Generative Model-based Classifier Robust to Out-of-distribution data

Aakash Sudhirbhai Vora
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ, USA
avora8@asu.edu

Rohit Reddy Suluguri
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ, USA
rsulugur@asu.edu

Aksh Kantibhai Patel
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ, USA
apate160@asu.edu

Aditya Goyal
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ, USA
agoyal61@asu.edu

ABSTRACT

Discriminative models are widely used for classification tasks and they produce reliable results as long as the samples fall under the distribution of their training set. However, for samples outside their training set, they confidently misclassify them to belong to one of the classes. This reduces the reliability of models when used for critical applications. We propose to use a hybrid model to overcome this flaw. The hybrid model will help determine samples that do not belong to any of the classes by modeling the joint probability of the sample and its class. The hybrid model will consist of a generative model alongside a discriminative classifier. The discriminative classifier will be trained on samples that are out of distribution but lies in proximity to in-distribution samples. Our hypothesis is that the hybrid model will be resistant to misclassifying out-of-distribution samples as in-distribution, which is correctly verified through experiments.

KEYWORDS

Convolutional Neural networks, Flow models, Out-of-distribution Samples, Joint Probability, Robustness, Out-of-distribution Generalization

1 INTRODUCTION

As machine learning is gaining popularity, it is finding its applications in a lot of places. Due to this trend, more and more companies in the industry are using ML models in their applications. Out of all the applications of ML models, a large number of applications use them for classification tasks. And one of the widely used types of model for classification is the discriminative one.

The discriminative models for classification have been present for a long time and have been studied extensively. Such models try to estimate the probability of a class given the input i.e. P(y|X). These models have become very popular due to their effective learning capabilities. They can encode complex relations between input and classes very effectively.

However, the effectiveness of discriminative models comes at a cost. The performance of this model depends on the assumption that the data on which it is evaluated in the future is similar to training data. The data similar to the training data is known as in-distribution data and data which is different from the training data and does not fall in its distribution is known as out-of-distribution(OOD) data. These models, though with high train and test accuracy, tend to be overconfident on out-of-distribution data and hence perform questionably on it.

There are serious implications for using such models in real-world applications. Given the model is going to assign a high score to any data irrespective of whether it comes from the distribution or not, it becomes difficult to predict the behavior of the model given unseen data. A model trained for identifying whether something is a cat or not might show very high accuracy if it is trained on images of cats and dogs. But, the same model can end up predicting an image of an elephant as a cat.

Another issue with using such a model in the real world is the security aspect. Famous attacks like one-pixel attack[11] have shown how defenseless current state-of-the-art discriminative models are to even small perturbations to the input data. As we start using these models extensively, it would become important to address such concerns for safe and reliable deployment of the model in the real world.

To address the issues discussed earlier, researchers have explored multiple avenues. Initial approaches tried to improve the ability of discriminative data to work with OOD data by multiple methods like introducing noise or OOD data in the training dataset, changing loss functions, etc. However, each method had its assumptions which hinders its applicability. Some researchers explored a new direction and experimented with generative models. A generative model tries to learn the distribution of the input data i.e. $P(X)$. The earlier success of models like GAN showed a lot of promise in the generative approach. New likelihood-based generative models like Normalizing flow, Real-NPV flow, and Glow models[2][5][4] are highly expressive in both estimating the distribution of the data and sampling high-quality images which are realistic. When flow-based models are used for OOD classification, they show very promising results for most of the data but sometimes fail to identify the OOD data somewhat similar to in-distribution data.

Our approach is based on leveraging properties of both generative models and discriminative models i.e. hybrid models. The hybrid model tries to learn the joint distribution of both input data

and output classes i.e. $P(X, y)$. Our approach learns the joint distribution by learning P(X) through a generative model and $P(y|X)$ through a discriminative model thus providing the best of both worlds. The flow-based generative model would be able to identify most of the OOD data different from the in-distribution data. The OOD data similar to in-distribution data would be used to train an $n + 1$th classifier which classifies them as the extra class. Thus, while classifying an input, we would combine the likelihood score of both the generative model and discriminative model to decide whether to reject it or classify it.

2 RELATED WORK

In the modern world, neural networks play a major role in classification tasks. However, the unpredictability of neural networks with new or unseen data can still pose a challenge. A system with such behavior when put into practice in the real world may produce unexpected and unreliable results. The issue is specifically how to prevent trained neural network models, from classifying out-of-distribution (OOD) data in the wrong way.

