Language Guided Out-of-Distribution Detection

William Gan

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2021-139
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-139.html

May 18, 2021

Acknowledgement

First, I would like to thank Professor Trevor Darrell for advising me, and Professor Avideh Zakhor for reviewing this thesis. Next, I would like to thank Sayna Ebrahimi for being my mentor these past two years and for her guidance in this project. I am very grateful for all the work we did. Last but not least, I would like to thank Vanessa, my friends, and my family for their continued support.

Language-Guided Out-of-Distribution Detection

by William Gan

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Trevor Darrell
Research Advisor

5/13/2021

(Date)

* * * * * * *

Professor Avideh Zakhor
Second Reader

5/8/2021

(Date)

Language-Guided Out-of-Distribution Detection

by

William Gan

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Trevor Darrell, Chair
Professor Avideh Zakhor

Spring 2021

Language-Guided Out-of-Distribution Detection

Abstract

Language-Guided Out-of-Distribution Detection

by

William Gan

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Trevor Darrell, Chair

In machine learning, most models are trained under the assumption that their test data will come from the same distribution as their training data. However, in the real world, this may not be true, necessitating a method to detect out-of-distribution (OOD) inputs. Thus far, prior works mostly evaluate when the OOD inputs are different classes, e.g. an image of a dog passed to a cat breed classifier. They do not consider OOD inputs that are of the same class but with a stylistic change, e.g. a cat under red lighting. In this work, we distinguish these two types as semantic and stylistic OOD data, respectively. We also propose to use a new modality, natural language, for the problem. As both the in-distribution dataset and stylistic OOD differences can be described with natural language, a model that utilizes it can be beneficial. We use OpenAI CLIP to encode style-contextual descriptions of our training dataset and at test time compare these to the encoded image. Our method, which we call DesCLIPtions, requires no additional training yet outperforms baselines for certain tasks. Overall, we conclude that natural language supervision is a promising direction for OOD detection.

Contents

List of Figures

List of Tables

Acknowledgments

First, I would like to thank Professor Trevor Darrell for advising me, and Professor Avideh Zakhor for reviewing this thesis. Next, I would like to thank Sayna Ebrahimi for being my mentor these past two years and for her guidance in this project. I am very grateful for all the work we did. Last but not least, I would like to thank Vanessa, my friends, and my family for their continued support.

Chapter 1

Introduction

In machine learning research, the training and testing data almost always come from the same *distribution*. In fact, they are usually constructed by randomly dividing a single dataset into two. However, in the real world, the data seen by a model at inference time may be different than that at training time. For example, an autonomous vehicle may suddenly encounter road conditions it has never seen before. As another example, a model trained to classify viral diseases may encounter a new strain. In these cases, instead of relying on the model, we want to defer to, say, a human expert.

To safely deploy real world machine learning models, there must be a mechanism to detect these different inputs [2]. This is what out-of-distribution (OOD) detection, also known as outlier, novelty, or anomaly detection, seeks to do. A good amount of research has been done on this topic in computer vision, but thus far, prior works mostly evaluate when the OOD inputs are different classes, e.g. an image of a dog passed to a cat breed classifier. They do not consider OOD inputs that are of the same class but with a stylistic change, e.g. a cat under red lighting. In this work, we categorize OOD into different types and call the aforementioned two semantic and stylistic OOD data, respectively. We also propose to use a new modality, natural language, for the problem. As both the in-distribution dataset and stylistic OOD differences can be described with natural language, a model that utilizes it can be beneficial. We use OpenAI CLIP to encode style-contextual descriptions of our training dataset and at test time compare these to the encoded image. Our method, which we call DesCLIPtions, requires no additional training yet outperforms baselines for certain tasks. To the best of our knowledge, our method is novel for OOD detection.

The rest of this thesis is organized as follows. In Chapter 2, we give an overview of OOD detection and prior methods. In Chapter 3, we describe the method, going over motivations. In Chapter 4, we detail our experiments and discuss our results. Lastly, in Chapter 5, we conclude our study and discuss future work. Overall we would describe our contributions as developing distinctions between and benchmarks for different types of OOD data, demonstrating the potential of natural language supervision for the problem, and evaluating its limits.

Chapter 2

Background

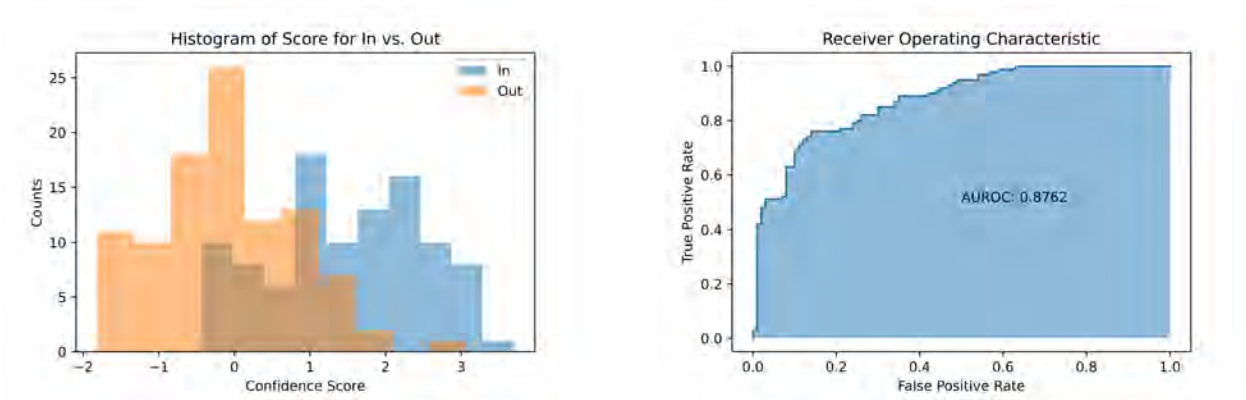
In this chapter, we go over the OOD detection problem statement, clarify different types of OOD detection, draw distinctions between different types of OOD data, and go over prior work.

2.1 Problem Statement

Out-of-distribution detection, also known as outlier, novelty, or anomaly detection, seeks to determine if an input belongs to the same distribution as the training dataset, called the In dataset. To do so, most OOD detection methods devise a *confidence score* corresponding to how *in-distribution* an input is. The higher the confidence score is, the more likely that the input is from the training distribution, and a threshold can be set to classify inputs as in-distribution or not. Sometimes, an *anomaly score*, which can be treated as simply the inverse, is devised instead. To evaluate methods, one can use detection accuracy assuming the optimal threshold, but as this depends on the amount of in-distribution and OOD data, typically the area under the receiver operating characteristic (AUROC) is used instead. The AUROC can be interpreted as the probability that the confidence score for a random in-distribution sample is higher than that for a random OOD sample. This is mathematically appealing since it means that the AUROC is invariant to simple transformations on the confidence score. Figure 2.1 shows an example histogram for a confidence score on In and Out datasets and the corresponding ROC curve.

2.2 Types of OOD Detection

With OOD detection, there are three variants: unsupervised, semi-supervised, and supervised. In unsupervised OOD detection, a method may use the training data of the In dataset to formulate its confidence score, but not data from Out datasets. In semi-supervised OOD detection, the use of *auxiliary* Out datasets at training time is allowed, though not the Out dataset that is being evaluated. For example, Hendrycks et al. [25] uses CIFAR-10 (In)



(a) Our confidence score is generally higher for In data, which is good.

(b) The corresponding ROC curve and AUROC.

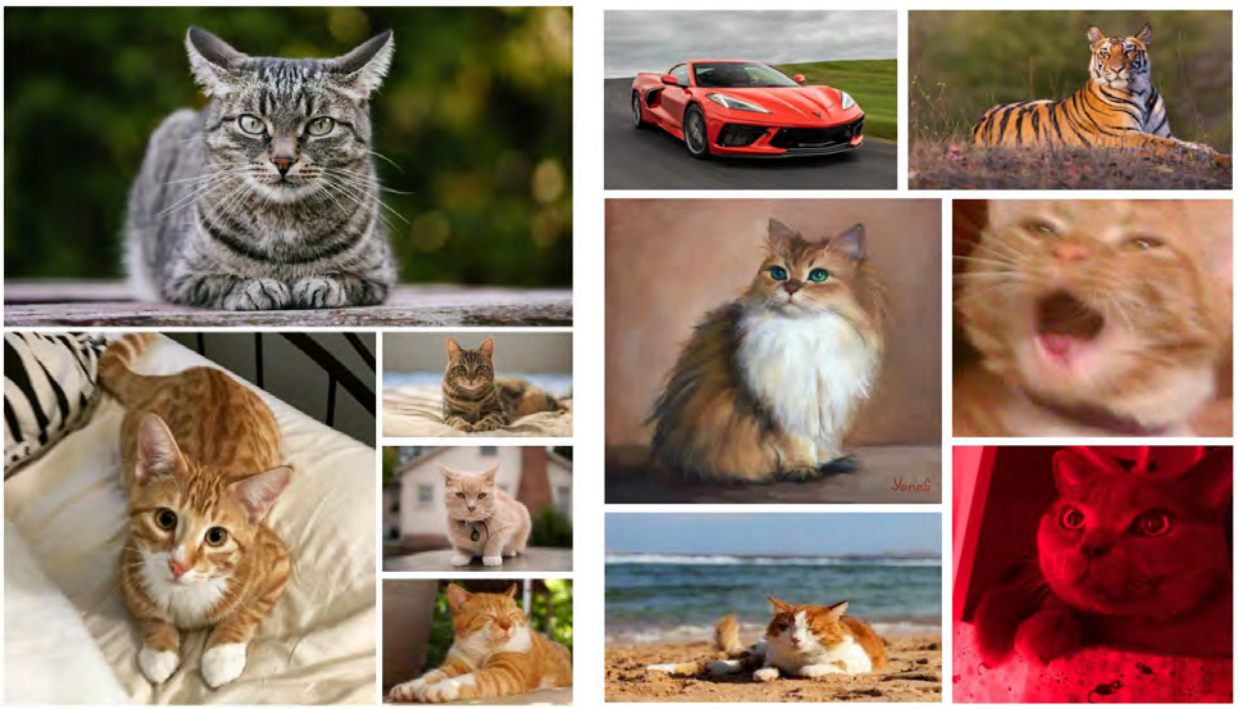
Figure 2.1: Histogram of confidence scores and ROC curve for an example OOD detection method.

