, then it also allows for the discovery of labeling errors in the latent space of representations.

First, a sample's representation created by the encoder tend to be close to similar samples representations and far from others, independently of the associated noisy label. Therefore, in the latent space, samples are globally regrouped into clusters corresponding to their true label.

This has the consequence for a mislabeled example to be far away from the cluster associated with its true label. To some extent, a labeling error can be seen as an outlier in the distribution of representations of its associated class. Given that, it is possible to use outliers' detection methods on the latent space to detect labeling errors. Section IV.C benchmarks the best outliers' detection methods described earlier for this purpose.

The first problematic in dealing with outlier detection algorithms is the way of measuring their performance. In most cases, these algorithms generate an anomaly or outlier score that does not correspond to a probability of being an outlier.

Hence, considering samples with a score higher than 0.5 as outliers may not work well.

Yet, without knowing the threshold to use for a particular method that separates well errors from clean samples, it is still possible to speculate how good this method can be at the task of separating the distribution of errors and clean samples. To do so, we will use AUC ROC scores.

The process to test out an outlier detection method is the following:

- The dataset is the representations of the 50000 CIFAR10 images generated with the encoder, with the labels flipped according to a noise matrix.
- Class by class, we use the algorithm to assign an outlier score to every sample belonging to this particular class, and the scores are normalized. A sample belongs to a class if its noisy labels correspond to this class. In theory, we consider that outliers are errors and should not belong to that particular class.
- Knowing both the true and noisy labels, we can calculate the AUC ROC of this algorithm for each class, and the global AUC ROC along all the samples.

Finally, we can compare the performance of many outlier detection methods with various hyperparameters, on different noise and sparsity levels, with data produced by Contrastive learning.

IV. Results

A. Robust training under label noise on CIFAR10

For the first experiment, we use contrastive learning with a ResNet18 encoder to produce representations of CIFAR10 images in a 512 dimensions vector, with a projection head of size 128 and a batch size of 256. We also choose the temperature parameter of the NCE loss to be 0.5, and we train for 600 epochs.

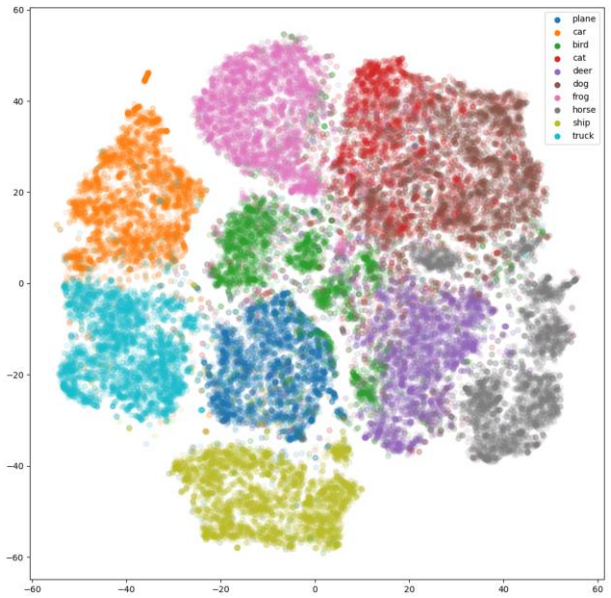

Fig. 2. The representations of CIFAR10 images obtained via contrastive learning, represented as 2 dimensions points thanks to t-SNE [13] dimensionality reduction. Each color correspond to a true label.

Noise	Sparsity							
	0.0	***0.1***	***0.2***	***0.3***	***0.4***	***0.5***	***0.6***	***0.7***
With Cross-Entropy loss								
0.0	90.72							
0.1	90.56	90.66	90.54	90.75	90.53	90.57	90.64	90.53
0.2	89.58	89.98	89.50	89.78	90.50	90.16	89.89	88.87
0.3	89.51	89.37	88.61	89.67	89.40	89.00	85.55	87.46
0.4	87.69	87.03	86.89	86.05	83.44	77.43	72.44	66.90
0.5	82.57	76.90	69.32	73.06	62.79	65.75	54.60	56.49
0.6	65.82	61.80	58.96	54.00	51.47	50.22	45.11	40.44
0.7	41.26	35.35	34.00	36.86	35.46	26.20	27.23	28.54
With ELR loss								
0.0	91.31							
0.1	91.02	91.22	91.00	91.04	91.01	91.10	90.90	90.95
0.2	90.51	90.63	90.63	90.63	90.57	90.54	90.69	90.64
0.3	90.16	90.02	90.41	90.43	90.70	90.21	89.91	90.29
0.4	89.80	88.79	87.15	89.85	85.75	82.42	79.96	71.97
0.5	85.78	80.83	68.11	73.55	64.44	74.17	46.91	56.21
0.6	70.89	73.67	57.21	64.93	42.39	57.36	40.77	40.24
0.7	42.43	36.48	29.71	42.23	40.52	31.17	23.40	21.10

Table 1. Top-1 accuracies of the MLP on different noise and sparsity levels, using the Cross-Entropy loss in the first table, and a noise robust loss (ELR) in the second table, with ResNet18 as encoder.

