Anomaly detection using self-supervised point clouds

Agafonova Ekaterina [1] **Volkov Dmitry** [1] **Sidnov Kirill** [1] **Dembitskiy Artem** [1]

Abstract

This work was made an attempt to construct a robust pipeline for anomaly detection using self-supervised learning. This goal was achieved via train SSL architectures of SimCLR and Barlow Twins on one-class anomaly detection case and comparing structures of normal and anomaly object's point clouds of embeddings applying Hausdorff distance and Manifold topology difference metrics.

GitHub repo: GitHub
Video presentation: YouTube

1. Introduction

Self-supervised learning (SSL) aims to learn useful representations of the input data without relying on human annotations. Recent advances in self-supervised learning for visual data show that it is possible to learn self-supervised representations that are competitive with supervised representations. A common underlying theme that unites these methods is that they all aim to learn representations that are invariant under different distortions (also referred to as "data augmentations"). This work is dedicated to testing of hypothesis whether the structures of point cloud produced from the SSL embeddings allows finding anomalies in the image data.

The main contributions of this work as follows:

- Two architectures of unsupervised models namely SimCLR and Barlow Twins were adapted and trained on CIFAR10 dataset.
- Based on the trained models were implemented code for embeddings clouds construction.
- Were selected and evaluated two metrics capable to compare structures of build point clouds and measure the difference between clouds representations.

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Dmitry Volkov <dmitry.volkov@skoltech.ru>.

Final Projects of the Machine Learning 2020 Course, Skoltech, Moscow, Russian Federation, 2020.

2. Related Work

Two main direction of this project include: 1. Development of SSL model capable to learn embeddings which are invariant to distortions of the input sample. 2. Derive a methodology how to construct point clouds for further anomaly detection from obtained image representations.

The task of anomaly detection in an unlabelled dataset has many applications to a wide variety of data. In this work SSL methods are considered as an implementations aimed to generalize mutual information on the images representations to distinguish a certain CIFAR10 classes (e.g. cats, dogs, trucks, etc.). Such a task is to obtain a parametric encoder capable of extracting the most useful representation of the data. Methods for solving this problem are based on two main approaches: generative and discriminative. Generative methods (Hinton & Teh, 2006) (Kingma & Welling, 2013) (Goodfellow & Bengio, 2014) learn to model pixels in the input space. However, this approach is very resource intensive. Discriminative models (Chen et al., 2020) (Zbontar et al., 2021) (He et al., 2019) (Caron & Douze, 2018) (Grill et al., 2020) (Chen & He, 2020) learn representations that can generalize information about samples based on contrastive learning.

A popular approach to SSL is to train a pair of networks on distinct image representations. In this approach, the task of both networks is to maximize the agreement between input representations in order to obtain a new generalized description of objects.

Since 2020, several interesting methods have emerged that have become popular and have generated competition in the domestic field of contrastive learning. One of the most prominent representatives of paired methods are "SimCLR" (Chen et al., 2020), "BarlowTwins" (Zbontar et al., 2021), "SwAV" (Caron & Joulin, 2020), "SimSiam" (Chen & He, 2020), "BYOL" (Grill et al., 2020).

SimCLR. In a Simple framework for Contrastive Learning of visual Representations ("SimCLR") two separate data augmentation operators are sampled from the same family of augmentations and applied to each data example to obtain

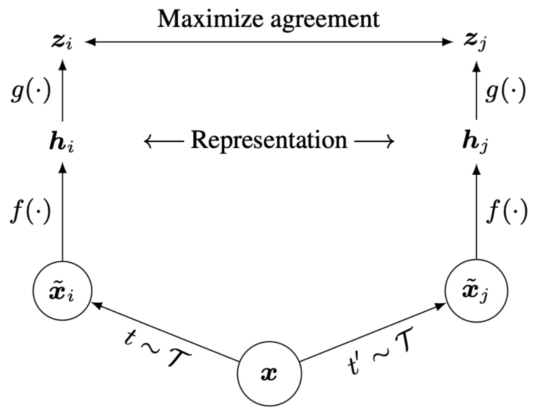

Figure 1. Sketch of SimCLR architecture (Chen et al., 2020)

two correlated views (Figure 1). A base encoder network and a projection head are trained to maximize agreement using a contrastive loss. After training is completed, the projection head is thrown away and encoder and representation are used for downstream tasks (Chen et al., 2020).