and TinyImages (auxiliary Out) at training time to evaluate on CIFAR-10 (In) and SVHN (Out). In supervised OOD detection, access to samples from the evaluated Out dataset at training time is allowed. In this setting, the OOD detection problem reduces to a binary classification problem, and thus is generally less considered. Between semi-supervised and unsupervised OOD detection, semi-supervised generally performs better, but comes with the question of how to choose the auxiliary Out dataset, if such data is even available.

2.3 Types of OOD Data

Generally, OOD data are considered to be inputs that differ from the training dataset and have a high likelihood of being misclassified. However, this can come in different forms, which we illustrate in Figure 2.2. For a classifier trained to determine cat breed, the OOD input could simply be a different object such as a car or a tiger. It could also be a cat but in a different background, such as if the cat is on a beach but all our training images are cats in homes, backyards, etc. It could be a cat but under a different style, such as a painting or a photo under red lighting. It could also be a cat but with the photo corrupted, such as a blurry photo. For these last 3 types, we note that its important to detect as the classifier's accuracy can be reduced by 50% or more on them.

We note that this list of types of OOD inputs is not exhaustive. Alcorn et al. [1] use 3D renderers to alter the pose of objects in natural images and find that, for example, while a classifier can correctly identify a schoolbus in its canonical pose, it will to fail to do so 97% of the time if the schoolbus is in a different pose, e.g. rotated, flipped over, etc. While admittedly some of their renders are unrealistic, this shows that pose is also an OOD factor. We are sure there are many more as well.



(a) In dataset (b) OOD data

Figure 2.2: Example OOD data for a classifier trained to distinguish cat breeds.

Each type of OOD input comes with its own challenges. Tigers are clearly different from cats but at the pixel level, they may be hard to distinguish. They have similar colors and their pictures might have similar backgrounds. Certainly, the pixel-level variations between different images of cats are just as large as that between images of cats and tigers. On the other hand, a picture of cat under red lighting is clearly different from a color standpoint, though depending on how the neural network operates, it may not be able to distinguish it anyway, e.g. if it only looks at edges insteads of colors. Hendrycks et al. [27] point out that there are many faces of robustness such as camera quality, geography, etc. However, from our understanding, most prior works in OOD detection do not make the distinction between different types of OOD data. To develop this, we introduce new terms describing different types of OOD data. We call OOD data of a different object or class *semantic* OOD data, OOD data of a different background *contextual* OOD data, and OOD data of a different style (e.g. painting) or texture (e.g. blurry photo) *stylistic* OOD data. In this work, we focus on semantic and stylistic OOD data [1].

We do want to distinguish OOD detection from robust OOD detection [7]. Similar to how classifiers can be fooled by images that appear identical to training samples but that have been adversarially perturbed, Chen et al. [7] showed that good OOD detectors can be

[1] We skip over contextual due to lack of datasets.

fooled as well. While this is an important problem, we believe it to be its own unique issue, and thus in this work, we only focus on images that are naturally occuring.

2.4 Prior Work

In this section, we go over prior work. OOD detection is a long running topic of research that extends beyond images. A survey from 2009 by Chandola et al. [6] reveals various techniques in different domains such as Cyber-Intrusion Detection, Fraud Detection, Medical Anomaly Detection, and Industrial Damage Detection. From our understanding, however, the topic became less active until research done by Hendrycks and Gimpel [23] reintroduced the topic in a deep learning context (particularly deep image classification). In this section, we do our best to cover prior work before and after Hendrycks and Gimpel [23], but as our work is on OOD detection for deep image classification, we only focus on works after Hendrycks and Gimpel [23] in later sections.

Methods Before Deep Learning Era

For methods before Hendrycks and Gimpel [23], we summarize from the previously mentioned survey [6]. These methods are mostly unsupervised and can be categorized as classifier-based, nearest-neighbor-based, clustering-based, or statistical. Classifier-based methods such as that proposed by Stefano et al. [10] train a classifier, but instead of applying a softmax function to the K output logits for each class, apply K sigmoid activations to each logit. This can be interpreted as having K one-class classifiers (with a shared backbone), where each classifier is trained with the corresponding class as the In dataset and the rest of the classes as the Out dataset. When a new input is tested, the method looks at each of the K outputs and thresholds based on them. Stefano et al. [10] apply this to multi-layer perceptrons (MLPs), learning vector quantization (LVQ), and probabilistic neural networks (PNNs) as the classifier. Another classifier-based method proposed by Hawkins et al. [20] involves Replicator Neural Networks, which reconstructs the input as output. They use reconstruction error as the anomaly score as it is expected to be higher for OOD inputs. Overall, an advantage of classifier-based methods is usually that they are fast at testing time.

Nearest neighbor methods, like their name suggests, form their confidence score by looking at the distance of a test input to the training data of the In dataset. For example, Byers and Raftery [5] use the k-th nearest neighbor distance as the anomaly score and find success in detecting land mines from satellite ground images. As an alternative, Eskin et al. [15] use the sum of the distances from the k nearest neighbors. As another alternative, Knorr and Ng [34] use the number of neighbors that are not more than d distance from the test input as the confidence score. This last alternative in particular can be viewed as estimating the density of the test input, since it calculates the number of neighboring points in a radius-d

hypersphere, and it leads us to other similar techniques based around relative density. Breunig et al. [4] devise an anomaly score called Local Outlier Factor (LOF) using the inverse of the k-th nearest neighbor distance. As the k-th nearest neighbor distance tells us the radius of a hypersphere that contains k points, its inverse represents local density. LOF calculates its anomaly score by looking at average of the local density of an input's k nearest neighbors compared to its own local density. The advantage of nearest neighbor methods is that they require little training but are less efficient at test time, as finding the nearest neighbors is expensive. Structures like k-d trees can be used, but such structures do not scale well with dimension.

Clustering-based methods are similar to nearest neighbor methods. The most common technique is to cluster the training In dataset into centroids, and use the distance to the nearest centroid as the anomaly score at test time. Smith et al. [49] try this with Self-Organizing Maps (SOM), K-means clustering, and Expectation Maximization (EM). Clustering-based methods are fast at test time, but highly depend on the effectiveness of the clustering algorithm.

Finally, among methods before the deep learning area, there are statistical methods, which generally fit a probability model to the In distribution. In statistics, there is a long history of outlier detection involving hypothesis tests, and statistical OOD methods can be seen as based on them. A common method is to fit a normal distribution to the data using maximum likelihood estimation (MLE) and then reject if the test input is a number of standard deviations away. For multivariate data, this becomes equivalent to thresholding on the Mahalanobis distance $\sqrt{(x-\mu)^\top \Sigma^{-1}(x-\mu)}$, where μ and Σ are the mean and covariance matrix. Other statistical tests have also been researched, with Surace and Warden [51] using student's t-test in detecting damage to structural beams and Ye and Chen [58] using a chi-square test statistic in detecting OOD inputs in operating system call data. Additionally, more complicated statistical models have been used, such as Gaussian mixture models. Spence et al. [50] use them to detect anomalies in mammographic image analyses.