Their inability to comprehend out-of-distribution (OOD) data has led researchers to investigate several strategies to assist in its recognition and assignment of lower probability to OOD inputs. Wei et al. (2022) [9] advise using the L2 norm of the logits before applying the softmax operation. This approach reduces the overconfidence of neural network predictions. DeVries et al. (2018) [1] propose a new approach called Learning Confidence for Out-of distribution Detection (LCOD) which focuses on training the deep learning neural architecture to output the confidence score for each prediction. This strategy is based on the findings that a network is more likely to give inaccurate output for an OOD input when it is unsure about a prediction. These methods, however, have drawbacks, such as intensive hyperparameter tuning, and they fail to take real-world data distribution into account.

A method for boosting the effectiveness of models during training while simultaneously taking into account data distribution was presented by Sensoy et al. (2020) [9]. To produce OOD samples of data, generative models were utilized to estimate out-of-distribution data. The classifier then utilized these samples to increase its confidence in the in-distribution data. A similar strategy was put forward by Vernekar et al. (2019) [12], who created generative models to represent OOD data, sampled from those models, and then used those models for training $n + 1$ classifiers that interpret OOD data as a new class, the None class. A method is introduced for generating effective out-of-distribution (OOD) samples using a manifold learning network and training an $n + 1$ classifier for OOD detection. By generating out-of-distribution samples, the model adds an extra class to the in-class classifier. The samples produced by the model are such that they lie on the boundary or near the boundary of the in-class samples. The model aims to create a well-defined boundary around the in-class distribution for generating samples outside of it in the paper [12]. The main problem with these techniques is that they try to model OOD data which itself is large and cannot be modeled.

Another strategy that has been considered is giving the classifiers a rejection option. The method involves examining an input before sending it to a classifier. Depending on the outcome, the input is either rejected or sent on to the model that would categorize it. Geifman, Y. et al. (2017) [3] suggested that classifiers might be able to reject an input based on the MC-dropout and SR approaches. A rejection threshold is chosen depending on the expected risk during the training phase. Another technique developed by Yin, X. et al. in 2019 [15] proposes K-binary classifiers that are trained to correspond to each class, which allows them to discriminate between data from that class and treat data from other classes as OOD. The performance of such systems depends entirely on the selection of threshold and that becomes a bottleneck. To overcome it, Shafaei, A. et al. (2018)[10] offered a generalized structure and method to assess the efficacy of such a rejection mechanism.

However, Wang, W. et al. (2017) [13] suggested that the rejection technique might also be based on how closely the input matches the distribution of the actual data. They propose training K generative model-based classifiers to understand the distribution of each class. During testing, the similarity between input data and the sample generated by each model is computed. If the similarity is higher than a threshold, input is classified as OOD else it is assigned to the most similar class.

The main issue with this method is that it can be challenging to identify an appropriate similarity score metric and that the number of models needed to train increases linearly with the number of classes. Instead of training K in many generative models for each class, we plan to build on the concept mentioned in Wang, W. et al. (2017) [13] and train a hybrid model that will represent $P(X, Y)$.

3 METHODOLOGY

We consider that we have a dataset with each point labeled with some class. The data is represented by X which is a set of vectors from R^n where n is the dimensionality of the data. Further, the set of classes is represented by y_S which is a subset to N. The dataset X has been sampled from an unknown distribution $p_{un}(X)$. Also, The OOD data is represented by X_{ood} which represents data from a distribution other than $p_{un}(X)$. The goal of a classification problem is to find an approximate function f_{approx} which models actual function $f : R^n \rightarrow y_S$. Such a definition does not account for the validation of whether a given sample x belongs to X or not. We define the safer classification problem as a mechanism that can determine whether a given x belongs to X or X_{ood} before assigning a class to it. Also, we define a new class set y which is created by adding one more class namely c_{ood} to the original set y_S.

$$p(X = x, y = c_{ood}) > p(X = x, y = c); c \in y_S \text{ and } x \in X_{ood} \quad (1)$$