Fig. 2 shows that Contrastive Learning creates globally well separated representations of the CIFAR10 images in the latent space, except for the labels cats and dogs where there is no distinct separation. It also seems that the clusters are intuitively organized in the sense that animals are separated from vehicles, cars, and trucks are close to each other, as well as for cats and dogs, and in the middle, birds are not so far from planes.

For further experiments, we trained this model for 2000 epochs. t-SNE representations [13] and contrastive losses over epochs can be seen in Appendix B. We use the resulting ResNet18 encoder to represent every image of the CIFAR10 dataset in the latent space, creating a dataset of embeddings on which a simple Multilayer Perceptron (MLP) can be trained. As a first step, we used the Cross-Entropy loss and trained the network on the embeddings in a supervised way, given the noisy labels generated in II.B.

Furthermore, we also trained this network with the noise robust loss function called Early Learning Regularization (ELR) [9], which prevents memorization effect of neural networks by capitalizing on the early learning phase. The use of this loss significantly improve accuracy in almost all noise configurations compared with Cross-Entropy, as Table 1 shows. For comparison, we use as baselines a ResNet18 with Cross-Entropy loss, and a ResNet18 with ELR loss (See Appendix C).

As intended, training an MLP on top of the encoder significantly improves performance compared to the baselines approaches. One may also notice that the method is not robust to sparsity under high-level noise.

A promising future work would be to push the algorithm further by using deeper models, larger batches, and to train for more epochs to observe the resulting representations. This

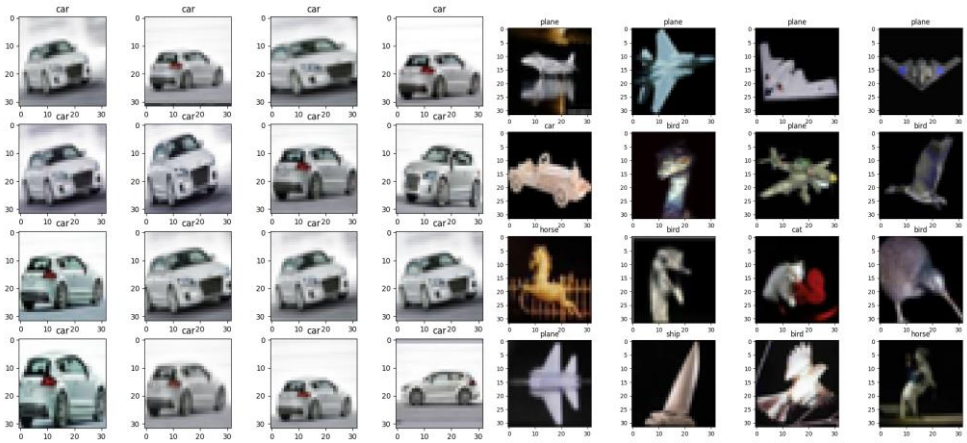

Fig. 4. On the left, 16 randomly chosen CIFAR10 images from the cluster circled in red in Fig. 3. On the right, 16 randomly chosen images from the cluster circled in Blue in Fig. 3.

should increase the accuracies, and may help for the errors' detection methodology described in III.B.

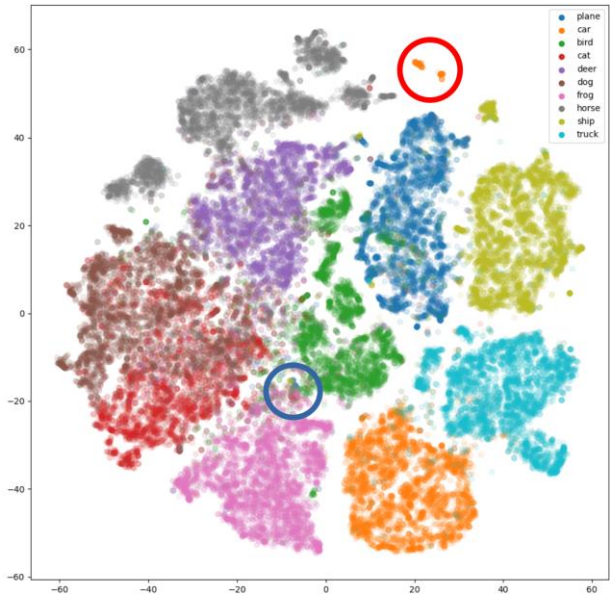

Fig. 3. The t-SNE representations of CIFAR10 images with ResNet18 encoder for 2000 epochs. Each color correspond to a true label.