Methods Since Deep Learning Era

Despite the abundance of OOD detection research done before the deep learning era, most techniques cannot be directly applied to image classification. This is because images are high-dimensional (even a 32x32 RGB image has $32 \times 32 \times 3 = 3072$ values) and because they do not follow Euclidean distance well. An image that is close in Euclidean distance to another image will look similar but the converse is not true: two images of dogs that humans would say are similar may actually be far apart in pixel space due to where the dogs are positioned, the dogs's pose, background objects, etc. This means that nearest neighbor, clustering, and statistical methods will not work well in pixel space, and the use of deep convolutional neural networks (CNNs) are needed. CNNs generally project an image down into a lower dimensional space where the desired task is performed. For example, in a ResNet-18 [21]

model, 224x224 RGB images are converted to a 512 dimensional vector [2] and then are passed through a fully-connected layer to make a classification. Early deep OOD detection methods tried to use the CNNs directly. For example, Hendrycks and Gimpel [23] use the maximum softmax probability (MSP) as the confidence score. CNN classifiers output logits that are transformed to a probability distribution under the softmax function. This is trained with cross entropy loss so that all the probability mass is put on the correct class. At testing time, the class with the maximum probability is used as the prediction, and we expect its value to represent the confidence, leading to MSP. Liang et al. [37] later show that scaling the logits before applying the softmax function improves AUROC, as well as perturbing the input. For the input perturbation, their method, called ODIN [3], performs a backpropogation on the test input and uses the sign of the gradient at the input to shift it. The confidence is obtained after passing in the modified input. This input preprocessing is the negative of the fast gradient sign method introduced by Goodfellow et al. [16] to generate adversarial examples. Adversarial examples are small perturbations designed to decrease the probability of the true class, perhaps leading to a different classification. By adding the perturbation in the opposite direction, Liang et al. [37] increase the probability of the true class, and importantly it happens to have a larger increase for in-distribution inputs. This leads to better AUROC, but from our understanding, their best results requires hyperparameter tuning on OOD data. In another line of work, Lee et al. [35] use the outputs of the CNN at different layers and calculate the class-conditional mean $\mu_c^{(\ell)}$ and total variance $\Sigma^{(\ell)}$ for the training In dataset. At test time, they used the predicted class and these means / variances to calculate the Mahalanobis distance $\sqrt{(x-\mu_c^{(\ell)})^\top \Sigma^{(\ell)^{-1}} (x-\mu_c^{(\ell)})}$ at each layer, and these are linearly weighted to form the anomaly score. Their method can be thought of as an extension of statistical methods before the deep learning era to CNNs. However, their method requires Out dataset samples to calculate the linear weights.

MSP and Mahalanobis distance on CNN outputs are rooted in methods before the deep learning era. While one can similarly apply other methods before to CNNs, research has halted in this direction. This is because it turns out CNNs are quite overconfident in their predictions. Guo et al. [19] devise a metric called the Expected Calibration Error (ECE) and find that it is high for modern deep neural networks. The ECE works by looking at the MSP for the test data of the In dataset. MSP values are binned; for example, those between 0.5-0.6 will be put into a bin, those between 0.6-0.7 will be put into a bin, etc. Within each bin, the accuracy is calculated. For well-calibrated classifiers, the accuracy in the 0.5-0.6 bin should be roughly 50%-60%, but Guo et al. [19] find that this is not the case. In particular, many values have high MSP (ex. 0.9-1) but the accuracies in these bins do not match. Ovadia et al. [41] perform a similar study, looking at how accurate the MSP is under dataset shift, i.e. under OOD inputs. They find that even a classifier that has been calibrated to give accurate confidence estimates on the test dataset will still give

[2] These are generally called the features.

[3] Not to be confused with the ODIN mentioned earlier as a nearest neighbor method.

confident predictions for OOD inputs. Jiang et al. [29] modify the MSP confidence score by also looking at distance in the CNN output space. In particular, they look at the distance of the test input from an α-high-density-set of training samples of the predicted class and from an α-high-density-set of training samples from the nearest other class. If the distance to the predicted class is greater than the nearest class, this is an indication that the classifier is making a mistake, leading to low confidence. However, they find that while their score works well in low dimensions, it does offer much improvement over MSP on images. These works together suggest that CNNs may not naturally be able to distinguish OOD inputs, and applying classical methods on their outputs is futile. This also appears to be a general issue with deep neural networks, as Nalisnick et al. [40] show that likelihood models such as VAEs also assign higher probabilities to OOD inputs than their training dataset. For example, they show that a CIFAR-10 Variational Autoencoder (VAE) [33] assigns higher likelihood probabilities to SVHN. This may be telling in the case of Pidhorskyi et al. [44], whose work revolves around the principle that the In dataset lies on a manifold and that distance from the manifold can be used as the anomaly score. To learn the manifold, they use a VAE but with an adversarial loss on the $\mathcal{N}(0,1)$ prior in the latent space as opposed to the usual KL divergence, as well as an adversarial loss on the reconstruction in addition to the usual mean square reconstruction error. From our understanding, their method shows improvements on CIFAR-100 but not CIFAR-10, and we believe this is due to the issue Nalisnick et al. describes. Much of the OOD research work following these findings about deep neural networks have been on how to train better calibrated CNNs or likelihood models, such as by modifying the training loss function.

For the remainder of this section, we will be covering these works. Mandelbaum and Weinshall [38] once again look at the k nearest neighbors in the CNN feature space to form their confidence score. For each neighbor, the Euclidean distance d is raised to e^{-d}, and the confidence score is the sum of the e^{-d} that have the predicted class label over the total. However, unlike methods before, they add terms to the loss during training so that features of the same class are minimized in distance while features of different classes are at least m distance apart. They also experiment with a different loss based around the fast gradient sign method [16].

Devries and Taylor [13] add a separate branch to the penultimate layer of the CNN classifier, representing confidence. To train the branch [4], they use cross entropy loss between $c \cdot p + (1-c) \cdot y$ and y. Here, c is the output confidence, p is the output probability distribution, and y is a one-hot vector representing the ground truth label. Under this training scheme, if the classifier is unconfident, it may use the ground truth. However, it generally becomes more confident (increasing c) on the training dataset as it is also trained to minimize $-\log c$ (i.e. maximize confidence). After training, Devries and Taylor [13] use c as the confidence score and find improvements over MSP.

[4] Normal cross entropy loss is used to train the main branch.

Lee et al. [36] add a KL divergence term $KL(\mathcal{U}(\mathcal{Y})\|p)$ to the training loss for OOD inputs. For these inputs, The KL divergence encourages their output probability distribution to be uniform and reduces their MSP at test time. However, this requires OOD data at training time. The authors try using noise and as semi-supervised variant, auxiliary Out datasets. They also try jointly training a Generative Adversarial Network (GAN) [17] to generate OOD data. While the semi-supervised variant shows success, the GAN variant has smaller gains. Hendrycks et al. [25] further evaluate the semi-supervised variant with more datasets. Referring to the technique as Outlier Exposure, they find it offers large improvements over most unsupervised methods.

Vyas et al. [54] extend upon ODIN [37], attempting to resolve the inherent issues of overconfidence with CNNs by training an ensemble of leave-out classifiers. In particular, they partition the training In dataset into the K parts and output K classifiers, each classifier being trained under standard cross entropy loss for $K-1$ partitions and being trained under a separate loss for the remaining partition. It should be noted that the classes of the each partition must be mutually exclusive. For example, on CIFAR-100, they let $K = 5$ and partition the 100 classes into parts with 20 classes each. In effect, the $K - 1$ partitions are the In dataset and the remaining partition is the Out dataset. The separate loss is a margin loss that ensures the average entropy of OOD inputs is at least m higher than that of in-distribution inputs. At test time, their method applies ODIN on the ensemble. It shows good improvements over ODIN, but requires a lot more training.

For likelihood models, Ren et al. [47] suggest that the reason why they assign higher probabilities to OOD inputs is because the probabilities are dominated by background statistics. For example, a Fashion-MNIST [5] likelihood model will learn to simply assign black pixels high likelihood, since Fashion-MNIST images are mostly black. This becomes an issue when the model is fed MNIST images, as MNIST images contain even more black. To solve this issue, the authors divide likelihood into background and semantic components, and argue that for OOD detection we only care about the semantic component. They train a normal likelihood model and a background likelihood model, with the background likelihood model being fed perturbed inputs. These perturbed inputs mess with the semantic component but not the background component, so when looking at the test inputs, one can divide the normal likelihood by the background likelihood to obtain the semantic confidence score. Their method shows success on Fashion-MNIST [57], MNIST and CIFAR-10.

Hendrycks et al. [28] show that training the CNN classifier with auxiliary self-supervised objectives makes the model better for OOD detection and generally more robust. In particular, they randomly rotate images by 0, 90, 180, or 270 degrees and add rotation prediction (done through a separate branch) as a task. At test time, their anomaly score uses the KL divergence between the output probability distribution and a uniform distribution (which they argue performs similarly to MSP) and the average cross entropy loss for the 4 rota-

[5] Similar to MNIST but black and white images of clothing.

tions. In particular, it is $KL(\mathcal{U}(\mathcal{Y})\|p) + \frac{1}{4}\sum_{r\in\{0^\circ,90^\circ,180^\circ,270^\circ\}} \mathcal{L}_{CE}(r, p_{rot})$. They find that their method performs better than even Outlier Exposure in certain cases.