Our approach is based on the key idea of modeling the joint probability distribution $p(X, y)$ instead of modeling only $p(X)$ or $p(y|X)$. Given any input x to classify, we can find value of $p(X = x, y = c)$ for all c belongs to y and if $p(X = x, y = c_{ood})$ is greater than other values than we can consider that x belongs to X_{ood} i.e. OOD data. However, forcing the constraint in Eq. 1 can be challenging given that enormous size of X_{ood}. Hence, we relax the condition and divide X_{ood} in two groups, X_{ood}^{diff} which contains data completely different X and X_{ood}^{sim} which contains data somewhat similar to X. We assume that X_{ood}^{sim} can be generated by applying various perturbation to the actual distribution X. The X_{ood}^{diff} can

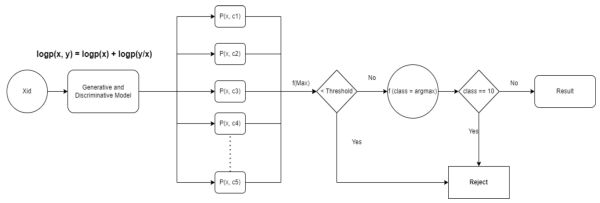

Figure 1: Test for rejecting OOD sample or classifying in-distribution data

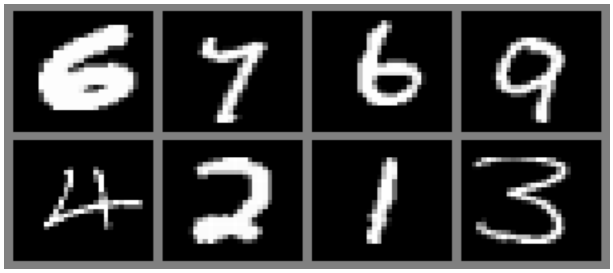

Figure 2: MNIST samples

Figure 3: TransformedMNIST samples

be identified when value of $max_{c \in y} p(X = x, y = c)$ is less than a threshold t. Whereas for identifying X_{ood}^{sim}, $max_{c \in y} p(X = x, y = c)$ should be greater than threshold t and $argmax_{c \in y} p(X = x, y = c)$ should be equal to c_{ood} hence satisfying the Eq. 1.

For modeling the joint distribution, we use both generative and discriminative models. Basically, using the Bayes theorem, we can divide $p(X, y)$ into components as follows:

$$p(X, y) = p(X) \cdot p(y|X) \tag{2}$$

$$\log p(X, y) = \log p(X) + \log p(y|X) \tag{3}$$

Hence, we can propose to use a generative model to learn $p(X)$ or $\log p(X)$ and a discriminative model to learn $p(y|X)$ or $\log p(y|X)$ separately. The discriminative model is an n+1 classifier trained on a new dataset capturing distribution of X_{ood}^{sim} created from X and synthetic OOD data generated by applying transformation on X which is assigned new class c_{ood}. We would like to highlight that compared to previous approaches using $n + 1$ classifiers, we have relaxed to requirement of training data to belong to only X_{ood}^{sim} instead of entire X_{ood}. Once both models have learned the distribution, the combined likelihood score is used to decide whether a given input is OOD data or not. The equation 4 and image 1 shows the proposed test for rejecting OOD sample or assigning a proper class to the in-distribution.

$$f(x) = \begin{cases} reject, & max_{c \in y} p(X = x, y = c) < t \text{ or} \\ & argmax_{c \in y} p(X = x, y = c) == c_{ood} \\ argmax_{c \in y} \{ & \\ p(X = x, y = c)\}, & otherwise \end{cases} \tag{4}$$

4 DATA PREPARATION

4.1 Training data

The dataset used for training is MNIST which consists of handwritten digits from zero to nine and is a popular dataset used for image classification[7]. There are sixty thousand samples in the MNIST training set each with a dimension of 28x28. For training our model, we use fifty thousand of these samples. Moreover, we artificially create OOD samples by rotating the images in the MNIST dataset (both train and test) which we call TransformedMNIST. The original training set helps in training our Flow-based model for in-distribution data. Transformed images assigned with class c_{ood} are important to train CNN to identify OOD images that are close to in-distribution samples. Hence, the CNN model is trained on data that comes from both MNIST and TransformedMNIST. Visualization of samples is shown in the figure 2 and 3.

4.2 Validation data for in-distribution training

We need to validate and tune our model to recognize samples from the distribution of MNIST. For this, we use the remaining ten thousand samples of the MNIST training set [7]. We apply the same preprocessing that we used for training. Again, the artificially created OOD samples from TransformedMNIST are also used along with the MNIST dataset. These samples help us in tuning the Flow-based and CNN models.