B. *Clusterization of the latent space*

Fig. 3 presents the t-SNE representations of the CIFAR10 images, obtained via Contrastive Learning with a ResNet18 after 2000 epochs. The representations seem to separate classes, but to generate out-of-distribution clusters as well. In particular, we circled two remarkable small clusters. The red circle indicates an out-of-distribution cluster of cars, while the blue circle is a group of samples that are close to each other but do not belong to the same class. This could be a side effect of applying t-SNE for visualization, or an intrinsic behavior of the learned representations of the Contrastive Learning algorithm.

Fig. 4 shows the images inside the two clusters. We are now better able to understand how the model generates representations. The red cluster contains similar cars with the same color and the same background, and the blue cluster contains images from different classes on a black background.

Having a cluster of samples from the same class is not a problem since a classifier will easily match it to the associated label. However, clusters like the blue one raise a problem and can be hard to identify for a classifier.

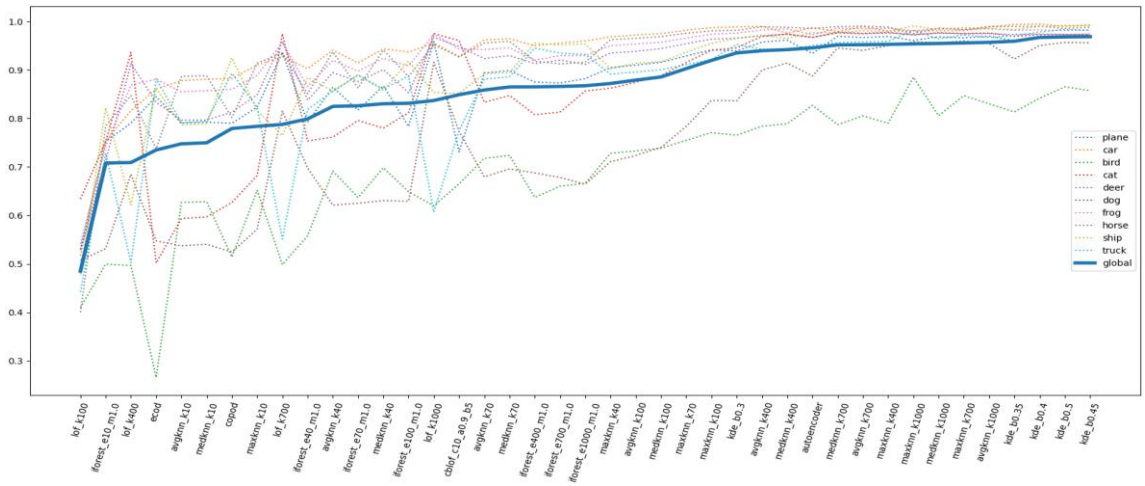

Fig. 5. AUC ROC scores with 30% noise and 20 % sparsity, for every outlier detection models. We compute the score for every class, and globally. The models are sorted by global AUC ROC to make the visualization clearer.

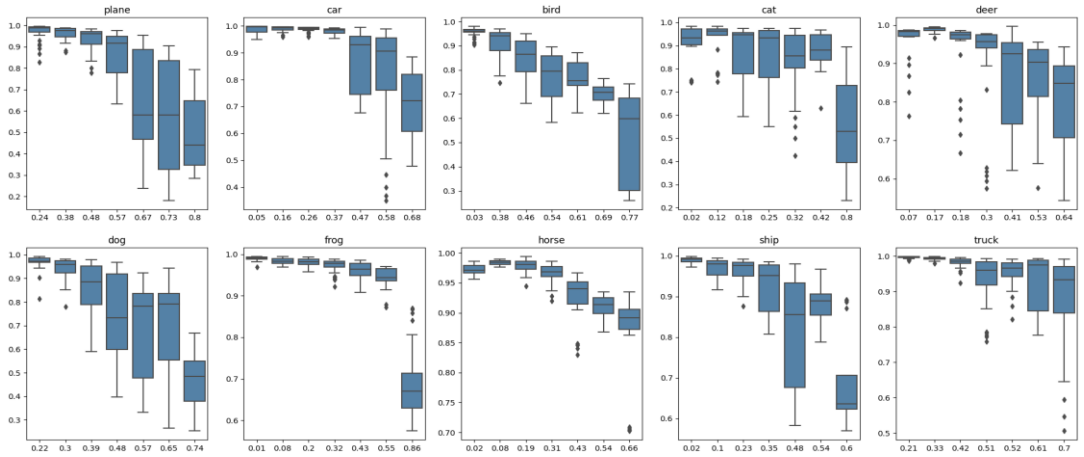

Fig. 6. Class by class distributions of AUC ROC scores computed by KDE models in all noise configurations. i.e. for the class "plane", the subplot on the upper left shows, for every noise rate of this class, the distribution of all the AUC ROC computed by KDE models (five models with various bandwidth parameter)

To overcome this phenomenon, an option is to change data augmentations applied to original samples. Adding stronger color distortion may help to avoid focusing on the background of certain images and improve performance. In general, Contrastive Learning is highly dependent on the data augmentations used [14], and the quality of the set of augmentations is very specific to the dataset itself.