Thulasidasan et al. [53] and Yun et al. [59] approach better OOD detection through data augmentation, which is known to improve classification accuracy. Thulasidasan et al. [53] evaluate mixup training, which convexly combines different images (and their labels) in the training In dataset to create new images (and labels). They find that for mixup-trained models, the MSP has higher AUROC. Yun et al. [59] extend this with their own data augmentation strategy, CutMix. A disadvantage of mixup training is that it augments an entire image as opposed to just regions. Cutout [12], which randomly replaces patches with black pixels, does this, but black pixels are not ideal. CutMix patches the regions of other images instead, and like mixup training reweights the label vectors. Yun et al. [59] show further improvements for a CutMix-trained classifier.

Shalev et al. [48] propose a novel training method utilizing word embeddings. In particular, they utilize pretrained word embedding models such as the Google News Skip-Gram model [39], GloVe [43], and Fast-Text [30]. For each of these word embedding models, they train their own embedding model on the training images. These models seek to return a vector that is close to the embedding of their class's name. If the vectors $\boldsymbol{e}^k(y)$ denote the embedding from the k-th pretrained word embedding model on the name of class y, then the loss for a sample x with label y is $\overline{\ell}(\boldsymbol{x}, y; \theta) = \sum_{k=1}^{K} d_{\cos}(\boldsymbol{e}^k(y), \boldsymbol{f}^k_{\theta^k}(\boldsymbol{x}))$. Here $\boldsymbol{f}^k_{\theta^k}$ is the image embedding model corresponding to the k-th pretrained word embedding model, and $d_{\cos}$ is cosine distance, which is $d_{\cos}(\boldsymbol{u}, \boldsymbol{v}) = \frac{1}{2}\left(1 - \frac{\boldsymbol{u}\cdot\boldsymbol{v}}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|}\right)$. For OOD detection, they use the sum of the Euclidean norms of the image embeddings as the confidence score, i.e. $\sum_{k=1}^{K} \|\boldsymbol{f}^k_{\theta^k}(\boldsymbol{x})\|_2^2$. Their method shows improvements over ODIN [37], although not to the same degree as some of the previous methods mentioned.

Hendrycks et al. [24] show that using pretrained models improves OOD detection. For a given In dataset, they finetune a model pretrained on Downsampled-ImageNet, and afterwards, simply use the MSP as the confidence score. This leads to decent AUROC gains when CIFAR-10 and Tiny ImageNet are the In dataset and large gains when CIFAR-100 is the In dataset. This method is somewhat similar to the previously mentioned Outlier Exposure [25] [6]. In particular, it shows that exposing the model to large amounts of data is important for OOD-detection, which we believe as well.

The last set of prior works we cover are related to contrastive learning, a recently popular, alternative, and more general way to train neural networks. In contrastive learning, specifically in the SimCLR [8] objective, a model is trained to produce features that have high cosine similarity with the features for transformations of the image. Given a set of transformations $\mathcal{T}$ (such as brightness adjustment, cropping, and always the identity transformation), a minibatch of N images is augmented to $2N$ images by randomly applying a transformation to each image. The images are then passed to the model to obtain a repre-

[6] And is done by the same authors.

sentation, and then to an additional model to project it to a lower dimension [7]. Letting $\boldsymbol{z}_i$ be the projections for each of the $2N$ images (with $i = 2k - 1$ and $i = 2k$ corresponding to the same original image) the loss between a pair of images is defined as

$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)} \tag{2.1}$$

with sim indicating cosine similarity and τ being a temperature scaling parameter. The overall loss for the minibatch is then defined as

$$\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)] \tag{2.2}$$

The goal of contrastive learning is to learn good features for images that can later be used for other tasks. For example, if a model learns class-clustered features, the features can be passed to a single fully-connected layer (sometimes known as a linear probe [46]) for classification. It turns out that this does happen, and for OOD detection, Winkens et al. [56] show that the features are also good. The authors fit a multivariate normal distribution to the features of each class, using the empirical mean and covariance matrix. At test time, they use the maximum probability over the distributions as the confidence score. The method outperforms rotation prediction [28] and even Outlier Exposure. Tack et al. [52] take this further by adding shifting transformations, which are a set of transformations $\mathcal{S}$ (e.g. rotation) that are considered to make an image OOD. Their basic method augments a minibatch with the shifting transformations before applying SimCLR, essentially making the transformations different samples. They find while such training does not improve features for image classification, they help for OOD detection. They also add rotation prediction as an auxiliary task. At test time, they compute their basic confidence score as

$$s(x) = \max_m \ \mathrm{sim}(z(x_m), z(x)) \cdot \|z(x)\| \tag{2.3}$$

Here the maximum over m is the maximum over the training In dataset, and $z(\cdot)$ represents the feature [8]. The multiplication by the norm of the feature $\|z(x)\|$ seems to follow the results of Shalev et al. [48], i.e. that in-distribution features have high norm. Tack et al. further improve their confidence score by taking an expectation over shifting transformations, using the rotation branch, and ensembling over non-shifting transformations, but we refer to their paper for those details.

[7] This additional model is only used during training.

[8] We technically also used z to denote the projection during training earlier.

Chapter 3

Language-Guided Out-of-Distribution Detection

In this chapter, we discuss the motivation behind natural language supervision, provide background on CLIP, and describe our method.

3.1 Motivation

As mentioned in the Introduction, our work incorporates natural language as an additional modality for OOD detection. We utilize natural language supervision, which refers to training a model with image-caption pairs. That is, instead of using a number (0, 1, 2, ...) corresponding to a class as the label for an image, a piece of text describing it is used instead. Our motivation for this is two-fold. For one, captions are more informative. Two images might be similar but in the real world they will contain many differences. For example, in ImageNet, one image of a swimsuit is actually a group people of posing for a photo at a backyard party. The swimsuit label comes from the fact that the people in the image are wearing swimsuits. On the other hand, another image is simply a swimsuit laid out against a white background. In standard image classification, these two images would be given the same label, but the issue with this is that it ignores the nuances. This is okay and may even be desirable only for classification, but with OOD detection in mind, we believe this is detrimental. Using captions as labels can lead to better representations.

Our second motivation is that using captions as labels enables supervised learning on an infinite variety of data. With classes, every concept needs its own label. Adding additional concepts requires additional labels, which usually requires adding another entry in the final weight matrix of the model. However, because natural language labels are continuous, they do not suffer from this issue. For OOD detection, we believe we want to learn from an infinite variety of data, i.e. semi-supervised over unsupervised. Prior works [25, 24] show that using auxiliary Out datasets leads to substantial performance gains. Furthermore, it does not appear that the way the auxiliary Out datasets are used is important, only that the model

be exposed to it. We believe this makes sense, as when the goal of a neural network is only to classify in-distribution images correctly, it cannot fully learn the abstractions (shape, pose, texture, color, identifying the subject) and knowledge (tigers may look like cats but they are larger, have a unique striped pattern, etc.) that are required to distinguish OOD images. Humans have these abstractions and knowledge because we have a lifetime of training data.

3.2 CLIP

To incorporate the motivations above, we need a model trained on enormous amounts of image-caption data. Such a model is prohibitively hard to train for most researchers and organizations, but fortunately, researchers at OpenAI have recently done so. Their model, Contrastive Language-Image Pretraining (CLIP for short) [46], consists of an image encoder and text encoder trained on 400 million image-caption pairs. The encoders are trained to embed images and text in the same feature space, with an image and text encoding being similar if they are similar in concept. For example, a photo of a dog lying on the grass and the sentence "A photo of a dog lying on the grass" will be similar. The authors learn such a model under a contrastive learning framework, which we note is promising for OOD detection as mentioned in the Prior Works section. At each iteration during training, an image minibatch of size N and the corresponding caption minibatch are fed to their respective encoders to obtain features. These features are optimized so that the N pairs of corresponding image-captions are maximized in cosine similarity while the other $N^2 - N$ pairs are minimized in cosine similarity [1]. With a large minibatch size (they use 32768), the authors learn good representations. Figure 3.1, which is taken from their paper, illustrates this process.

3.3 Methods

Using CLIP, we devise two zeroshot OOD detection methods, which we collectively call DesCLIPtions. One method is meant specifically for semantic OOD data, while the other method is meant for other types of OOD data, e.g. stylistic. Our first method mostly follows the zeroshot classifier described in the CLIP paper [46] and is as follows. For a given training dataset, we encode and then normalize the dataset classes using CLIP. For example, if the classes are airplanes, cats, etc., we encode then normalize "a photo of an airplane", "a photo of a cat", etc. Normalization is done as the features are also normalized in CLIP's training process, and we find it leads to slightly better results. In some cases, it is better to have multiple descriptions for a given category. For example, for airplanes, we might want to have multiple sentences such as "a photo of an airplane on a runway", "a photo of an airplane in the sky", "a cropped photo of an airplane in the sky", etc. In these cases, we normalize each sentence, average them [2], and renormalize again.

[1] Technically this is done after projecting the features into a different space, as done in SimCLR [8].

[2] We also experimented with different ways to use them, but found that averaging works well.

