RE 'LEARNING AND EVALUATING REPRESENTATIONS FOR DEEP ONE-CLASS CLASSIFICATION' (ICLR2021)

Nikolaos Chazaridis, Marios Christodoulou, Ian Simpson
COMP6248 MSc students, Team K9DE, University of Southampton
{nc2n20,mc20g20,ijs1c20}@soton.ac.uk

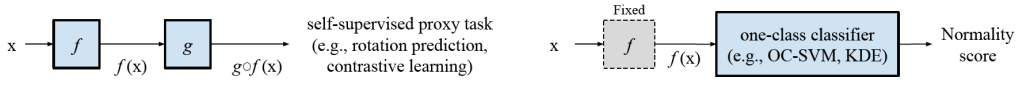
ABSTRACT

This report summarises COMP6248 Team K9DE's analysis of the reproducibility of ICLR 2021 paper 'Learning and Evaluating Representations for Deep One-class Classification' Sohn et al. (2021), including partial reproduction of results. A repository[1] of code used in the reproduction is provided.

1 INTRODUCTION

One-class classification problems permit training of models only on data belonging to the class of interest. The objective under testing is then for the model to classify whether a sample is a member of this class (an 'inlier'), or not (an 'outlier'). The main contribution claimed by the report is the utilisation of a two-stage framework for 'deep one-class classification', comprising of a self-supervised deep network trained on data $\boldsymbol{x}$ as $g(f(\boldsymbol{x}))$, thence in the second stage outputs are fed to a one-class classifier. The report focuses on ResNet18/ResNet50 architectures for the first stage, with the MLP classifier ranging in size of 1 to 8 layers, upon which one-class Support Vector Machines (OC-SVM) or kernel density estimators (KDE) are employed for the second stage. Further, the paper applies a novel contrastive loss function on this framework, which it claims yields State-of-the-Art results for one-class image classification problems.

This report is laid out in the order in which the reproducibility challenge was logically approached. First, the paper was assessed to determine what the scope ought to be, given time and resource constraints. Then, a methodology for detailed reproducibility was decided upon and experiments carried out. Finally, results were analysed and conclusions on reproducibility were made.



(a) Self-supervised representation learning (b) One-class classifier

Figure 1: Sohn et al. (2021) proposes a two-stage framework, whereby representations f learnt from a model $g(f)$ are classified in a second stage.

2 REPRODUCIBILITY AND EXPERIMENTAL SCOPE

2.1 GENERAL REPRODUCIBILITY OF THE PAPER

The paper reports the results of 118 unique model runs and shows plots of model performance on 150 unique configurations. Incorporating the appendices, there are over 500 unique model results in this paper, which by sheer volume alone presents a challenge for reproducibility.

The paper was generally well written, and many (although not all) experimental aspects were detailed. The report structure could benefit from a clearer flow and better clarity about which settings relate to which experiment. A table of parameters would have assisted with reproduction.

[1] K9DE reproduction repository:
https://github.com/COMP6248-Reproducability-Challenge/LRDOCC_reproduction

A GitHub code repository was provided and was claimed to contain all code needed to reproduce the experiments. As we decided upon independent authoring of PyTorch code, we did not directly employ this code, but we did run it once to verify it worked and were able to refer to it for reproduction guidance. The codebase is 19 *.py* files containing 3,594 lines of code. The scripts were moderately interdependent, only minimal documentation via the readme file and code commenting was provided. Such interdependency is to be expected of such a complex project, but documentation and code commenting is highlighted as an opportunity to improve reproducibility.

Two key report results require 2,048 epochs of ResNet50 training, requiring serious GPU capability to reproduce. Provision of checkpoint files might ease this burden: verification of convergence performance between two intermediate checkpoints would at least allow a 'gross check' of the method.

2.2 Specific Reproducibility of Experiments

The experiments presented were assessed for difficulty of reproduction. After reflection on the findings of Pineau et al. (2020), three metrics were formulated. The 'report rating' assesses 'how well does the report describe what tasks are required to recreate the experiment?', and the 'code rating' assesses 'how easy is it to identify and understand the code relevant to the experiment?', both with *Rating 1: 'well'; 2: 'adequately'; 3: 'insufficiently'*. The 'compute rating' assesses 'how much compute resource is needed to reproduce one 'headline' result for this configuration?', with *Rating 1: <1hr; 2: 1-12hr; 3: >12hr estimated processing time*. This scheme was specific to our setup as we used Google Colab Pro, and so assumed a Nvidia P100 GPU, with a 12hr session time-out. Results of our assessment are shown in Table 1. With 6 as an expected score, the ResNet18/50 baseline results score a 4 - these use widely available data and models, and most specifics of application are given clearly in the paper. Denoising and RotNet require more complex modelling, but reasonable detail is given in the paper, and these score 5 and 6 respectively. MVTec Anomaly Detection, Rotation Prediction and both Contrastive models score 7 or higher, due to model complexity, the latter two requiring training of a ResNet 50 over 2048 epochs, which it is estimated would take at least 512 compute hours with a P100 GPU.