C. Finding errors and outliers

Fig. 5 reports AUC ROC scores computed on the predictions of eight different outlier detection methods with various hyperparameters, on a particular noise and sparsity setting (0.3 / 0.2). We take the average scores across the five seeds to get a representative value, instead of over-interpreting scores from one noise configuration. Results with other noise and sparsity settings are in Appendix D.

The first observation we can make is that the Kernel Density Estimation (KDE) technique outperforms every other methods when comparing the global AUC ROC score (0.97), while keeping a relatively narrow distribution of scores along the ten classes. Still, we can see that it struggles more with the "bird" class. In contrast, KNN-based methods also perform well when the number of neighbors is high enough. For the "maxknn_k1000", which takes the maximum distance to the 1000 nearest neighbors in the class, the global AUC ROC is still high, but the scores of all classes are closer together. In other words, this method does not "sacrifice" any class.

Similarly, the Auto-Encoder performs well, even though we did not tune any hyperparameters. It might need some parameters tuning to perform to its full potential, but still got an AUC ROC around 0.95.

Conversely, the ECOD and COPOD methods do not perform well. The LOF model neither, whereas some combinations of hyperparameters made the IsolationForest algorithm reach an AUC ROC of 0.87.

To compare the difficulties in finding labeling errors inside the classes of CIFAR10, we took the five KDE models of various bandwidths and plotted in Fig. 6 the distributions of AUC ROC for all noises. Unsurprisingly, for all classes, the scores globally drop with the increase of label noise. What is more interesting is the disparity between the classes of when this drop occurs. This dissimilarity might come from a particular disposition of noise matrices, or from an intrinsic difficulty for the algorithm to detect labeling errors in this class. Plots using KNN-based models and all models are in Appendix D.

V. Conclusion

Data cleaning can be used as a preprocessing step to train robust AI models. In this paper, we investigate how to use embedding spaces learned through a contrastive learning approach to detect anomalous data. To evaluate several anomaly detection methods, we controlled the noise associated to each class.

First, we found that when the dataset contained labeling errors, classifiers trained on embedding spaces performed better than classifiers trained directly on the data input. It would be interesting to compute a confidence statistic from the embedding of a data point in future work. This confidence statistic could be used to weight the loss accordingly during the learning phase of the classifier.

Second, we can identify groups of data points inside the embedding space that share a strong intrinsic similarity with each other but are less similar to data points of their own class. Based on this observation, we compared the projection of data points with and without noise to evaluate anomaly detection algorithms. We observed that the performance of anomaly detection algorithms depends strongly on the initial parameters. The more robust methods scales with high hyperparameters values such as KNN or KDE (k=1000 or bandwidth=0.5). The performances also vary inter-classes, with "cat", "bird" and "plane" being the most difficult classes to separate from input noise.

Further work is required to evaluate how to train an embedding space using contrastive learning, taking into account the properties of the available dataset. Ideally, one should be able to choose the learning parameters based on properties such as batch size and embedding dimension to optimize the projection. We could test deeper models, such as ResNet50, to see if the performances of the anomaly detection methods applied to the embedding space scale by the size of the network. Similarly, large batch size might affect the outcome of the embedding space.

Another direction could be the addition of a constraint on the loss of the embedding space, such as a spherical embedding or decision boundary to learn a confidence score

associated with the data heterogeneity with respect to the dataset.

REFERENCES

[1] R. A. Krishna, K. Hata, S. Chen, et al. “Embracing error to enable rapid crowdsourcing”, in : Proceedings of the 2016 CHI conference on human factors in computing systems, p. 3167-3179, 2016.

[2] B. Settles, “Active learning literature survey”, 2009.

[3] A. Ratner, S. H. Bach, H. Ehrenberg, et al. “Snorkel: Rapid training data creation with weak supervision”, in : Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases, p. 269, NIH Public Acces, 2017.

[4] C. E. Brodley, M. A. Friedl. “Identifying mislabeled training data”, Journal of artificial intelligence research, vol. 11, p. 131-167, 1999.

[5] C. Northcutt, L. Jiang, I. Chuang. “Confident learning: Estimating uncertainty in dataset labels”, Journal of Artificial intelligence research, vol. 70, p. 1373-1411, 2021.

[6] H. Zhang, M. Cisse, Y. N. Dauphin, et al. “Mixup: Beyond empirical risk minimization”, arXiv preprint arXiv:1710.09412, 2017.

[7] B. Han, Q. Yao, X. Yu, et al. « Co-teaching : “Robust training of deep neural networks with extremely noisy labels”, Advances in neural information processing systems, vol 31, 2018.