4.3 Test data for both out-of-distribution and in-distribution performance

The problem we are targeting has two goals, one is to correctly classify everything that falls under our data distribution i.e., the MNIST dataset. The second goal is that it should recognize samples that do not fall in our distribution. For the first goal, we use the test set from the MNIST dataset. To accomplish the second goal, we use three datasets namely, TransformedMNIST, Fashion MNIST, and Omniglot. TransformedMNIST is generated by applying rotation to test set of MNIST as discussed earlier. Like MNIST, Fashion MNIST dataset[14]. is also a popular benchmark to test models. It consists of ten classes of different clothes and accessories. For validating, out of distribution classification performance of the model, we use ten thousand samples of the test set of Fashion MNIST. Visualization of samples is shown in the figure 4.

Omniglot dataset[6] like Fashion MNIST and MNIST is popular data for benchmarking machine learning models. The Omniglot

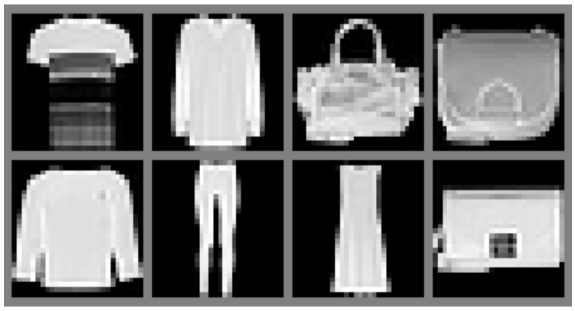

Figure 4: Fashion MNIST samples

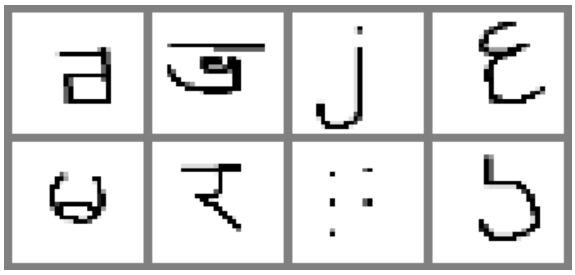

Figure 5: Omniglot samples on white background

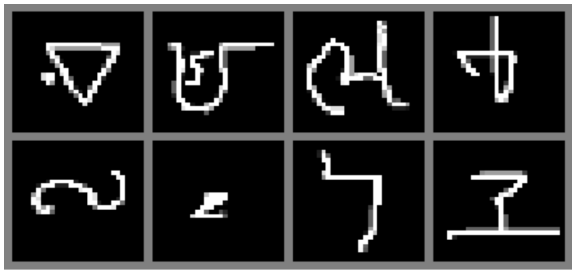

Figure 6: Omniglot samples on black background

dataset has thousand six hundred and twenty-three different handwritten characters from fifty different alphabets. The whole dataset consists of thirty-eight thousand and three hundred samples, out of which we randomly sample ten thousand instances for generating our test data. The Omniglot dataset has black images on a white background whereas images used in MNIST and Fashion MNIST have white characters on a black background. So, for images in Omniglot, we swap colors between the background and characters. Visualization of samples in the original dataset is shown in figure 5. Images after exchanging colors between background and characters are shown in figure 6. Also, images have a dimension of 105x105 whereas our training set has images of size 28x28. So, we resize images in Omniglot to be 28x28 in size.

5 IMPLEMENTATION

5.1 Generative model

We use a flow-based model as a generative component of the hybrid model. A flow-based model learns the distribution of input data $p(X)$ which can be used to find the likelihood of any input and also to sample new data points from the learned distribution. The flow models are based on the change of variable approach for the distribution.