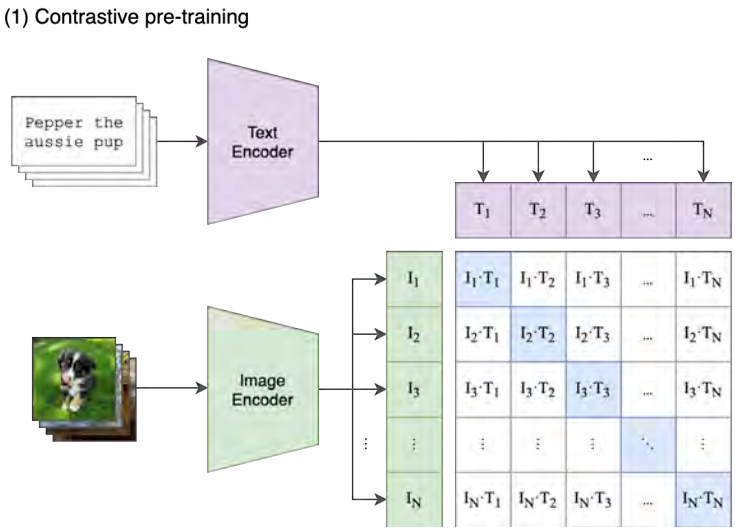


Figure 3.1: CLIP training process, taken from their paper [46].

At test time, we encode an image input and normalize it. We then take a dot product with each of our descriptions and simply use the max as the confidence score. The dot product here represents cosine similarity, and so if an image belongs to our In dataset, it will have high cosine similarity with one the descriptions. With our cosine similarities, we could apply the softmax function to obtain probabilities, however, we find that this slightly worsens results. With the softmax function, if we have K descriptions, our maximum probability is guaranteed to be at least $1/K$, but with maximum cosine similarity, we may get a number close to -1 if the image is far from every description. The softmax probability also does not work in a one-class scenario, where the In dataset only has one class. Figure 3.2 contains PyTorch code describing our algorithm. It provides a function that returns confidence scores for a minibatch of images.

For other types of OOD data, the above method will not a work. CLIP is relatively robust and will return a high cosine similarity between, for example, a painting of a cat and "a photo of a cat". We suspect this is also due to the fact that a photo is general word; it is not unusual to say "a photo of a painting of a cat". However, CLIP still has knowledge that a painting is a painting. A painting of a cat will have even higher cosine similarity with "a painting of a cat". If we have a general idea of kinds of OOD data that we want to guard against, we can use this to our advantage. For our second method, we do this for stylistic OOD data by also encoding other styles of classes, which we call out descriptions. For example, if our classes are again airplanes, cats, etc., we encode "a painting of an airplane", "a painting of a cat", etc. in addition to a "a photo of an airplane", "a photo of a cat", etc. At test time, we again look at the cosine similarity with all these descriptions. However, this time, we scale the similarities by a temperature parameter and then apply the softmax function, returning a probability distribution. We then use the max probability among "a photo of an airplane", "a photo of a cat", etc. The idea here is that if we, for example,

```python
import torch
import torch.nn.functional as F

import clip

def descliptions_semantic(clip_model, class_descriptions):
    weights = []
    for descriptions in class_descriptions:
        texts = clip.tokenize(descriptions).cuda()
        text_encodings = clip_model.encode_text(texts)
        text_encodings = F.normalize(text_encodings, dim=-1)
        average = torch.mean(text_encodings, dim=0)
        average = F.normalize(average, dim=-1)
        weights.append(average)
    weights = torch.stack(weights, dim=1)

    def confidence_score(image_minibatch):
        image_encodings = clip_model.encode_image(image_minibatch)
        image_encodings = F.normalize(image_encodings, dim=-1)
        cosine_similarity = image_encodings @ weights
        scores, _ = torch.max(cosine_similarity, dim=-1)
        return scores

    return confidence_score
```

Figure 3.2: DesCLIPtions Algorithm for Semantic OOD Data

receive a painting of a cat at test time, all the probability mass will be placed there, leaving to low probability to "a photo of a cat".

Figure 3.3 illustrates our second method and Figure 3.4 provides PyTorch code describing the algorithm. We note that the method does require knowledge of the kinds of OOD data that the will be passed in at test time. However, it does not need to know a specific kind. In our Experiments, we show that providing many kinds e.g. (painting, sketch, stone engraving, sculpture) is fine. In other words, one can come up with a superset. We also believe that having knowledge of kinds of the OOD data is not an unrealistic assumption, as for example all models want to guard against styles like blurry photos. Finally, we note that while it is possible to train a model to distinguish each kind on its own, e.g. detect if something is a painting, this may not be easy. To do so, the model of course we need to be fed images of the kind (as the positives), as well as every possible other kind (as the negatives). Failing to include a kind may lead unexpected results if it is encountered at test time. In other words, such a model may have its own OOD concerns.

This leads into another advantage of CLIP. CLIP is pretrained and as a result our methods

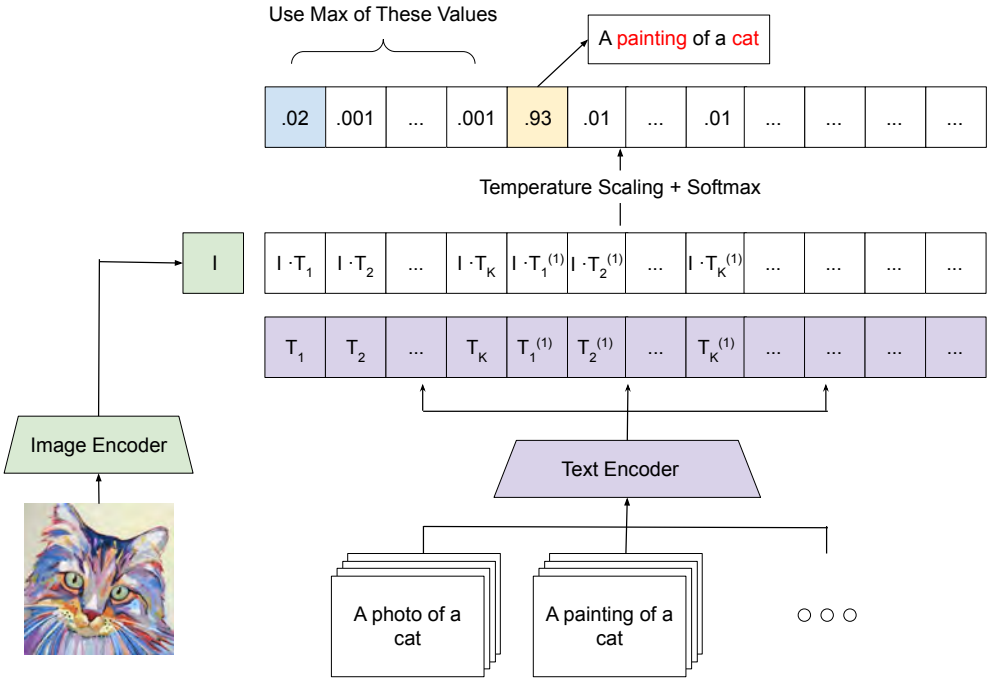


Figure 3.3: DesCLIPtions Method for Other Types of OOD Data

are zeroshot. This is nice bonus, especially compared to prior works, some of which require training VAEs [47] and GANs [36]. Additionally, though our method is semi-supervised, it does not have the issue of choosing an auxiliary Out dataset like other semi-supervised methods. We note that CLIP itself does require an enormous amount of training. However, it can be used for many cases beyond our own. Overall, we believe CLIP follows a trend seen with models like GPT-2 [45] in natural language processing where extremely large models that learn general representations are used for downstream tasks.

```python
import torch
import torch.nn.functional as F

import clip

def descliptions_other(clip_model, in_descriptions, out_descriptions, temperature=0.01):
    weights = []
    for description in in_descriptions + out_descriptions:
        text = clip.tokenize([description]).cuda()
        text_encoding = clip_model.encode_text(text)
        text_encoding = F.normalize(text_encoding, dim=-1)
        weights.append(text_encoding[0])
    weights = torch.stack(weights, dim=1)

    def confidence_score(image_minibatch):
        image_encodings = clip_model.encode_image(image_minibatch)
        image_encodings = F.normalize(image_encodings, dim=-1)
        cosine_similarity = image_encodings @ weights
        logits = cosine_similarity / temperature
        probs = torch.softmax(logits, dim=-1)
        in_probs = probs[:, :len(in_descriptions)]
        scores, _ = torch.max(in_probs, dim=-1)
        return scores

    return confidence_score
```

Figure 3.4: DesCLIPtions Algorithm for Semantic OOD Data

Chapter 4

Experiments

In this chapter, we evaluate our method on various datasets covering different types of OOD data. In particular, we cover semantic and stylistic OOD inputs. We skip over contextual OOD inputs as we are unable to find datasets for it [1]. For semantic OOD inputs, we also have a section for fine-grained settings. We evaluate on higher-resolution datasets (224x224), as that is the resolution that CLIP was trained with. For lower-resolution datasets such as CIFAR-10, we find that zeroshot CLIP provides a lower classification accuracy compared to standard classification models, and that its OOD detection AUROC is lower than baseline methods.

