

## A APPENDIX

We provide details on experimental procedures from the main text and a few auxiliary experiments.

### A.1 DETAILS OF VISUALIZATION EXPERIMENTS

While this is stated in the main text, we emphasize the fact that we use a pretrained SimCLR model from Google’s public repository: <https://github.com/google-research/simclr>. This characterizes one of the strengths of the DUM approach: it can leverage existing algorithms as they are. This implementation of SimCLR used slightly different data augmentations than ones we trained from scratch: it does not normalize the images and does not center crop images to 224 by 224 pixels during evaluation. Furthermore, this implementation uses a ResNet50x1 encoder. To train DUM on ImageNet representations, we optimize the DUM objective for 50 epochs using SGD with learning rate 0.01, batch size 256, momentum 0.9, and no weight decay. The input to the DUM model are the post-pooling ResNet50 features (2048 dimensions) after the final convolutional layer. The DUM encoder is a 3-Layer MLP with 4096 hidden dimensions.

### A.2 ADDITIONAL VISUALIZATIONS

We include a more expansive set of visualizations showing the least and most certain examples to embed sorted by variance norm of the encoded distribution. Fig. 3 show more classes chosen randomly from ImageNet whereas Fig. 2 shows 80 of the images with the lowest and highest variance norm for 10 datasets in the Meta-Dataset collection (Triantafillou et al., 2019). Note that these variances were extracted with a ResNet18 encoder pretrained on CIFAR10, which suggests that the features captured generalize to varied image distributions.

### A.3 DETAILS OF CORRUPTION EXPERIMENTS

CIFAR10-C, CIFAR100-C, and TinyImageNet-C datasets were downloaded from <https://github.com/hendrycks/robustness>. The standard TinyImageNet dataset, which we need for its test set, was found at <https://tiny-imagenet.herokuapp.com>. All hyperparameters for ImageNet are as in the visualization experiments, detailed above. For CIFAR10 and CIFAR100, we train a ResNet18 encoder using the SimCLR objective with output dimension 128. We use SGD with batch size 128, learning rate 0.03, momentum 0.9, weight decay 1e-4 for 200 epochs with no learning rate dropping. The data augmentations we use are a composition of random cropping to 224 by 224 pixels, random color jitter, random horizontal flipping, and random grayscale, plus normalization using dataset statistics. For ResNet18, we use the pre-pooling features after the last convolutional layer as the input to the DUM model. All following details are as above. To conduct the two-sample t-test, we use SCIPY.STATS.TTEST\_IND. To compute AUROC, we use SKLEARN.METRICS.ROC\_AUC\_SCORE.

### A.4 DETAILS ON ANOMALY DETECTION EXPERIMENTS

We first describe the preprocessing for each dataset, which we found to not be obvious from prior literature. For Arrhythmia, all entries with missing data (denoted by “?”) were replaced with 0. Everything except class 1 is considered to be an outlier. For Covertypes, PIMA, SpamBase, and Skin, the least frequent class is chosen as the outlier. For Ionosphere, if the label is “g”, it is considered in outlier. For Isolet, we use the split ISOLET1+2+3+4.DATA and treat classes “C”, “D”, and “E” as outliers. Note that we do not only use 10 instances of each class. All other classes are inliers. For KDD1999, we treat the LOGGED\_IN column as the outlier label. In addition, we ignore the following columns as they contain categorical, duplicate, or label information: NUM\_OUTBOUND\_CMDS, LABEL, IS\_HOST\_LOGIN, PROTOCOL\_TYPE, SERVICE, FLAG, LAND, IS\_GUEST\_LOGIN. For MFeat, we concatenate the following file contents into one: FAC, FOU, KAR, MOR, PIX, ZER. Classes 6 and 9 are considered inliers whereas class 0 is considered an outlier (again we do not only choose 10 points of class 0). For OptDigits, we consider classes 3 and 9 as inliers and all of class 0 as outliers. For PAMAP2, we concatenate all subject files from 1 to 10 and drop the third column as it contains too much missing data. The second column is treated as the outlier label. For Record, we concatenate data in bloc files 1 to 10. We drop columns CMP\_FNAME\_C2 and CMP\_LNAME\_C2 and replace