The model maps an unknown distribution $p(X)$ to a known distribution $p(z)$. $p(X)$ here includes all the valid samples of the MNIST dataset. $p(z)$ here represents normal probability distribution, i.e., a gaussian curve. This type of transformation is called a flow[4]. Each such transformation will be invertible and differentiable so that we can calculate the $p(X)$ based on $p(z)$. The model consists of multiple layers, like dequantization layers and coupling layers. The dequantization layer's task is to convert discrete data into continuous data so that a continuous density function can be fit over it[4]. Coupling layers are the ones where actual flow transformation between $p(X)$ to $p(z)$ takes place. Having multiple such coupling layers gives more expressive flow. Coupling layers have squeeze and split layers between them. The squeeze layer's task is to reduce the size of the data by eliminating pixels so that transformation can be more efficient. The split layer's task is to split the input so that on one part multiple transformations can take place and then that result is concatenated with the second half of the input. Here, we are using a pre-trained flow model based on Flow++ architecture proposed by Ho et al. (2019)[4]. Representation of the architecture is shown in figure 7.

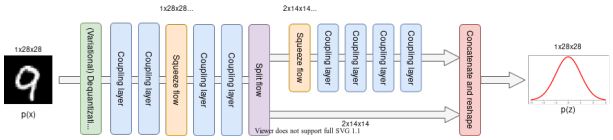

Figure 7: Architecture for flow model[8]

5.2 Discriminative model

For the discriminative model, we trained a convolutional neural network with $n + 1$ classes i.e. 11 classes. We chose CNN since it performs well on image classification tasks. The neural network consists of two convolutional layers with a relu activation function followed by a Maxpool layer, a Dropout layer, a dense layer, and another dropout layer followed by a dense layer. Since the images used are white on black background, only one channel is used as input. The convolutional layers have the output of 32 channels and 64 channels. The dropout layers use a probability of 0.25 and 0.5. The output of the last dense layer instead of being 10 is 11. Here the 11th class represents out-of-distribution classification. Representation of the model is shown in figure 8.

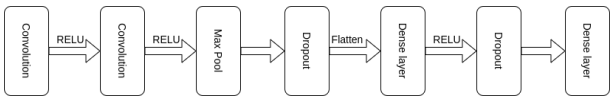

Figure 8: Architecture of Convolutional Neural Network

6 EXPERIMENT AND EVALUATION

As discussed in the approach, we divide OOD data into two groups and handle them separately, one that is very different from the in-distribution data and the other that is somewhat similar to in-distribution data. Hence, while testing the approach, we need to have a dataset for each of them. The FashionMNIST dataset acts as OOD data different from in-distribution data i.e. X_{ood}^{diff} whereas Transformed MNIST and Omniglot accounts for OOD data similar to in-distribution data i.e. X_{ood}^{sim}. As the threshold t, we used the least log-likelihood value assigned to the MNIST validation data which was -1000.

We would use images that have never been used for training or validation for the evaluation of our results. We expect that our approach would be able to identify all the images in the above dataset as OOD data instead of classifying them into one of the MNIST classes. We have selected accuracy, precision, recall, and F1 score to evaluate our approach. For OOD data, we consider the act of rejecting the sample equivalent to assigning a positive class and classifying as one of the MNIST classes as negative class. Thus, for an ideal OOD identifier, the values of metrics mentioned earlier would be very high. Table 1 shows the accuracy, precision, recall, and F1 score of our approach for identifying and correcting flagging the images from FashionMNIST, TransformedMNIST, and Omniglot datasets as OOD data.

Dataset	TransformedMNIST	FashionMNIST	Omniglot
Accuracy	97.73%	99.75%	65.12%
Precision	1.00	1.00	1.00
Recall	0.98	1.00	0.65
F1 Score	0.99	1.00	0.79

Table 1: Results for identifying OOD data

Furthermore, the end goal of our research is to create a classifier that can also classify the in-distribution data and assign the correct label to it. The classifier based on our approach would be useless if it can identify OOD data but is unable to classify in-distribution data. Hence, we need to also validate the performance of our approach on the classification of data in the test set of the MNIST dataset. Again, we have considered accuracy, precision, recall, and F1 scores as our metrics to get a fair evaluation of the performance of the approach as a classifier for in-distribution data across all MNIST classes. Table 2 and 3 summarize the accuracy, precision, recall, and F1 score for each class in the MNIST dataset.

7 DISCUSSION

To analyze and understand the performance of the approach for identifying OOD data, we need to dive deeper and look at the likelihood values assigned to them and compare them with those assigned to in-distribution data. We expect the model to assign high log-likelihood values i.e. $logp(X, y_S)$ for an image only if it belongs to in-distribution data. Thus, if an image belongs to OOD data, it would have likelihood values for each class less than some threshold value or it would have assigned high log-likelihood to OOD class i.e. $logp(X = x, y = c_{ood})$, where c_{ood} depending on whether images are very different or similar to in-distribution data respectively.