For our method, OpenAI has released two CLIP models, one where the image encoder is based on Resnet-50 [21] and the other on ViT-B/32 [14]. The former has roughly 100M parameters while the latter has roughly 150M. We find they perform similarly, though we include results on both for completeness. CLIP does have more parameters than the average image classification model.

4.1 Semantic OOD Data

For semantic OOD data, we evaluate on ImageNet-30, a 30-class subset of ImageNet [11] introduced by Hendrycks et al. [28]. Only 30 classes are used because fuller versions of ImageNet (e.g. ImageNet-1K) contain objects present in OOD datasets. We primarily compare against Tack et al. [52], which from our understanding is state-of-the-art. They consider one-class OOD detection, where each of the 30 classes is treated as the In dataset with the rest as the Out dataset, as well as multi-class OOD detection, where the 30 classes are collectively treated as the In dataset. In this case, the Out datasets are CUB-200 (Birds) [55], Stanford Dogs [31], Oxford Pets [42], Food-101 [3], Places-365 [60], Caltech-256 [18], and Describable Textures Dataset (DTD) [9] [2]. Example ImageNet-30 images are shown in

[1] We believe Streetview Storefronts [27], which varies images by country, may be a good choice but the dataset is not pubicly available.

[2] For ImageNet-30, we use only the test split. For the Out datasets, we use all the available images.

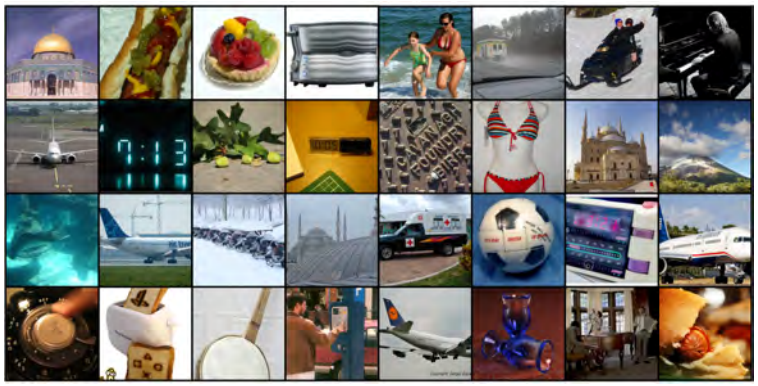


Figure 4.1: ImageNet-30 Images

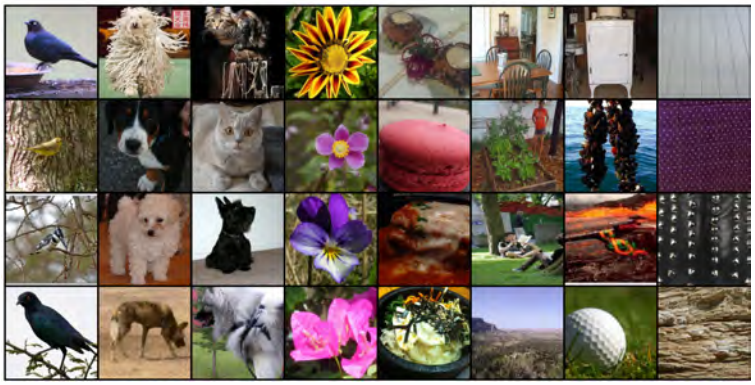


Figure 4.2: Out Images

Figure 4.1 and example Out images are shown in Figure 4.2. For the latter, the 8 columns show the 8 datasets, respectively.

For our method, we base our descriptions on the class names in ImageNet-30. For each class, we come up with a few phrases, such as "an airplane on the runway", "a passenger airplane", etc. [3]. We then apply these phrases to templates such as "a photo of ", "i took a picture of ", "an origami of ". This leads to a large variety of sentences which as previously mentioned get averaged in the encoding process. We base our templates off those used in zeroshot classification in the CLIP paper [46]. They include a wide variety of domains (e.g. the origami example) and this is desirable purely for semantic OOD detection. We note that using these sentences leads to an AUROC improvement of about 2-3% compared to simply using "a photo of an airplane", though we believe further gains can be achieved with even more tuning.

Table 4.1 shows our results for the one-class scenario, with AUROC values averaged over the 30-classes. Here, we compare against Hendrycks et al.'s [28] method with rotation

[3]The exact texts we use can be found in our code.

Method	Average AUROC
Rot	65.3
Rot+Trans	77.9
Rot+Attn	81.6
Rot+Trans+Attn	84.8
Rot+Trans+Attn+Resize	85.7
CSI	91.6
DesCLIPtions ResNet-50 (ours)	**99.7**
DesCLIPtions ViT-B/32 (ours)	**99.8**

Table 4.1: One-class results on ImageNet-30.

prediction, as well as against extensions with translation, attention, and resize prediction. These are not mentioned in our Prior Works section but are shown to further improve AUROC in their paper. Tack et al. [52]'s method is labeled CSI, and we use their results for the baselines. Both ResNet-50 and ViT-B/32 CLIP models show substantial improvement over these baselines. More importantly, the AUROC is near perfect!

Table 4.2 shows our results for the multi-class scenario, with the Birds, Dogs, etc. columns denoting when that dataset is used as the Out dataset. Here, we compare against the MSP baseline [23] on a model trained with cross entropy loss. We also compare with CSI using both their own confidence score and using a linear probe on their features (followed by MSP) [4]. We note that for the latter, the authors modify the training scheme by incorporating class label information, even though this is normally not present in SimCLR [5]. Furthermore, their method inherently also performs rotation prediction [6], which to the best of our knowledge was state-of-the-art before works with contrastive learning. We once again refer to Tack et al. for these baseline results. We also compare to MSP with pre-training [24], where the pre-trained dataset is ImageNet-1K. Our finetuned ResNet-18 achieves 98.23% accuracy on ImageNet-30. As this is above that of CLIP and other baselines, we believe it gives an accurate assessment of the method. Both ResNet-50 and ViT-B/32 CLIP models offer improvement over the baselines! Against fine-grained datasets like Birds, Dogs, Pets, Flowers, and Food, it is also near perfect. We note that our method does not offer as much improvement on Caltech-256 and DTD. We suspect Caltech-256 is difficult because it contains a wide variety of objects. Indeed, grand piano, revolver, rotary phone, snowmobile, toaster, and airplane are Caltech-256 classes that are actually also in ImageNet-30 [7]. We suspect DTD is difficult because the textures it describes, e.g. quilted, sometimes come from objects like pillows, which are in ImageNet-30.

[4] We specifically use the CSI-ens version.

[5] This is based on SupCLR [32].

[6] Rotation is one of their shifting transformations.

[7] We find a 1% increase in AUROC by pruning these samples.

Method	Average	Birds	Dogs	Pets	Flowers	Food	Places-365	Caltech-256	DTD
MSP	90.1	88.0	96.7	95.0	89.7	79.8	90.5	90.6	90.1
CSI	89.9	90.5	97.1	85.2	94.7	89.2	78.3	87.1	96.9
CSI w/ Linear Probe + MSP	95.0	94.6	98.3	97.4	96.2	88.9	94.0	93.2	**97.4**
MSP w/ Pretraining	95.7	97.8	97.6	98.1	96.6	91.0	95.3	93.3	96.2
DesCLIPtions ResNet-50 (ours)	**98.0**	99.6	**99.6**	99.1	**99.4**	98.6	97.9	94.5	95.3
DesCLIPtions ViT-B/32 (ours)	**98.0**	**99.7**	99.4	**99.2**	**99.4**	**98.8**	**98.1**	**94.8**	94.9

Table 4.2: Multi-class results on ImageNet-30.

(a) In Food (b) Out Food

Figure 4.3: Example of Food-101 images.

4.2 Semantic OOD Data in a Fine-Grained Setting

The previous section showed that CLIP is good at detecting semantic OOD inputs. However, in that evaluation, the classes of the In dataset (acorn, airplane, ambulance, etc.) were all of relatively common objects, and the classes of the Out dataset were all relatively different (birds, flowers, etc.). In this section, we test our method further by using fine-grained In dataset and Out datasets. Specifically, we perform two evaluations. The first is on Food-101, which contains 101 classes of different types of food. The second is the Oxford Flowers dataset, which consists of 102 classes of flowers. For the Food-101, we use the first 51 classes as the In dataset and the last 50 as the Out dataset. For Oxford Flowers, we use the last 51 classes as the In dataset the first 51 classes as the Out dataset [8]. Examples of In and Out images are shown in Figure 4.3 and Figure 4.4, for Food-101 and Oxford Flowers, respectively.

For our method, we again base our descriptions off the class names (e.g. apple pie, guacamole, etc. for Food-101 and sunflower, bird of paradise, etc. for Oxford Flowers). For Food-101, the class names are provided in the dataset. For Oxford Flowers, the classes are listed on the dataset website, but we modify some of the names. For example, we sometimes

[8]Our goal was to simply split the classes in half, but since the dataset is class-imbalanced and the last 51 classes have more training samples, we decided to use it as the In dataset.