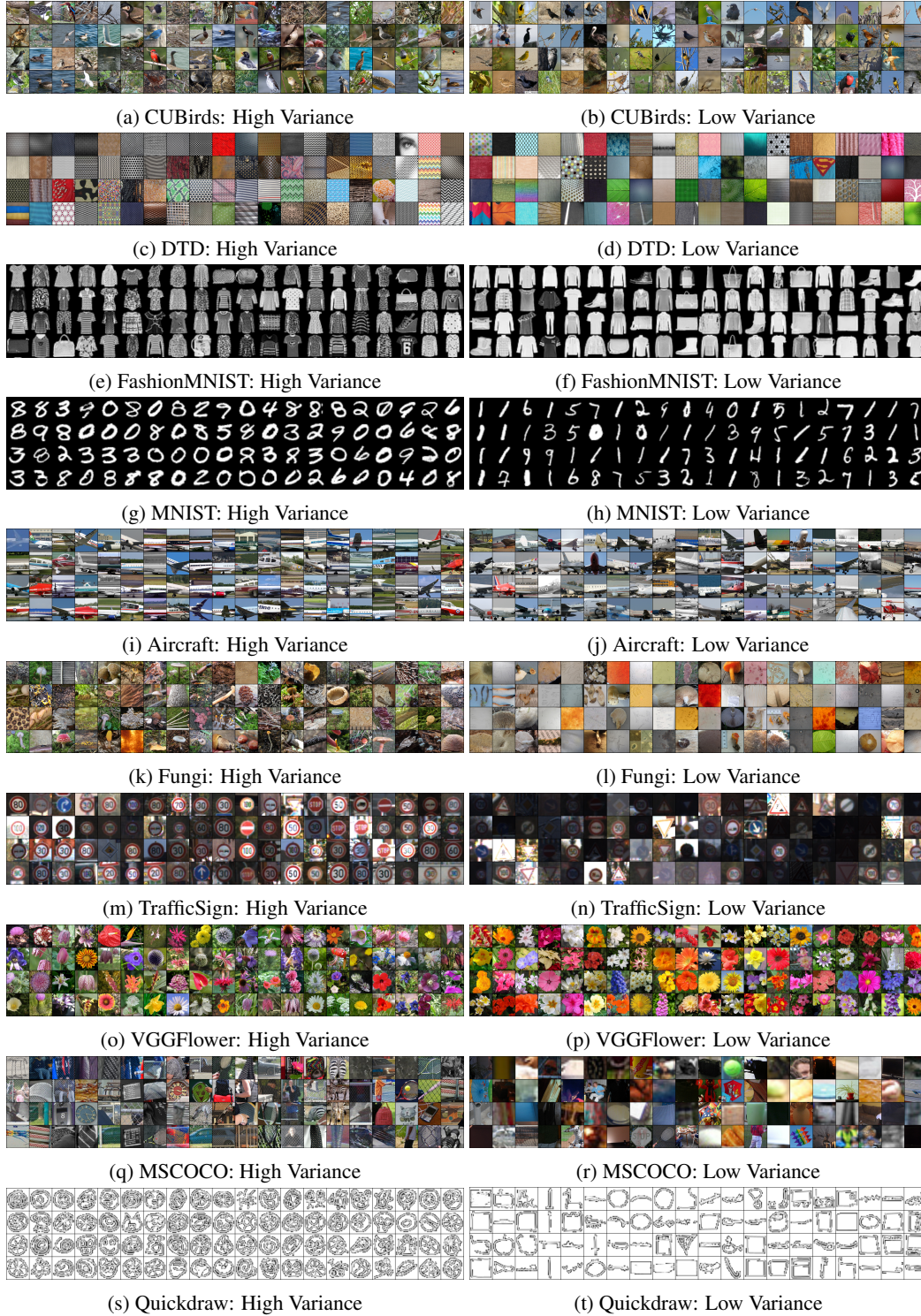


Figure 2: Top and bottom 80 Images sorted by the norm of the variance predicted by the variational encoder for datasets in the *Meta-Dataset* collection (Triantafillou et al., 2019).

all remaining missing entries, denoted as “?” with zero. We take the least frequent class (over the last column) as the outlier label. For StatLog, all rows with missing data are discarded, following which the least frequent class is chosen as the outlier. Finally, for WDBC, the second column con-