Barlow Twins. The objective function in "Barlow Twins" measures the cross-correlation matrix between the embeddings of two identical networks fed with distorted versions of a batch of samples, and tries to make this matrix close to the identity (Figure 2). This causes the embedding vectors of distorted versions of a sample to be similar, while minimizing the redundancy between the components of these vectors. "Barlow Twins" is competitive with state-of-the-art methods for self-supervised learning while being conceptually simpler, naturally avoiding trivial constant (i.e. collapsed) embeddings, and being robust to the training batch size (Zbontar et al., 2021).

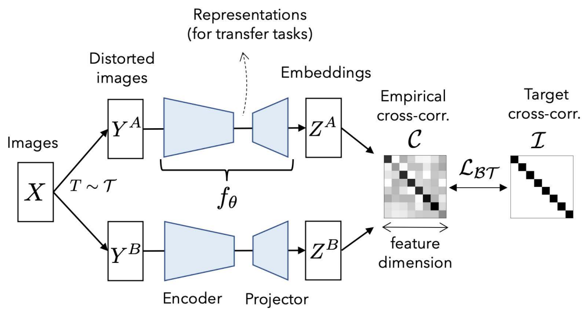

Figure 2. Barlow Twins architechture (Zbontar et al., 2021)

Model selection. Among the various architectures, we have considered "SimCLR" as one of the best implementations, as well as "Barlow Twins" as one of the newest compared to all the others. In addition, they are interesting in that they are somewhat similar. The architectures of both models are simple and differ from all others in that they do not use: - Predictor network - Stop gradient operation - Momentum encoder - Non-differential operations (clustering)

In paired networks, however, there is a problem of loss function collapse, when one of the networks learns to produce a constant representation that satisfies a given loss function (for example, cosine similarity), but such a representation may not carry any information about the object under study. All of the above methods of contrastive learning are designed in such a way as to circumvent this problem. In the architecture of "SimCLR" as well as of "Barlow Twins", this problem is solved relatively simply and in an original way through a modification of the loss function. But it is precisely in the loss functions that their main difference lies. In "SimCLR", loss function is based on computing a modified contrastive loss cosine similarity of the representations in the embedding space, which instead of simply achieving maximum agreement between representations introduces some mismatch to overcome the collapse problem (Chen et al., 2020). Whereas in "Barlow Twins", the loss function is based on the fact that the empirical cross correlation metric for the obtained features in the embedding space should be as close as possible to an identity matrix (Zbontar et al., 2021).

Finally, considered as a new method, "Barlow Twins" demonstrates performance on ImageNet (linear evaluation) using ResNet-50 between such popular methods as "BYOL" and "SimCLR". And since it is the newest among the presented methods, it is also of interest for testing in various conditions and to compare this model with the closest to it "SimCLR".

Hausdorff distance. The Hausdorff distance (HD) between two point sets is a commonly used dissimilarity measure for comparing point sets and image segmentations. Especially when very large point sets are compared using the HD, for example when the underlying applications are based on time critical tasks, like motion detection, then the computational complexity of HD algorithms becomes an important issue. It has efficient performance for large point set sizes as well as for large grid size; performs equally for sparse and dense point sets; and finally it is general without restrictions on the characteristics of the point set. However, it is worth noting that this metric can give unacceptable results if augmentations include rigid nonrigid transformations (Taha & Hanbury, 2015).

MTopDiv. "MTopDiv" (Manifold Topology Divergence) is a framework for comparing data manifolds, aimed, in particular, towards the evaluation of deep generative models. This method is based on the Cross-Barcode(P,Q) tool, that, given a pair of distributions in a high-dimensional space, tracks multiscale topology spacial discrepancies between manifolds on which the distributions are concentrated. This algorithm scales well (essentially linearly) with the increase of the dimension of the ambient high-dimensional space. It is one of the first TDA-based practical methodologies that can be applied universally to datasets of different sizes and dimensions, it is domain agnostic and does not rely on pre-trained networks. Finally, we considered this method as it avoids problems with isometric transformations (Barannikov et al., 2021).

3. Methodology

3.1. Data

For the model training and further evaluation of anomaly detection effectiveness was used, CIFAR10 dataset.The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are, 50000 training images and 10000 test images. Usage of this dataset was mainly motivated by the limited amount of computational resource i.e. GPU memory, and necessity of fast model prototyping and multiple hypothesis testing (Krizhevsky, 2009).