Class	Precision	Recall	F1 Score
0	0.97	0.97	0.97
1	0.99	0.99	0.99
2	0.97	0.96	0.96
3	0.97	0.97	0.97
4	0.98	0.97	0.98
5	0.97	0.97	0.97
6	0.97	0.97	0.97
7	0.97	0.96	0.96
8	0.96	0.95	0.96
9	0.97	0.95	0.96

Table 2: Results for classifying MNIST data

Accuracy	Avg. Precision	Avg. Recall	Avg. F1 Score
97%	0.97	0.96	0.96

Table 3: Results for classifying MNIST data with overall accuracy and average precision, recall and F1 Score over all classes i.e. 0,1,..,9

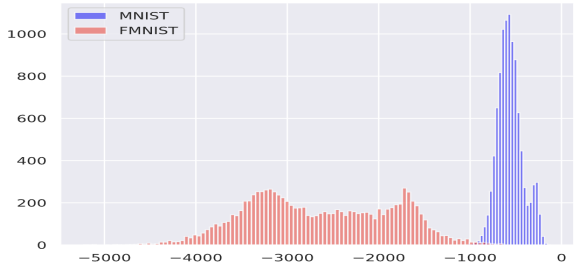

Figure 9: $maxlogp(X = x, y = c)$ distribution for FashionMNIST

The image 9 shows the distribution log-likelihood values assigned to FashionMNIST data in comparison to that assigned to MNIST data. Since for each image, there are n+1 log-likelihood values corresponding to each class i.e $logp(X = x, y = c)$ for $c \in y$, we have considered the maximum value of $logp(X = x, y = c)$ as we expect our approach to assigning lower maximum likelihood to OOD data. We can see that the distributions are well separated from each other showing that our approach can classify FashionMNIST as OOD data correctly.

For TransformedMNIST and Omniglot, since the log-likelihood values are larger than the threshold, we need to check whether our approach correctly assigned high log-likelihood values to $log\ p(X = x, y = c_{ood})$ as compared to log-likelihood values assigned to other classes i.e $\max_{c \in y} logp(X = x, y = c) - logp(X = x, y = c_{ood})$ must be negative for all images. Whereas for in-distribution data i.e MNIST, low log-likelihood should be assigned to $logP(X = x, y = c_{ood})$ as compared to other classes and hence $\max_{c \in y} logp(X = x, y = c) - logp(X = x, y = c_{ood})$ must be positive for all images. The image 10 and image 11 confirm the above expectation correctly for TransformedMNIST and Omniglot datasets respectively.

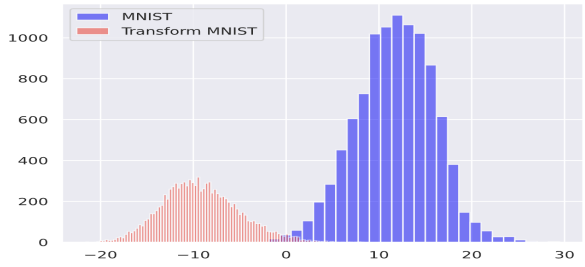

Figure 10: $\max_{c \in y} logp(X = x, y = c) - logp(X = x, y = c_{ood})$ **distribution for TransformedMNIST**

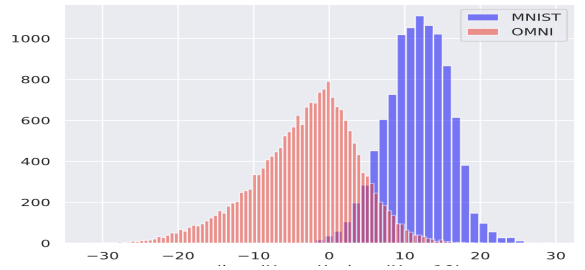

Figure 11: $\max_{c \in y} logp(X = x, y = c) - logp(X = x, y = c_{ood})$ **distribution for Omniglot**

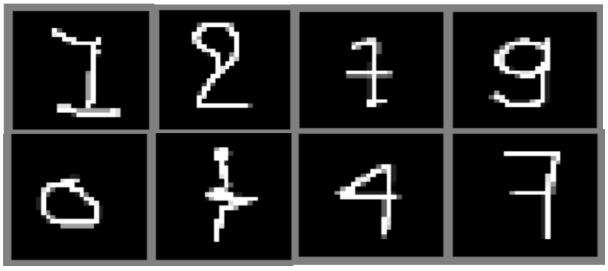

Figure 12: Number-like samples incorrectly identified as in-distribution from Omniglot dataset

It is observed that results on Omniglot data are not as good as TransformedMNIST and FashionMNIST. To explain this behavior, we analyzed the images that were incorrectly classified as in-distribution images and observed that large numbers of them contain digit-like symbols that resemble very closely to MNIST images. Image 12 shows some of the misclassified images.