[8] J. Li, R. Socher, S. C. Hoi. “DivideMix: Learning with noisy labels as semi-supervised learning”, arXiv preprint arXiv:2002.07394, 2020.

[9] S. Liu, J. Niles-Weed, N. Razavian, et al. “Early-learning regularization prevents memorization of noisy labels”, Advances in neural information processing systems, vol. 33, p. 20331-20342, 2020.

[10] T. Chen, S. Kornblith, M. Nozouri, et al. “A simple framework for contrastive learning of visual representations”, in : International conference on machine learning, PMLR, p. 1597-1607, 2020.

[11] Y. Xue, K. Whitecross, B. Mirzasoleiman. “Investigating why contrastive learning benefits robustness against label noise”, arXiv preprint arXiv:2201.12498, 2022.

[12] C. Northcutt, L. Jiang, I. Chuang. “Confident learning: Estimating uncertainty in dataset labels”, Journal of artificial intelligence research, vol. 70, p. 1373-1411, 2021.

[13] L. Van der Maaten, G. Hinton. “Visualizing data using t-SNE”, Journal of machine learning research, vol. 9, no 11, 2008.

[14] Y. Tian, C. Sun, B. Poole, et al. “What makes for good views for contrastive learning?”, Advances in neural information processing systems, vol. 33, p. 6827-6839, 2020.

[15] F. T. Liu, K. M. Ting, Z. H. Zhou. “Isolation forest”, in : 2008 eigth ieee international conference on data mining, IEEE, p. 413-422, 2008.

[16] L. J. Latecki, A. Lazarevic, D. Pokrajac, “Outlier detection with kernel density functions”, in : International workshop on machine learning and data mining in pattern recognition, Springer, Berlin, Heidelberg, p. 61-75, 2007.

[17] M. M. Breunig, H. P. Kriegel, R. T. Ng, et al. “LOF: Identifying density-based local outliers”, in : Proceedings of the 2000 ACM SIGMOD international conference on management of data, p. 93-104, 2000.

[18] Z. He, X. Xu, S. Deng. “Discovering cluster-based local outliers”, Pattern recognition letters, vol. 24, no 9-10, p. 1641-1650, 2003.

[19] Z. Li, Y. Zhao, N. Botta, et al. “COPOD: Copula-based outlier detection”, in : 2020 IEEE international conference on data mining (ICDM), IEEE, p. 1118-1123, 2020.

[20] M. L. Shyu, S. C. Chen, K. Sharinnapakorn, et al. “A novel anomaly detection scheme based on principal component classifier”, Miami univ coral gables FI dept of electrical and computer engineering, 2003.

[21] F. Angiulli, C. Pizzuti. “Fast outlier detection in high dimensional space”, in : European conference on principles of data mining and knowledge discovery, Springer, Berlin, Heidelberg, p. 15-27, 2002.

[22] Z. Li, Y. Zhao, X. Hu, et al. “ECOD: Unsupervised outlier detection using empirical cumulative distribution functions”, IEEE transactions on knowledge and data engineering, 2002.

[23] P. Vincent, H. Larochelle, Y. Bengio, et al. “Extracting and composing robust features with denoising autoencoders”, in : Proceedings of the 25th international conference on machine learning, p. 1096-1103, 2008.

[24] K. Sohn. “Improved deep metric learning with multi-class n-pair loss objective”, Advances in neural information processing systems, vol. 29, 2016.

VI. Appendix

A. *Noise matrices used in our experiments*

We generated noise matrices with noises in (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7) and sparsities in (0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7), Fig. 5 presents these noise matrices.