3.2. Self-supervised models

Initial stage of the project might be determined as a problem of contrastive learning. The idea here is to train a self-supervised model (SSL) which will pull together the representations of the target image after augmentation. In other words, to learn embeddings which are invariant to distortions of the input sample. This work after creating of several views of an image via augmentation, during the training of the model we force similar views (originating from the same sample) to be close to each other in feature space, and respective different views (originating from different samples) to be far away. This task raises a great interest in the ML community in the last few years, though a number of frameworks were established for solving the described problem, namely: MoCo (He et al., 2019), SimCLR (Chen et al., 2020), BYOL (Grill et al., 2020), Barlow Twins (Zbontar et al., 2021).

In this work, we consider to use only two mentioned architectures, namely Barlow Twins and SimCLR (Chen et al., 2020). Both of them reported demonstrating a decent performance on various benchmark datasets (e.g. ImageNet, CIFAR10, MNIST) and does not require large batches of samples and though big amount of GPU memory. Usage of this architecture was done within a computer-vision framework for self-supervised learning Lightly (Igor Susmelj, 2020), which provide its optimized implementations.

Below, we will describe these models in more details.

3.2.1. SimCLR

SimClr (Simple Framework for Contrastive Learning) (Chen et al., 2020) might be classified as a contrastive learning approach. Its define positive and negative sample pairs which are treated differently in the loss function. General pipeline of SimCLR architecture presented in Figure 1. It might be described as follows: separate data augmentation operators are sampled from the same family of augmentations ($t\tau$ and $t_0\tau$) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g((\cdot)$ are trained to maximize agreement using a contrastive loss. After training is completed, we throw away the projection head $g((\cdot)$ and use encoder $f(\cdot)$ and representation h for downstream tasks.

3.2.2. Barlow Twins

Barlow Twins architecture use a principle called redundancy-reduction and uses an objective function which tries to make the cross-correlation matrix computed from twin embeddings as close to the identity matrix as possible. In the same manner as SimCLR Barlow Twins operates on a joint embedding of distorted images (Figure 2). More specifically, it produces two distorted views for all images of a batch X sampled from a dataset. The distorted views are obtained via a distribution of data augmentations.

In other words, Barlow Twins's objective function designed to measure the crosscorrelation matrix between the embeddings of two identical networks fed with distorted versions of a batch of samples, and tries to make this matrix close to the identity. This causes the embedding vectors of distorted versions of a sample to be similar, while minimizing the redundancy between the components of these vectors. Barlow Twins is competitive with state-of-the-art methods for self-supervised learning while being conceptually simpler, naturally avoiding trivial constant (i.e. collapsed) embeddings, and being robust to the training batch size.

3.3. Augmentations

As far as we use self-supervised models as implemented in lightly repository, we use set of augmentation suggested by the authors of lightly for training of the model and implemented in their *ImageCollateFunction* (we use default parameters and input size of the CIFAR10 images - 32, the full list of augmentation can be found in lightly documentation [rerefence!]). These augmentations are adopted from the mentioned papers on SimCLR and Barlow Twins and include normalization, grayscale, jitter, gaussian blur and other random transformations as implemented in *PyTorch*. In the case of testing, we implemented two approaches of generating embeddings. The first is usage of the mentioned above default augmentations, while the first is the usage of the simple set of random augmentations from *PyTorch* in the following list: resize and crop, horizontal flip, colour jitter, and grayscale.)

3.4. Evaluation Metrics

After building a representation of data and corresponding point clouds of embeddings. It's necessary to deliver a way of their difference numerical evaluation, mainly answering the question if two point clouds distinguishable or not. According to this project goals we formulated following requirements for the metric: 1. Value of metric will allow measuring how our generated by SSL embedding representation stable to data argumentation. In other word from the view point of metric SLL capable to produce close enough embeddings to augmented images of the same class. 2. Numerical value of the metrics will allow distinguishing images of the different classes, i.e. images which we consider as an anomaly. 3. Metrics evaluation should not be computationally demanding.

Based on formulated requirements, for this work we have chosen two metrics, namely Hausdorff distance and a more sophisticated tool for a topology dissimilarity measurement - Manifold Topology Divergence (MTopDiv).