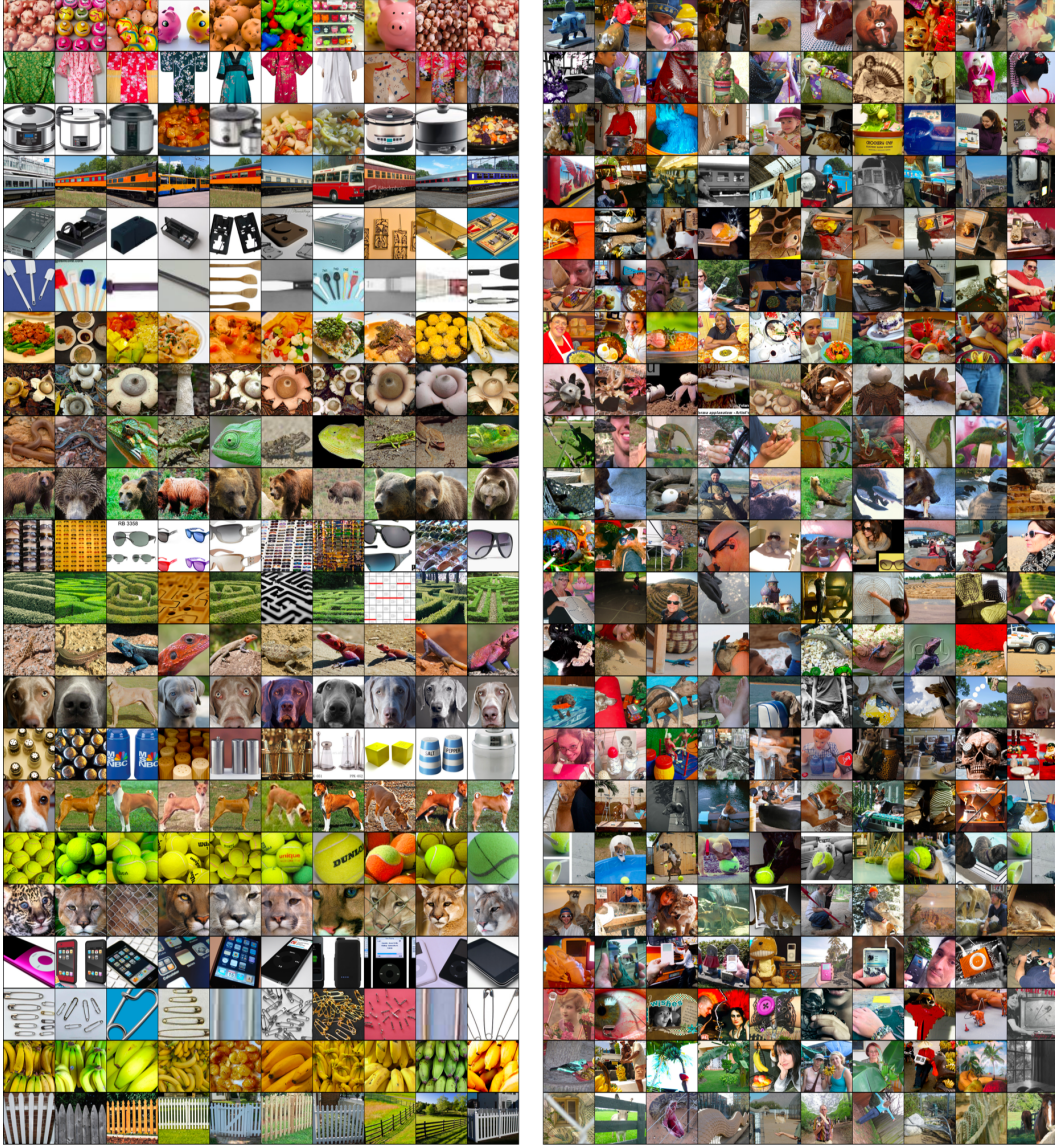


Figure 3: Expanded set of ImageNet classes showing highest and lowest DUM variance norms.

tain the outlier labels, where the label “B” is treated as an outlier and all other labels are inliers. This preprocessing procedure is largely based on the one described in [Sugiyama & Borgwardt \(2013\)](#).