The team chose reproduction of a cross-section of complexity of models, denoted by * in Table 1.

Table 1: Qualitative assessment of difficulty of reproduction of results for each of the main experiments conducted by the paper. Higher rating: more difficult; scored by 3 team-member consensus.

Model	Model Applications	Report Rating	Code Rating	Compute Rating	Overall Difficulty
ResNet 18 (random)*		1	2	1	4 *(7)*
ResNet 50 (ImageNet)*	CIFAR10*,	1	2	1	4 *(7)*
RotNet	CIFAR100*,	2	2	2	6 *(5)*
Denoising*	f-MNIST*,	2	2	1	5 *(6)*
Rotation Prediction*	Cat-vs-Dog,	**3**	**3**	**3**	**9 *(1)***
Contrastive	CelebA eyeg.	2	2	**3**	7 *(4)*
Contrastive (DA)		**3**	**3**	**3**	**9 *(1)***
Anomaly Detection	MVTec	2	**3**	**3**	8 *(3)*

*: reproduction attempted.

3 Reproduction

3.1 Reproduction Methodology

The paper used a Tensorflow centric codebase. Wishing to reproduce results independently, we used a PyTorch centric codebase. The reproduction experimental workflow was:

1. Read the paper and establish an understanding of their stated method.
2. Author models in Colab using PyTorch, and Torchvision datasets, where possible.
3. Compare our result with their result *(null hypothesis: the two results are not different)*.
4. If results significantly different: inspect their codebase, adjust our code (if necessary), repeat step 2.

3.2 RESNET18 (RANDOM) AND RESNET50 (IMAGENET PRETRAINED) REPRODUCTION

ResNet18 with random weights and ResNet50 pretrained on ImageNet generate representations, f, on which classifiers are fit. For each class, a model is fit on inlier data only, and then a test AUC reported on the whole test set. Performance is tabulated as the mean AUC over all classes. ResNet models were authored easily in PyTorch. Both our code and the paper's used SkLearn's OC-SVM.

Our linear OC-SVM results were significantly higher for all six experiments. An investigation was carried out per step 4 of our methodology, but no reason for the difference could be found. Their image processing, repesentation generation, normalisation, and OC-SVM fitting were fundamentally the same as ours. We obtained the same results over several random initialisations of the ResNet18 weights. The method of AUC computation for a OCC was identified as a potential area for discrepancy - although we both used SkLearn's implementation, other means of calculation are possible. Computing AUC manually with a single ROC vertex at (FPR, TPR) led to results that were within 10%pt of the paper's. Our RBF OC-SVM results matched on 3 out of 6 results. OCC algorithm was the focus of investigation again. Unlike for linear, One-point AUC estimates bore no correlation to any paper results. RBF scale parameter $\gamma = 10/|f|\dot{v}ar(f)$ is specified by the paper, and is 10 times wider than the SkLearn default, which drew attention and led to experimentation of the effect of mulitples 0.01, 0.1 (SkLearn default), 5, 8, 10, 100 of the paper's γ on results. The response in mean AUC on changing γ varied across the 6 models, some changing drastically, others not so much. However it was striking that for $\gamma * 8$, all 6 results came close to the paper's value. There is therefore some evidence that the paper's results were generated from a model with this hyperparameter.

3.3 DENOISING REPRODUCTION

Encoder and Decoder models were established using standard autoencoder architectures. The paper reports on the performance of denoising representations, but details of their experimental setup are omitted. Therefore, we chose to follow the process outlined in the cited paper, Vincent et al. (2008), i.e. corrupt input data by setting a fixed number of randomly selected components to 0. We set the rate of corruption to 25% of the components empirically. The corruption process was also applied recursively to the intermediate representations to attempt to produce highly robust features.

We train the denoising autoencoder for 10 epochs per dataset. We observe that the result obtained for f-MNIST with RBF OC-SVM is equivalent to the reported result. Although the performance of our implementation of the linear OC-SVM on the f-MNIST dataset is lower than that of the RBF OC-SVM as expected, it is considerably higher than the one reported in the paper. The results obtained for CIFAR10 with both versions of OC-SVM, are slightly lower than what is reported. In an attempt to obtain better results, we trained the model for more than 10 epochs but saw no significant improvement. The discrepancy that is observed could be attributed to the fact that the setup of this experiment was not described, therefore providing no guidance for setting the hyperparameters.