3.4.1. Hausdorff distance

Currently, one of the main application of the Hausdorff distance is image matching, used for instance in image analysis, visual navigation of robots, computer-assisted surgery, etc. Basically, the Hausdorff metric used to check if a template image is present in a test image; the lower the distance value, the best the match. That method gives robust results, even in presence of noise or occlusion (when the target is partially hidden) (Dubuisson & Jain, 1994). Of course usage of this metric is not limited by the mentioned above applications and might be adopted for the problem of matrices or point clouds comparison (Maiseli, 2021). By definition, Hausdorff distance - is the measure of how far two subsets of a metric space are from each other. It turns the set of non-empty compact subsets of a metric space into a metric space in its own right. Informally, two sets are close in the Hausdorff distance if every point of either set is close to some point of the other set. The Hausdorff distance is the longest distance you can be forced to travel by an adversary who chooses a point in one of the two sets, from where you then must travel to the other set. In other words, it is the greatest of all the distances from a point in one set to the closest point in the other set. More formally, Hausdorff distance from set P to set Q is a *MaxMin* function, defined as

$$h(P,Q) = \max_{p \in P} \left\{ \min_{q \in Q} d(p,q) \right\} \tag{1}$$

where p and q are points of sets P and Q respectively, and $d(p,q)$ is any metric between these points; for simplicity, we'll take $d(p,q)$ as the Euclidian distance between p and q.

It should be noted that Hausdorff distance is oriented (we could say asymmetric as well), which means that most of the time $h(P,Q)$ is not equal to $h(Q,P)$. This asymmetry is a property of *MaxiMin* functions, while *MinMin* functions are symmetric.

3.4.2. Manifold Topology Divergence

Manifold topology difference (Barannikov et al., 2021) - is novel tool based on the Cross-Barcode(Q, P) that, given a pair of distributions in a high-dimensional space, tracks multiscale topology spacial discrepancies between manifolds on which the distributions are concentrated (Khrulkov & Oseledets, 2018). Generally this metrics was designed to assess the performant of deep generative models in various domains. It was shown that metrics allows to accurately detect various degrees of mode-dropping, intermode collapse, mode invention, and image disturbance. It is important to note that the norm of $Cross - Barcode_i(P,Q), i \geq 0$, is bounded from above by the Hausdorff distance, which was mentioned in the previous section.

$$||Cross - Barcode_i(P,Q)|| \leq h(P,Q) \tag{2}$$

4. Experiments and Results

4.1. Experimental setup

All the experiments described in this work require extensive GPU usage. Proper experimental setup utilize GPU on stages of model training, embedding generation and further metrics calculations. For this purposes were used free GPU resource provided by Kaggle and Google Colab platforms with P100 Tesla and CUDA10.2 or higher. **Note** stage of MTopDiv library installation strictly require enabled GPU acceleration.

4.2. Model training

4.2.1. Resnet20

To construct a backbone for SimClr and Barlow Twins models, we used a custom implementation of resnet20 which differs from classical resnet (He et al., 2015) by adjusted number of parameters specially designed for CIFAR10 dataset. These model took as input 32x32 image and proceed it through the 20 residual blocks followed by 2D adaptive average pooling before passing into fully connected linear layer. Resnet20 architecture uses convolution with 3x3 kernels and padding 1, followed by batch normalization and ReLU activation functions. In total, this implementation has 0.27M parameters. As backbone were used only feature part

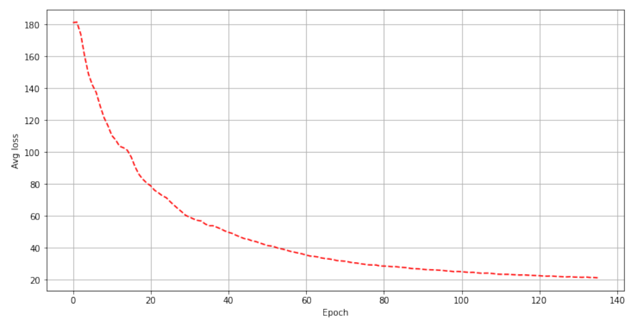

Figure 3. Training loss of BarlowTwins over 150 epochs.

of network without classification head. Implementation of this architecture provided to us by TA Nikita Balabin.

4.2.2. Barlow Twins

For Barlow Twins model were used, hyperparameters proposed in the paper. Maximum number of epochs were set to 150 with relatively small batch size of 32 images. Was used SGD optimizer with initial learning rate set to 0.06 and momentum value of 0.9. Minimum of loss function were achieved after 150 epochs as it is shown in training curve Figure3 further the performance on training set starts to degrade.