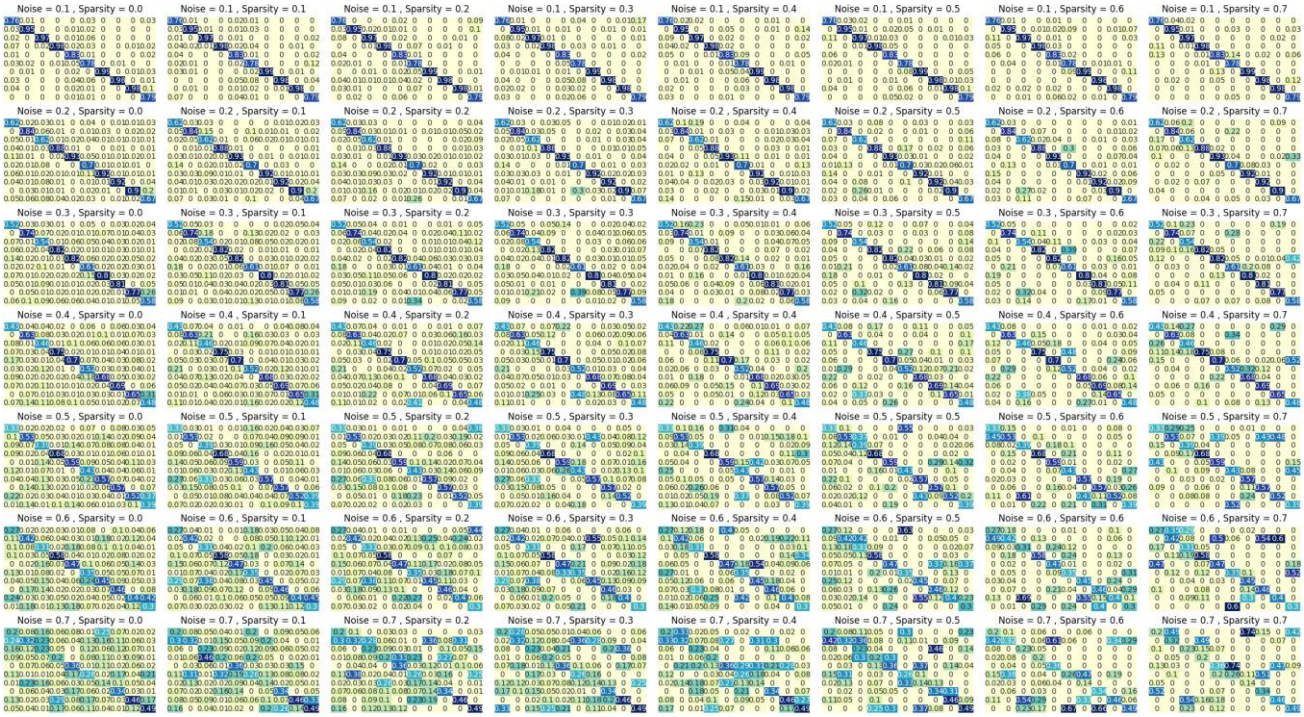

Fig. 7. Noise matrices with seed 0.

B. *Evaluation of Contrastive learning with ResNet18 for 2000 epochs and batches of 256.*

The loss functions shown in Fig. 8 tells that the model is starting to overfit on the training dataset, but the testing loss still decreases after 2000 epochs.

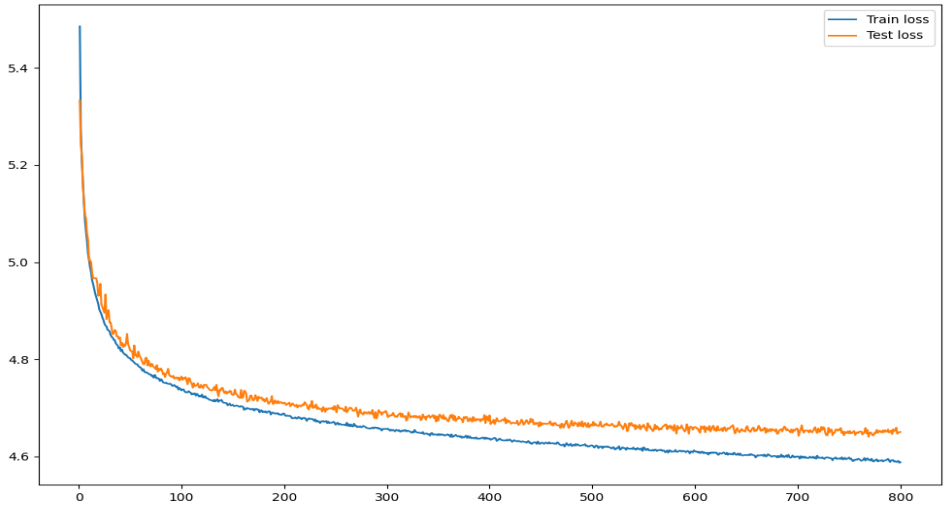

Fig. 8. Evolution of the loss function of Contrastive learning with ResNet18 encoder with batches of 256 over 2000 epochs.