(a) In Flowers (b) Out Flowers

Figure 4.4: Example of Oxford Flowers images.

Method	AUROC
MSP w/ Pretraining	85.06
DesCLIPtions ResNet-50 (ours)	82.63
DesCLIPtions ViT-B/32 (ours)	85.26

(a) Results on Food-101.

Method	AUROC
MSP w/ Pretraining	94.83
DesCLIPtions ResNet-50 (ours)	87.06
DesCLIPtions ViT-B/32 (ours)	90.61

(b) Results on Oxford Flowers.

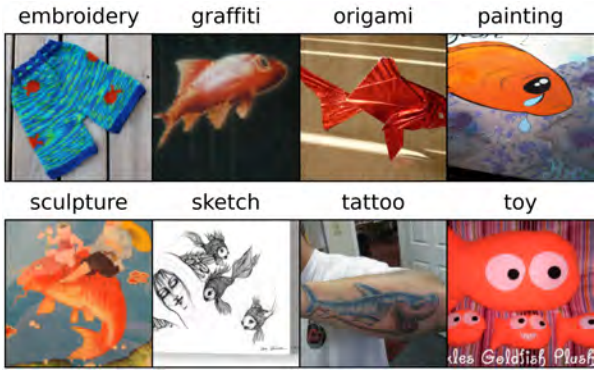
Table 4.3: Results on Fine-Grained Datasets.

change the class name to a more common name, such as "pink evening primrose" instead of "pink primrose" and "matilija poppy" instead of "tree poppy". We also believe that some of the names may be inaccurate, e.g. "watercress" seems to be me more accurately described as "nasturtium", and alter them. For templates, we only use one template, "a photo of a ". We find that using the wide variety of templates as in the ImageNet-30 benchmark leads to poor results. We suspect this is because while objects in ImageNet-30 can come in a variety of forms, every image in Food-101 and Oxford Flowers is a simple photo of the class.

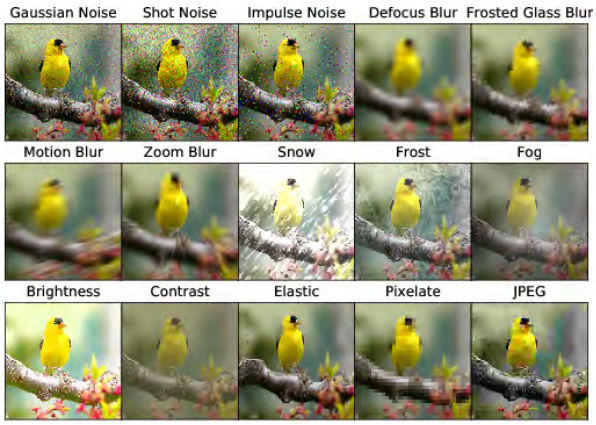
Table 4.3 shows our results. For this benchmark, we compare to MSP w/ pretraining [24]. Our finetuned classifiers on Food-101 and Oxford Flowers achieve 89.27% and 99.2% accuracy, respectively, which is standard for these datasets. While for Food-101, the ViT-B/32 CLIP model remains on par with the baseline, our method falls short for Oxford Flowers. This implies that unfortunately CLIP can not be simply used out of the box for fine-grained OOD data.

4.3 Stylistic OOD Data

Despite the results in the previous section, our method shows much promise on stylistic OOD data, which we evaluate in this section. We use the second variant as mentioned



(a) ImageNet-R renditions. We note for tattoo, the class is actually great white shark. The dataset does not have every renditon for every class.



(b) ImageNet-C corruptions, taken from their paper [22].

Figure 4.5: Stylistic OOD Data.

in the Methods section and for our datasets we use ImageNet-R [27] and ImageNet-C [22]. ImageNet-R is a dataset containing different renditions of 200 classes in ImageNet-1K. While the authors provide quite a few, we use 8 of them: embroidery, graffiti, origami, painting, sculpture, sketch, tattoo, and toy. We choose this subset as we found that some of the other categories were a bit noisy [9] [10]. ImageNet-C is a dataset containing different corruptions of ImageNet-1K, such as photos with zoom blur, gaussian noise, and snowy artifacts. We use the 15 main corruptions provided at the 3rd (middle) intensity level. For both of these datasets, it is important to detect them as OOD. We find that ImageNet-R reduces the accuracy of a classifier by two-thirds while ImageNet-C reduces by half. Figure 4.5 shows example images from these datasets.

For our method on ImageNet-R, we use the 8 different renditions to form our out descriptions. However, to show that a superset may be we used, we also add 8 additional renditions: art, cartoon, deviantart, mural, stained glass, sticker, stone engraving, and video game. For our method on ImageNet-C, we form 10 different out description sets that cover the 15 corruptions.

- "an overexposed photo of a"
- "a low contrast photo of a"
- "a stretched photo of a"
- "a blurry photo of a"
- "a foggy photo of a"
- "a frosty photo of a"
- "a noisy photo of a"
- "a pixelated photo of a"
- "a photo of a _ with JPEG artifacts"
- "a snowy photo of a"

[9] Ex. n02138441/cartoon_8.jpg is more like a sketch.
[10] Ex. n02701002/sculpture_0.jpg is LEGOs.

Method	Average	Embroidery	Graffiti	Origami	Painting	Sculpture	Sketch	Tattoo	Toy
MSP	88.52	91.6	92.6	88.7	81.7	89.2	85.4	91.3	87.9
DesCLIPtions ResNet-50 (ours)	**98.1**	99.1	99.2	**98.5**	**98.5**	97.2	**97.9**	**99.7**	**94.8**
DesCLIPtions ViT-B/32 (ours)	98.0	**99.3**	**99.4**	**98.5**	**98.5**	**97.5**	97.1	99.5	94.3

Table 4.4: Results on ImageNet-R.

Method	Average	Brightness	Contrast	Defocus Blur	Elastic	Fog	Frost	Gaussian Noise
MSP	78.8	60.8	79.6	85.6	71.5	76.9	84.5	82.3
DesCLIPtions ResNet-50 (ours)	**93.8**	**69.3**	**90.6**	**98.6**	**84.4**	88.8	**94.0**	**99.7**
DesCLIPtions ViT-B/32 (ours)	90.6	66.4	85.9	97.8	65.5	**90.8**	93.7	99.0

Glass Blur	Impulse Noise	JPEG	Motion Blur	Pixelate	Shot Noise	Snow	Zoom Blur
89.9	82.9	67.0	83.6	71.9	83.3	80.7	81.8
98.6	**99.5**	**91.6**	**98.9**	**97.9**	**99.6**	**97.3**	**97.9**
97.4	99.3	79.4	96.7	95.7	99.2	96.6	96.5

Table 4.5: Results on ImageNet-C.

We also add 10 additional descriptions to form our superset.

- "an saturated photo of a"
- "a rainy photo of a"
- "a distorted photo of a"
- "a high contrast photo of a"
- "a dim photo of a"
- "a cracked photo of a"
- "a grainy photo of a"
- "a photo of a _ with a lot of glare"
- "a photo of a _ under red light"
- "a photo of a _ under green light"

Table 4.4 and Table 4.5 show our results. Each column denotes when that rendition / corruption is used as the Out dataset. In both cases, we compare to MSP on a ResNet-50 model. Overall, our method tends to do much better! We note that for ImageNet-R, we have a bit of trouble with toy, and for ImageNet-C, a bit of trouble with brightness, contrast, and elastic transform corruptions. We suspect the brightness and contrast issue may be due to the natural variability in brightness and contrast that makes it hard to determine if a new photo is bright or low/high contrast. We also suspect the elastic transform is simply difficult since it is unlikely to be encountered in CLIP's training dataset (it seems to require a computer-engineered transformation on the image). Finally, we note that interestingly the smaller ResNet-50-based CLIP model edges out the larger ViT-B/32-based model on ImageNet-R and solidly beats it on ImageNet-C.

Chapter 5

Conclusion and Future Work

In this work, we develop distinctions between different types of OOD data (semantic vs. stylistic), propose two zeroshot OOD detection methods using pretrained natural language supervision models, and show that it performs well on certain types of OOD data. Our findings from this research (both related to our method and generally speaking) are that

- Stylistic OOD data can be harder to detect than semantic OOD data. This is shown by MSP's worse performance on ImageNet-R and ImageNet-C compared to ImageNet-30, Food-101, and Oxford Flowers. We believe future research should evaluate in this setting.
- For semantic OOD data, AUROC performance is positively correlated with classification accuracy on the in-distribution data. The publicly released CLIP models both have relatively low accuracy and AUROC on fine-grained datasets, so a bit more work is needed before they can be used in these situations.
- For MSP, the more the accuracy is lowered by a type of OOD data, the easier it is to detect.
- Natural language supervision enables a broad variety of data to be learned. Since it can describe style, it significantly improves stylistic OOD detection.