Training of the model took approximately 3 hours on GPU Tesla P100.

4.2.3. SimCLR

For SimCLR model, we decided to stick with hyperparameters comes from original work, though with lower batch size equal to 128 images for training speed up. Was used SGD (Ruder, 2016) optimizer with starting learning rate of $6 \cdot 10^{-2}$, momentum value of 0.9 and l2 regularization parameter set to $5 \cdot 10^{-4}$. Additionally, were applied cosine annealing scheduler for learning rate (Loshchilov & Hutter, 2016). As a backbone model for SimCLR we decided not to use standard pretrained implementations of resnet18/54, but to pass a custom resnet20 network especially designed for CIFAR10 dataset. Model convergence were achieved after 300 epochs of training and was not improved further, as it is shown in training curve Figure 4.

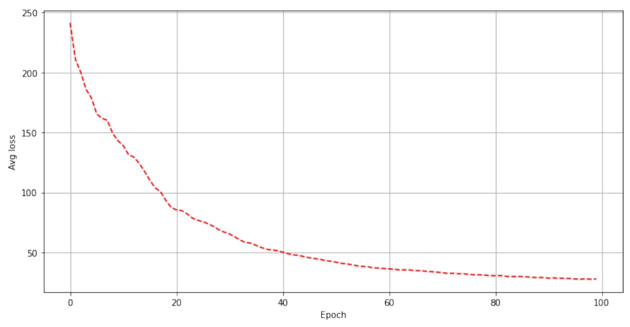

Figure 4. Training loss of SimCLR over 100 epochs.

Training of the model took approximately 3 hours on GPU Tesla P100.

4.3. Point Clouds Measurements

4.3.1. Bootstrapping procedure

For each image of class in the CIFAR10 test dataset and for each image from train dataset we constructed an embedding using one of the listed architecture (SimCLR and Barlow Twins) using mentioned augmentations. Hence, a matrix sized 1000 by 64 (5000 by 64 for train set) was obtained for each class (where 64 corresponds to the dimension of the embedding). After, to prove the network is able to distinguish normal images (images which the network was trained on) and abnormal images (the rest 9 classes) we compare the pointclouds from different classes constructed by performing the bootstrapping procedure. For this purpose 100 bootstrapped samples from the obtained matrices were treated as the pointcloud corresponding to the given class. This pointclouds were compared by general hausdorff distance metric and by MTopDiv with the analogues from the embeddings of the train dataset. The procedure was repeated 1000 times to construct representative distribution of the scores.

4.3.2. Hausdorff metric performance

When the score distribution is constructed we compare the histograms for train set vs. abnormal test set and train set vs. normal test set (which is cat images in our case) (fig. 5) to check if there are some differences between scores distributions obtained for (normal vs. normal) and (normal vs. abnormal) pointclouds.

In order to statistically evaluate the ability of the network to detect anomalies, we performed T-student test on the collected scores. According to the results, the SimCLR architecture, trained on the cats from CIFAR10 is able to treat airplanes and birds as anomalies. While horses are treated as normal i.e. their cloud representations lies close

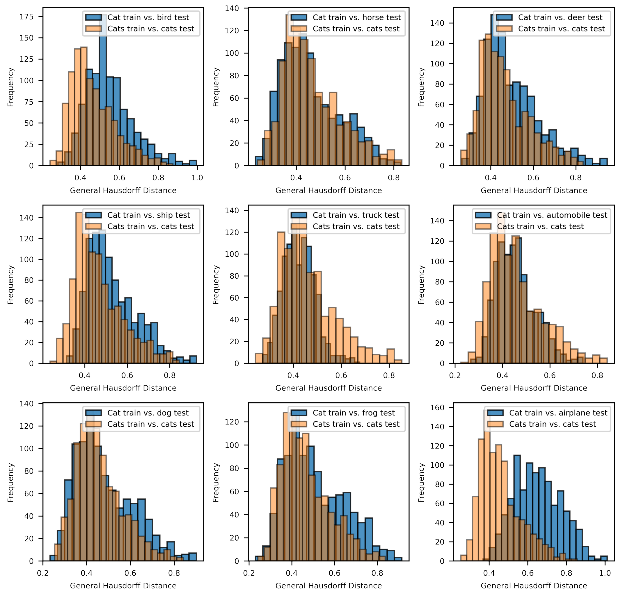

Figure 5. The distribution of Hausdorff distances between bootstrapped pointclouds of train and test datasets fed to Barlow Twins. To construct pointclouds we bootstrapped 100 samples from the datasets 1000 times.