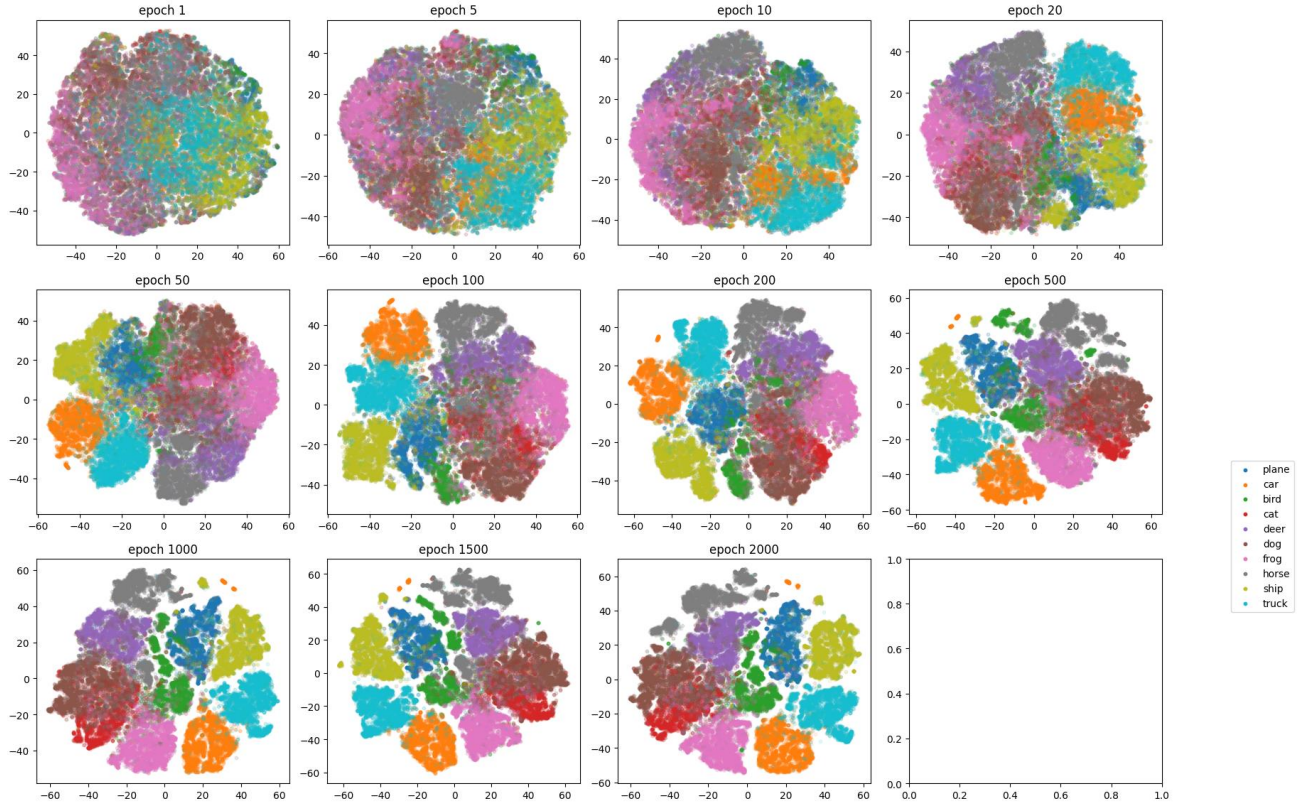

Fig. 9. Evolution of the loss function of Contrastive learning with ResNet18 encoder over 2000 epochs.

Fig. 9 shows the evolution of the t-SNE representations, and we can clearly see how the Contrastive Learning algorithm is splitting the dataset of representations into more and more clusters that globally correspond to their true class.

C. Results of experiments with baseline ResNet18, with Cross-Entropy and ELR loss.

Noise	**Sparsity**							
	0.0	***0.1***	***0.2***	***0.3***	***0.4***	***0.5***	***0.6***	***0.7***
Baseline ResNet18 with Cross-Entropy loss								
0.0	87.64							
0.1	79.88	81.56	79.58	79.26	81.02	80.56	81.25	81.32
0.2	79.15	76.46	78.90	77.67	79.14	76.97	76.22	78.45
0.3	71.76	74.16	72.34	76.00	74.33	75.88	73.56	72.08
0.4	70.71	68.81	66.30	69.00	69.64	65.80	69.59	61.68
0.5	58.45	59.27	60.31	57.91	55.09	59.50	48.61	54.90
0.6	52.77	48.44	42.93	47.79	41.89	48.56	39.18	42.37
0.7	32.20	34.81	35.60	33.46	30.76	31.85	27.89	30.41
Baseline ResNet18 with ELR loss								
0.0	88.44							
0.1	87.31	87.47	86.16	86.79	87.32	87.65	87.54	87.32
0.2	84.65	85.50	85.76	85.13	86.30	86.17	86.10	86.69
0.3	82.62	81.76	83.70	82.96	83.94	84.78	83.33	84.95
0.4	76.80	77.13	74.50	77.62	72.79	71.22	67.19	67.35
0.5	65.01	64.01	60.83	62.58	48.84	62.30	44.13	49.56
0.6	52.29	51.90	47.92	53.40	37.78	47.91	38.25	32.60
0.7	33.16	33.67	28.80	32.97	33.68	27.65	25.90	24.86