We use the implementations of ISO, LOF, SVM, EE found in scikit-learn package [Pedregosa et al. \(2011\)](#) in the following packages: `SKLEARN.ENSEMBLE.ISOLATIONFOREST`, `SKLEARN.NEIGHBORS.LOF.LOCALOUTLIERFACTOR`, `SKLEARN.SVM.ONECLASSSVM`, `SKLEARN.COVARANCE.ELLIPTICENVELOPE`. We found it unfair to give the models knowledge of the contamination rate, which is unknown in real world contexts. For KNN, ABOD, and AE, we use the implementations found in the Python toolkit for detecting outlying objects, PyOD [\(Zhao et al. 2019\)](#). For the autoencoder, we use a batch size of 32 if the datasize is less than 10k entries, otherwise a batch size of 256. In the first case, we train for 100 epochs whereas we train for 20 in the latter. The architecture of the AE is an MLP with the hidden sizes 16, 8, 8, 16. We base our PyTorch implementation of RAMODO/REPEN after the public implementation found at <https://github.com/GuansongPang/deep-outlier-detection>, although with significant refactoring. We use Adam optimizer with a learning rate of 1e-3, weight decay of 1e-5, batch size 256 and 30 epochs. For our proposed method, we use an MLP with three layers,

each with 4096 hidden nodes and followed by ReLU nonlinearity. We optimize with Adam with a learning rate of  $1e-3$ , batch size 256, and a temperature of 0.07. For REPEN and our method, we train for 5 epochs only for very large datasets like PAMAP2 or KDD1999. In RAMODO, we initialize the elements in the outlier set with a subsample size of 8 and an ensemble size of 50. The KDTree uses a euclidean metric.

#### A.5 ADDITIONAL EXPERIMENTS FOR ANOMALY DETECTION

We mentioned in the main text that training DUM on top of the raw features performs about the same as DUM on top of SimCLR embeddings train on the raw features. Table 4 shows results using SimCLR embeddings for a subset of the 14 datasets below (chosen for speed).

Table 4: Lesion: comparing DUM with and without SimCLR features.

Area under the Receiver Operating Characteristic (AUROC)				
	# data	# out	DUM+SimCLR	DUM
Arrhythmia	452	207	76.0	76.6
Ionosphere	351	126	82.1	81.0
Isolet	960	240	99.9	100.0
MFeat	600	200	99.1	99.9
OptDigits	1.7K	554	95.4	99.6
PIMA	768	268	82.0	81.5
Spambase	4.6K	1.8K	82.5	83.4
Statlog	6.4K	626	84.7	89.2
Wdbc	569	212	96.2	96.9

#### A.6 DETAILS ON OUT-OF-DISTRIBUTION DETECTION EXPERIMENTS

We first train SimCLR on each of the inlier distributions using a ResNet34 encoder (to be comparable to supervised baselines, which all use ResNet34), temperature  $\tau = 0.07$ , and an embedding dimension of 128. For optimization we use SGD, momentum 0.9, learning rate 0.03, batch size 128 for 200 epochs. All images are resized to 256 by 256 prior to augmentations. After this, we fit DUM on learned embeddings, using the same MLP architecture and hyperparameters as in Sec. 4.2. Our implementation of baselines is heavily based on the following public github repositories: [https://github.com/pokaxpoka/deep\\_Mahalanobis\\_detector](https://github.com/pokaxpoka/deep_Mahalanobis_detector), <https://github.com/EvZissel/Residual-Flow>, and <https://github.com/hendrycks/ss-ood>, <https://github.com/VectorInstitute/gram-ood-detection>, which in total contain implementations for all six baselines. In addition, these baselines contain pretrained backbone networks on CIFAR10, CIFAR100, and SVHN, which we download and utilize in our replications of their results. The LSUN and TinyImageNet dataset splits were downloaded from the Mahalanobis public repository. For Rotation Prediction, we pretrain the joint supervised and contrastive objective with 0.5 weight on the rotation objective and 0.5 weight on the translation objective. We use SGD with a learning rate of 0.1, momentum 0.9, weight decay 0.0005, batch size 32 for 50 epochs with linear learning rate scheduling. For our proposed method, we optimize SimCLR with ResNet32 for 200 epochs using SGD, momentum 0.9, weight decay  $1e-4$ . The representation dimensionality is 128, and we use a temperature of 0.07. Following this, we train DUM for 100 epochs, using Adam with learning rate  $1e-3$ .