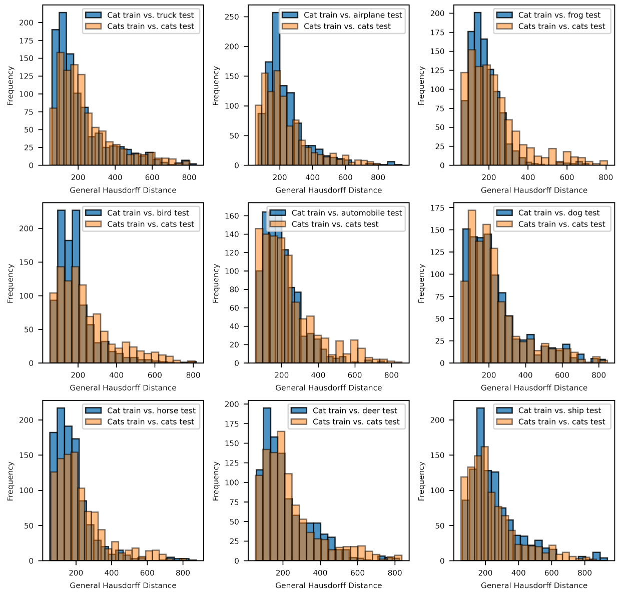

Figure 6. The distribution of hausdorff distances between bootstrapped pointclouds of train and test datasets fed to SimCLR. To construct pointclouds we bootstrapped 100 samples from the datasets 1000 times.

to normal class (cat). In table 1 we present Hausdorff scores for the pointclouds constructed from all the samples of each class. Here we calculate the scores between two clouds without using bootstrapping, i.e. we compare the whole 1000 by 64 matrices with each other. These scores can be interpreted as the evaluation of the distances between all embeddings of each class, i.e. the possibility to separate classes (normal and anomalies) in general. As per Hausdorff scoring, the performance of Barlow Twins is better than of SimCLR since more anomalies can be detected properly. Also, Barlow Twins network has the lowest scores for cats and dogs. It means, the constructed pointclouds for cats and dogs are almost indistinguishable, while ships or airplanes can be treated as anomaly.

Table 1. Hausdorff scores for SimCLR and Barlow Twins constructed pointlcouds

CLASSES	SIMCLR	TWINS
BIRD	349.2	537.1
HORSE	456.1	543.3
DEER	483.6	279.4
SHIP	413.3	**790.2**
CAT	423.0	205.6
TRUCK	458.0	306.7
AUTOMOBILE	412.4	499.6
DOG	**529.8**	162.7
FROG	406.3	465.4
AIRPLANE	390.3	553.2

4.3.3. MTopDiv metric performance

In the case of MTopDiv, we used less number of bootstrapped pointclouds (100) due to the computational resources required to get the scores in comparison to Hausdorff distances. The analysis of the score distribution presented at the figures 7 - 8 suggests that performance of Barlow Twins is better than of SimCLR.

Table 2. MTopDiv scores for SimCLR and Barlow Twins constructed pointlcouds.

CLASSES	SIMCLR	TWINS
DOG	0.21	0.12
FROG	0.12	0.48
AUTOMOBILE	0.21	0.44
DEER	0.29	0.29
HORSE	0.32	0.25
CAT	0.27	0.10
TRUCK	0.14	0.44
SHIP	0.32	0.59
BIRD	0.15	0.24
AIRPLANE	**0.75**	**0.69**