D. Results of Outlier detection : Distributions of AUC ROC

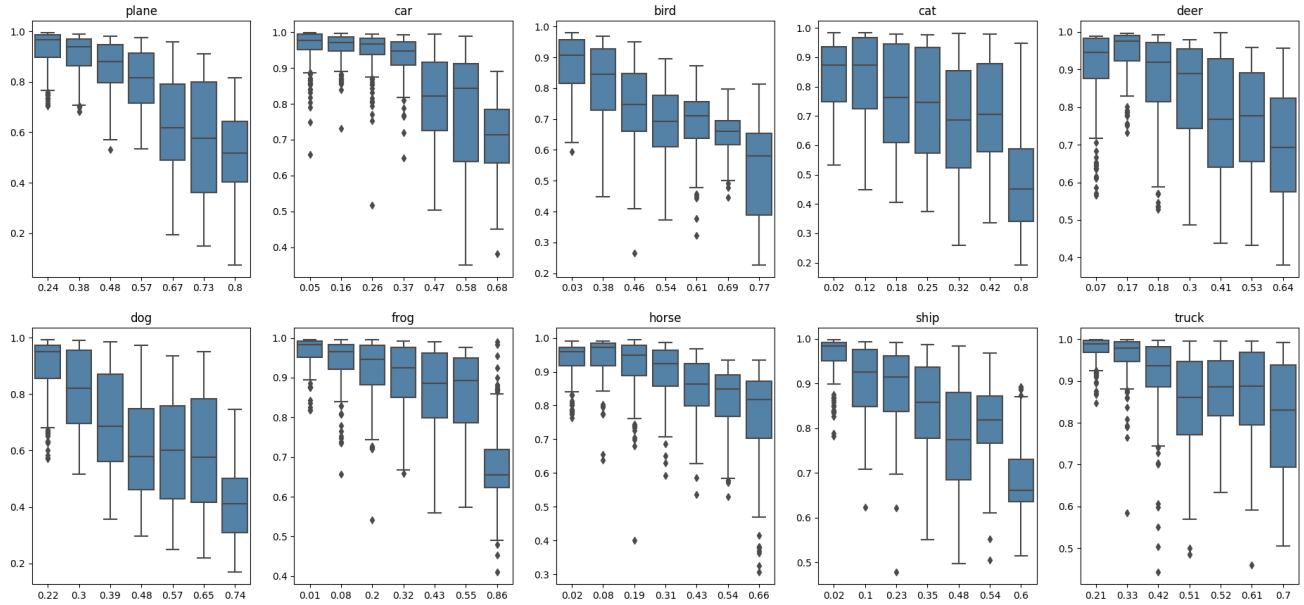

Fig. 10. Class by class distributions of AUC ROC scores computed by all models in all noise configurations

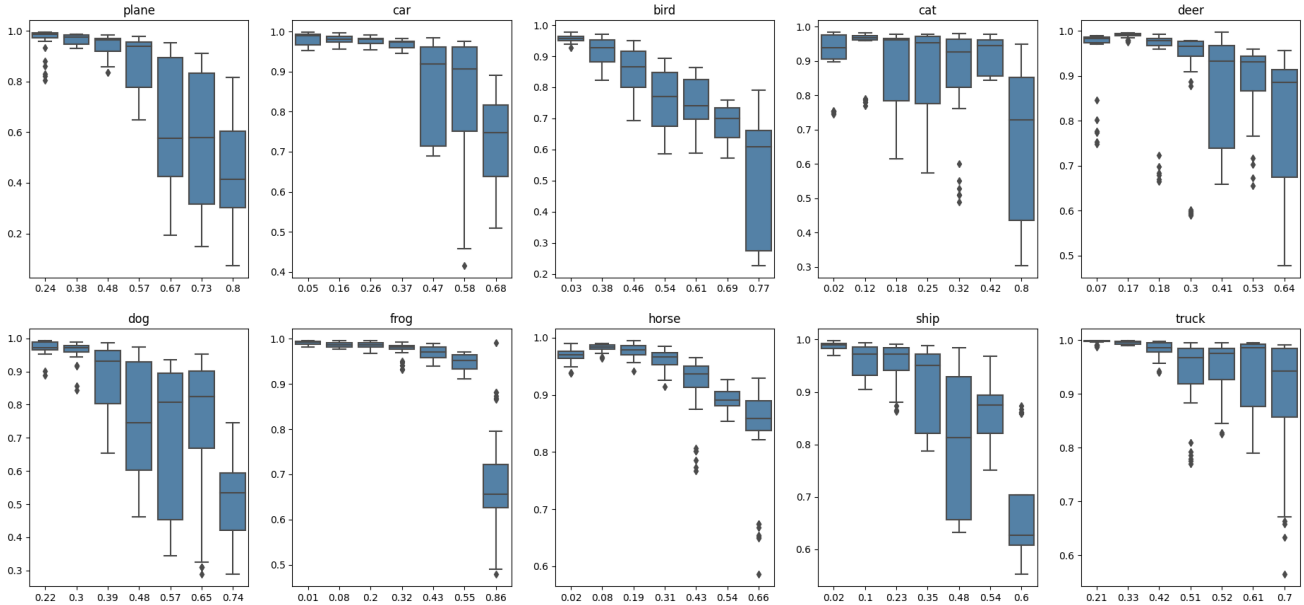

Fig. 11. Class by class distributions of AUC ROC scores computed by KNN-based models in all noise configurations