For future work, we believe evaluating CLIP with other types of OOD data (contextual, pose, etc.) would be enlightening. We also feel there is potential with finetuning CLIP. This may help it in fine-grained semantic OOD settings. However, ideally one can finetune it in a way that does not cause it to lose its performance on stylistic OOD data. Finally, recent works [26] have explored OOD detection in the context of image segmentation. Specifically, certain parts of an image are declared in-distribution and other parts OOD. We think this is an interesting problem that CLIP and natural language supervision in general can potentially still be applicable to.

Bibliography

[1] Michael A. Alcorn et al. *Strike (with) a Pose: Neural Networks Are Easily Fooled by Strange Poses of Familiar Objects*. 2019. arXiv: 1811.11553 [cs.CV].

[2] Dario Amodei et al. *Concrete Problems in AI Safety*. 2016. arXiv: 1606.06565 [cs.AI].

[3] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. "Food-101 – Mining Discriminative Components with Random Forests". In: *European Conference on Computer Vision*. 2014.

[4] Markus M. Breunig et al. "LOF: Identifying Density-Based Local Outliers". In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. SIGMOD '00. Dallas, Texas, USA: Association for Computing Machinery, 2000, pp. 93–104. ISBN: 1581132174. DOI: 10.1145/342009.335388. URL: https://doi.org/10.1145/342009.335388.

[5] Simon Byers and Adrian E. Raftery. "Nearest-Neighbor Clutter Removal for Estimating Features in Spatial Point Processes". In: *Journal of the American Statistical Association* 93.442 (1998), pp. 577–584. ISSN: 01621459. URL: http://www.jstor.org/stable/2670109.

[6] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly Detection: A Survey". In: *ACM Comput. Surv.* 41.3 (July 2009). ISSN: 0360-0300. DOI: 10.1145/1541880.1541882. URL: https://doi.org/10.1145/1541880.1541882.

[7] Jiefeng Chen et al. *Robust Out-of-distribution Detection for Neural Networks*. 2020. arXiv: 2003.09711 [cs.LG].

[8] Ting Chen et al. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020. arXiv: 2002.05709 [cs.LG].

[9] Mircea Cimpoi et al. *Describing Textures in the Wild*. 2013. arXiv: 1311.3618 [cs.CV].

[10] C. De Stefano, C. Sansone, and M. Vento. "To reject or not to reject: that is the question-an answer in case of neural classifiers". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 30.1 (2000), pp. 84–94. DOI: 10.1109/5326.827457.

[11] J. Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[12] Terrance DeVries and Graham W. Taylor. *Improved Regularization of Convolutional Neural Networks with Cutout*. 2017. arXiv: 1708.04552 [cs.CV].

[13] Terrance DeVries and Graham W. Taylor. *Learning Confidence for Out-of-Distribution Detection in Neural Networks*. 2018. arXiv: 1802.04865 [stat.ML].

[14] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2020. arXiv: 2010.11929 [cs.CV].

[15] Eleazar Eskin. "Anomaly Detection over Noisy Data Using Learned Probability Distributions". In: *Proceedings of the Seventeenth International Conference on Machine Learning*. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 255–262. ISBN: 1558607072.

[16] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: 1412.6572 [stat.ML].

[17] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[18] Gregory Griffin, Alex Holub, and Pietro Perona. "Caltech-256 object category dataset". In: (2007).

[19] Chuan Guo et al. *On Calibration of Modern Neural Networks*. 2017. arXiv: 1706.04599 [cs.LG].

[20] Simon Hawkins et al. "Outlier Detection Using Replicator Neural Networks". In: *Data Warehousing and Knowledge Discovery*. Ed. by Yahiko Kambayashi, Werner Winiwarter, and Masatoshi Arikawa. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 170–180. ISBN: 978-3-540-46145-6.

[21] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[22] Dan Hendrycks and Thomas Dietterich. *Benchmarking Neural Network Robustness to Common Corruptions and Perturbations*. 2019. arXiv: 1903.12261 [cs.LG].

[23] Dan Hendrycks and Kevin Gimpel. *A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks*. 2018. arXiv: 1610.02136 [cs.NE].

[24] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. *Using Pre-Training Can Improve Model Robustness and Uncertainty*. 2019. arXiv: 1901.09960 [cs.LG].

[25] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. *Deep Anomaly Detection with Outlier Exposure*. 2019. arXiv: 1812.04606 [cs.LG].

[26] Dan Hendrycks et al. *Scaling Out-of-Distribution Detection for Real-World Settings*. 2020. arXiv: 1911.11132 [cs.CV].

[27] Dan Hendrycks et al. *The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization*. 2020. arXiv: `2006.16241 [cs.CV]`.

[28] Dan Hendrycks et al. *Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty*. 2019. arXiv: `1906.12340 [cs.LG]`.

[29] Heinrich Jiang et al. *To Trust Or Not To Trust A Classifier*. 2018. arXiv: `1805.11783 [stat.ML]`.

[30] Armand Joulin et al. *Bag of Tricks for Efficient Text Classification*. 2016. arXiv: `1607.01759 [cs.CL]`.

[31] Aditya Khosla et al. "Novel Dataset for Fine-Grained Image Categorization". In: *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*. Colorado Springs, CO, June 2011.

[32] Prannay Khosla et al. *Supervised Contrastive Learning*. 2021. arXiv: `2004.11362 [cs.LG]`.

[33] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2014. arXiv: `1312.6114 [stat.ML]`.

[34] Edwin Knorr and Raymond Ng. "A Unified Approach for Mining Outliers". In: (Nov. 1997).

[35] Kimin Lee et al. *A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks*. 2018. arXiv: `1807.03888 [stat.ML]`.

[36] Kimin Lee et al. *Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples*. 2018. arXiv: `1711.09325 [stat.ML]`.

[37] Shiyu Liang, Yixuan Li, and R. Srikant. *Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks*. 2020. arXiv: `1706.02690 [cs.LG]`.

[38] Amit Mandelbaum and Daphna Weinshall. *Distance-based Confidence Score for Neural Network Classifiers*. 2017. arXiv: `1709.09844 [cs.AI]`.

[39] Tomas Mikolov et al. *Distributed Representations of Words and Phrases and their Compositionality*. 2013. arXiv: `1310.4546 [cs.CL]`.

[40] Eric Nalisnick et al. *Do Deep Generative Models Know What They Don't Know?* 2019. arXiv: `1810.09136 [stat.ML]`.

[41] Yaniv Ovadia et al. *Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift*. 2019. arXiv: `1906.02530 [stat.ML]`.

[42] Omkar M. Parkhi et al. "Cats and Dogs". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2012.

[43] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://www.aclweb.org/anthology/D14-1162.

[44] Stanislav Pidhorskyi et al. *Generative Probabilistic Novelty Detection with Adversarial Autoencoders*. 2018. arXiv: 1807.02588 [cs.CV].

[45] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: (2019).

[46] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV].

[47] Jie Ren et al. *Likelihood Ratios for Out-of-Distribution Detection*. 2019. arXiv: 1906.02845 [stat.ML].

[48] Gabi Shalev, Yossi Adi, and Joseph Keshet. *Out-of-Distribution Detection using Multiple Semantic Label Representations*. 2019. arXiv: 1808.06664 [stat.ML].

[49] Rasheda Smith et al. "Clustering approaches for anomaly based intrusion detection". In: (2002).

[50] C. Spence, L. Parra, and P. Sajda. "Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model". In: *Proceedings IEEE Workshop on Mathematical Methods in Biomedical Image Analysis (MMBIA 2001)*. 2001, pp. 3–10. DOI: 10.1109/MMBIA.2001.991693.

[51] C. Surace, K. Worden, and G. Tomlinson. "A Novelty Detection Approach To Diagnose Damage In A Cracked Beam". In: *Proc. of SPIE*. 1997, pp. 947–953.

[52] Jihoon Tack et al. *CSI: Novelty Detection via Contrastive Learning on Distributionally Shifted Instances*. 2020. arXiv: 2007.08176 [cs.LG].

[53] Sunil Thulasidasan et al. *On Mixup Training: Improved Calibration and Predictive Uncertainty for Deep Neural Networks*. 2020. arXiv: 1905.11001 [stat.ML].

[54] Apoorv Vyas et al. *Out-of-Distribution Detection Using an Ensemble of Self Supervised Leave-out Classifiers*. 2018. arXiv: 1809.03576 [cs.LG].

[55] P. Welinder et al. *Caltech-UCSD Birds 200*. Tech. rep. CNS-TR-2010-001. California Institute of Technology, 2010.

[56] Jim Winkens et al. *Contrastive Training for Improved Out-of-Distribution Detection*. 2020. arXiv: 2007.05566 [cs.LG].

[57] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. 2017. arXiv: 1708.07747 [cs.LG].

[58] Nong Ye and Qiang Chen. "An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems". In: *Quality and Reliability Engineering International* 17.2 (2001), pp. 105–112. DOI: https://doi.org/10.1002/qre.392. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qre.392. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/qre.392.

[59] Sangdoo Yun et al. *CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features*. 2019. arXiv: 1905.04899 [cs.CV].

[60] Bolei Zhou et al. "Places: A 10 million Image Database for Scene Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).