The same result was obtained after calculating scores for the

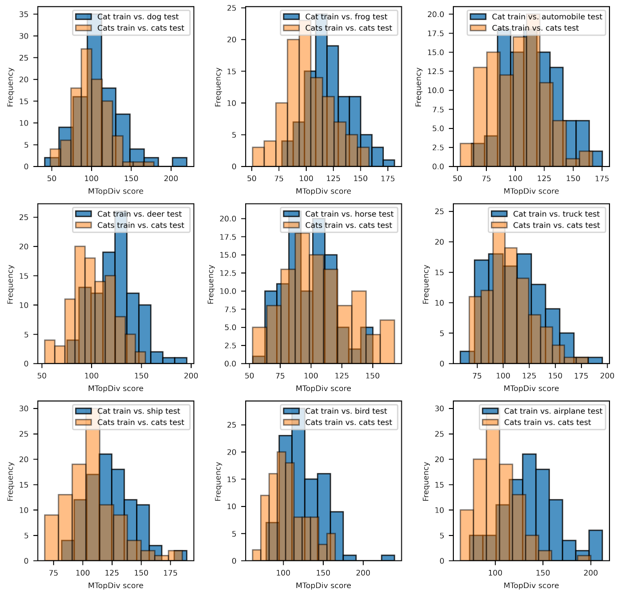

Figure 7. The distribution of MTopDiv score for bootstrapped pointclouds of train and test datasets fed to Barlow Twins. To construct pointclouds we bootstrapped 100 samples from the datasets 100 times.

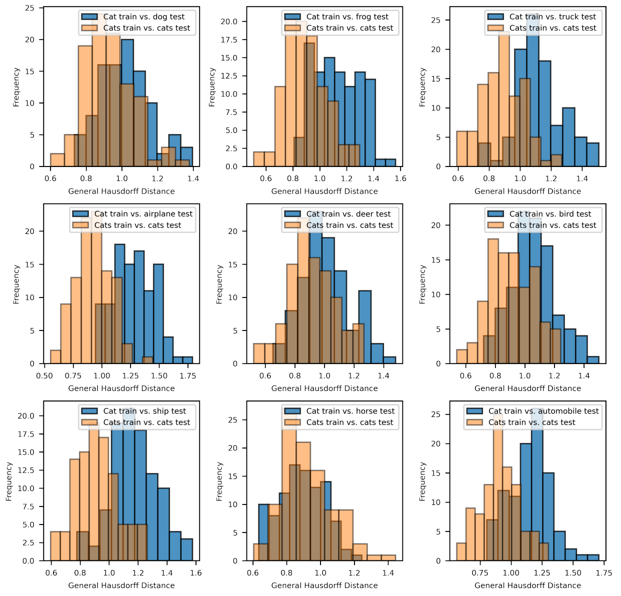

Figure 8. The distribution of MTopDiv score for bootstrapped pointclouds of train and test datasets fed to SimCLR. To construct pointclouds we bootstrapped 100 samples from the datasets 100 times.

whole test dataset of embeddings for normal and abnormal images without bootstrapping (table 2). As per calculations, the class of dogs and cats were treated by Barlow Twins as the normal class, while the rest classes has higher scores (0.10 - 0.12 vs. 0.24 - 0.69) and can be assigned to anomaly objects. The small difference in scores for cats and dogs can be explained by the similarity of these objects.

5. Conclusion and Future Work

In this work we have suggested the three-stage pipeline for anomaly detection using pointclouds of embeddings constructed by the self-supervised neural network. There are several pros and cons of the pipeline to be discussed. At the first stage we selected the architectures for the network, SimCLR and Barlow Twins as the most promising state-of-the-art models, and trained it on one class images using selected augmentations. We train the model on one class as if this class was the normal object, while the rest classes from the dataset were treated as anomalies. On the one hand, with this approach the network never seen the other classes, and they potentially will be better treated as outliers, on the other hand, the loss obtained during training on the one class images is difficult to interpret, and as it was shown in results section, the convergence of the loss does not guarantee the accurate separation of normal and abnormal objects. In this sense, the main disadvantage of the first stage of our pipeline is the training only on one class images. Probably, the more accurate approach, which can be used in the future work, is to perform training of the network on a both one-class and multi-class datasets and perform tests on some other dataset which was not fed to the network before. Such approach gives the option to estimate the accuracy of the network in advance, before trying to separate outliers and normal objects. At the second stage, we applied the trained backbones to construct the embeddings out of augmented images. We constructed the one hundred pointclouds each containing 100 points, where each point was corresponding to the embedding of the randomly sequentially augmented image. Each time we were comparing these embeddings of some random image from normal objects with embeddings of some random anomaly image. As it was shown, we were not able to properly construct normality/outlier threshold using selected metrics. At this point, it should be mentioned, that we did not perform ”grid search” on the optimal number of points in the point clouds to be compared. It means, that improvements of proposed anomaly detections pipline might be achived both from the viewpoint of architecture/training approach and the proper selection of the point clouds sizes or metrics for their comparison. In general we assume that this direction it rather promising and serves as a great field for further theoretical and experimental research.