3.4 ROTATION PREDICTION REPRODUCTION

Rotation prediction aims to generate representations from networks trained to predict rotation of images, which is a classification task, with 4 outcomes: 0°, 90°, 180° and 270° rotation. In our implementation we initially applied augmentation rotations to f-MNIST and fed that to a ResNet50 with a 4-layer projection head. 1 epoch of training took 15min on a P100, and so we resorted to implementing a ResNet18 with the same projection head and performed hyperparameter search with different learning rates and batch sizes. However, we could not achieve convergence within the available timeframe, and our network overfits heavily yielding 92.1% and 50% training and validation accuracies. For such a small training time, the results were well below those of the paper.

3.5 CONTRASTIVE LEARNING REPRODUCTION

A constrastive objective was used in Chen et al. (2020), where the SimCLR algorithm was introduced, comprising a ResNet encoder with a projection head. Data is transformed by two augmentations, obtaining two correlated views of the input, which are then fed to the network. A contrastive loss term is used which optimises on outputs from the same source data. After training, the projection head is discarded and generated representations are evaluated linearly.

Our implementation leveraged the SimCLR repository[2], specifically to obtain pre-trained weights. Published code[3] enabled checkpoints of their `ResNet50_1x` network to be converted to PyTorch. We developed our own augmentation scheme and generated the representations.

Application to f-MNIST yielded results similar to the paper's, although CIFAR10 results were significantly lower. Intuitively, this may be due to the reproduction steps we followed and the difference in complexity of the f-MNIST and CIFAR10 datasets. SimCLR provides checkpoints for vanilla pre-trained ResNet50, and wide `ResNet50_2x` and `4x`. Due to resource constraints we selected the smaller network, which Chen et al. (2020) report achieves lower performance than supervised representation learning contrary to `4x` which achieves SoTA results. Further, the repository of the reproduced paper includes an augmentation scheme for SimCLR which is not reported in the latter work, while also choosing to train networks for 2048 epochs with a small batch size instead of 1024 epochs and large batch sizes as reported for SimCLR.

3.6 Other Observations

Row means of all tables were verified. 16/20 means in main results Table 7 (and 7/10 in related paper Table 2), were outside rounding tolerances, 6 with an error of 2%pt to 5%pt. In such a key table, this discrepancy should be brought to the authors' attention.

Table 2: Reproduction of a subset of results from paper's Table 7.

		f-MNIST			CIFAR10			CIFAR100		
$\mathbf{f}(\cdot)$	**OC-SVM**	**Paper**	**Ours**	**Ours (γ*8)**	**Paper**	**Ours**	**Ours (γ*8)**	**Paper**	**Ours**	**Ours (γ*8)**
ResNet18	Linear	50.3	70.5 (++)	-	50.1	55.2 ($\approx$)	-	50.1	60.7 (+)	-
(random)	RBF	89.8	74.5 (-)	86.4 ($\approx$)	61.2	60.5 ($\approx$)	61.3 ($\approx$)	59.6	63.3 ($\approx$)	62.0 ($\approx$)
ResNet50	Linear	77.0	91.4 (++)	-	67.9	90.7 (++)	-	71.0	94.1 (++)	-
(ImageNet)	RBF	91.8	93.8 ($\approx$)	93.6 ($\approx$)	80.0	93.1 (+)	79.2 ($\approx$)	83.7	96.2 (+)	83.2 ($\approx$)
Denoising	Linear	55.0±3.9	92.4 (++)	-	72.6±1.8	62.5 (-)	-	-	-	-
	RBF	93.9±0.4	94.7 ($\approx$)	-	83.4±1.0	67.8 (-)	-	-	-	-
Rotation Pred.	Linear	89.0±1.5	57.1 (–)	-	88.5±0.8	-	-	-	-	-
	RBF	94.6±0.3	57.1 (–)	-	90.8±0.3	-	-	-	-	-
Contrastive	Linear	92.3±1.1	84.0 (-)	-	90.7±0.8	53.6 (–)	-	-	-	-
	RBF	93.9±0.3	88.4 ($\approx$)	-	92.5±0.6	58.7 (–)	-	-	-	-

4 Conclusion

Reproducibility of Sohn et al. (2021) was assessed, and results were partially reproduced. The paper was fairly comprehensive. Discrepancies in model results were found, with various potential causes for the differences identified. We highlight that publications introducing deep models need to be accompanied by intermediate model checkpoints to allow some verification of results. Additionally, we propose that a table of hyperparameters would improve readability and could significantly constrain the search space when tuning models and therefore allow for the reproduction of more results. Discrepancies found on paper Table 2 and 7 row means should be verified by the authors.

References

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, pp. 1597–1607, 2020.

Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program), 2020.

Kihyuk Sohn, Chun-Liang Li, Jinsung Yoon, Minho Jin, and Tomas Pfister. Learning and Evaluating Representations for Deep One-Class Classification. In *ICLR*, 2021.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pp. 1096–1103, 2008.

[2]SimCLR github repository: https://github.com/google-research/simclr

[3]SimCLR-Converter github repository: https://github.com/tonylins/simclr-converter