References

Barannikov, S., Trofimov, I., Sotnikov, G., Trimbach, E., Korotin, A., Filippov, A., and Burnaev, E. Manifold topology divergence: a framework for comparing data manifolds. *CoRR*, abs/2106.04024, 2021. URL https://arxiv.org/abs/2106.04024.

Caron, M., B. P. J. A. and Douze, M. Deep clustering for unsupervised learning of visual features. *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

Caron, M., M. I. M. J. G. P. B. P. and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. *NeurIPS*, 2020.

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.

Chen, X. and He, K. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020.

Dubuisson, M.-P. and Jain, A. A modified hausdorff distance for object matching. In *Proceedings of 12th International Conference on Pattern Recognition*, volume 1, pp. 566–568 vol.1, 1994. doi: 10.1109/ICPR.1994.576361.

Goodfellow, I., P.-A. J. M. M. X. B. W.-F. D. O. S. C. A. and Bengio, Y. SGDR: generative adversarial nets. *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Grill, J., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. Á., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. Bootstrap your own latent: A new approach to self-supervised learning. *CoRR*, abs/2006.07733, 2020. URL https://arxiv.org/abs/2006.07733.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. B. Momentum contrast for unsupervised visual representation learning. *CoRR*, abs/1911.05722, 2019. URL http://arxiv.org/abs/1911.05722.

Hinton, G. E., O. S. and Teh, Y.-W. SGDR: a fast learning algorithm for deep belief nets. *Neural computation*, 18:1527– 1554, 2006. URL https://doi.org/10.1109/tpami.2015.2408351.

Igor Susmelj, Matthias Heller, P. W. J. P. M. E. e. a. Lightly. *GitHub. Note: https://github.com/lightly-ai/lightly*, 2020.

Khrulkov, V. and Oseledets, I. Geometry Score: A Method For Comparing Generative Adversarial Networks. *arXiv preprint arXiv:1802.02664*, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, 2009.

Loshchilov, I. and Hutter, F. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016. URL http://arxiv.org/abs/1608.03983.

Maiseli, B. Hausdorff distance with outliers and noise resilience capabilities. *SN Computer Science*, 2, 09 2021. doi: 10.1007/s42979-021-00737-y.

Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Taha, A. and Hanbury, A. SGDR: an efficient algorithm for calculating the exact hausdorff distance. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 37:2153–63, 2015. URL https://doi.org/10.1109/tpami.2015.2408351.

Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. Barlow twins: Self-supervised learning via redundancy reduction. *CoRR*, abs/2103.03230, 2021. URL https://arxiv.org/abs/2103.03230.

A. Team member's contributions

Ekaterina Agafonova (25% of work)

- Literature review on the metric for comparing point clouds
- Barlow Twins architecture, embeddings generation code
- Preparing the GitHub Repo

Kirill Sidnov (25% of work)

- Literature review on the related works
- Augmentation implementation
- Preparing the presentation

Dmitriy Volkov (25% of work)

- Literature review on the networks architecture
- SimCLR training on Kaggle
- Preparing the video
- Preparing the report

Artem Dembitskiy (25% of work)

- Literature review on the metric for comparing point clouds
- SimCLR architecture, embeddings generation code
- Code for plotting
- Preparing the report

B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   General comment: If the answer is **yes**, students must explicitly clarify to which extent (e.g. which percentage of your code did you write on your own?) and which code was used.

   Students' comment: We have used lightly implementation of the networks building blocks and layers and explicitly mentioned this in the text. The code for the construction of the point clouds and the construction of the distribution of the metrics scores as well as the plotting code was written by us.

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

4. A complete description of the data collection process, including sample size, is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   ☐ Yes.
   ☐ No.
   ☑ Not applicable.

   Students' comment: None

7. An explanation of how samples were allocated for training, validation and testing is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

9. The exact number of evaluation runs is included.

   ☐ Yes.
   ☐ No.
   ☑ Not applicable.

   Students' comment: None

10. A description of how experiments have been conducted is included.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None

11. A clear definition of the specific measure or statistics used to report results is included in the report.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None

12. Clearly defined error bars are included in the report.

    ☐ Yes.

- ☐ No.
- ☑ Not applicable.

Students' comment: None

13. A description of the computing infrastructure used is included in the report.

- ☑ Yes.
- ☐ No.
- ☐ Not applicable.

Students' comment: None