8 CONCLUSION

In summary, we have developed a hybrid model that can classify samples as either in or out of distribution, as well as correctly identify their label. We relax the requirement of data for training $n + 1$ classifier to only perturbed data which can be easily generated. Our approach trained using the MNIST dataset and tested on TransformedMNIST, FashionMNIST, and Omniglot, resulted in an OOD classification accuracy of 97.73%, 99.75%, and 65.12% respectively along with in-distribution classification accuracy of 97%. These results demonstrate that our model can successfully identify samples that do not belong to the known distribution without affecting performance on the classification task.

In conclusion, we introduced a new model that combines the best parts of two existing models to better classify data. Our model can accurately distinguish between in and out-of-distribution samples, and identify the correct label for each sample. We believe our model has the potential to make machine learning models more reliable and effective in a variety of real-world situations.

9 FUTURE WORK

This model can be tested on more complex datasets such as ImageNet to assess its effectiveness since ImageNet has more classes and more channels per image compared to MNIST. Additionally, as our approach has only been assessed on an image dataset, different types of data such as text and audio can be used to check whether the proposed approach is data agnostic.

REFERENCES

[1] Terrance DeVries and Graham W. Taylor. 2018. Learning Confidence for Out-of-Distribution Detection in Neural Networks. *arXiv preprint arXiv:1802.04865* (2018). https://arxiv.org/abs/1802.04865

[2] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803* (2016). https://arxiv.org/pdf/1605.08803

[3] Yonatan Geifman and Ran El-Yaniv. 2017. Selective classification for deep neural networks. *Advances in neural information processing systems* (2017). https://proceedings.neurips.cc/paper/2017/file/4a8423d5e91fda00bb7e46540e2b0cf1-Paper.pdf

[4] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. 2019. Flow++: Improving flow-based generative models with variational dequantization and architecture design. (2019). http://proceedings.mlr.press/v97/ho19a/ho19a.pdf

[5] Durk P Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems* 31 (2018). https://proceedings.neurips.cc/paper/2018/file/d139db6a236200b21cc7f752979132d0-Paper.pdf

[6] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science* 350, 6266 (2015), 1332–1338. https://doi.org/10.1126/science.aab3050

[7] Yann Le Cunn. 1999. *THE MNIST DATABASE of handwritten digits.* http://yann.lecun.com/exdb/mnist/

[8] Phillip Lippe. 2022. *Normalizing Flows for image modeling.* https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial11/NF_image_modeling.html

[9] Murat Sensoy, Lance Kaplan, Federico Cerutti, and Maryam Saleki. 2020. Uncertainty-aware deep classifiers using generative models. (2020). https://ojs.aaai.org/index.php/AAAI/article/view/6015

[10] Alireza Shafaei, Mark Schmidt, and James J Little. 2018. A less biased evaluation of out-of-distribution sample detectors. *arXiv preprint arXiv:1809.04729* (2018). https://arxiv.org/abs/1809.04729

[11] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* (2019). https://ieeexplore.ieee.org/iel7/4235/4358751/08601309.pdf

[12] Sachin Vernekar, Ashish Gaurav, Vahdat Abdelzad, Taylor Denouden, Rick Salay, and Krzysztof Czarnecki. 2019. Out-of-distribution detection in classifiers via generation. *arXiv preprint arXiv:1910.04241* (2019). https://arxiv.org/pdf/1910.04241

[13] William Wang, Angelina Wang, Aviv Tamar, Xi Chen, and Pieter Abbeel. 2017. Safer classification by synthesis. *arXiv preprint arXiv:1711.08534* (2017). https://arxiv.org/pdf/1711.08534.pdf

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

[15] Xuwang Yin, Soheil Kolouri, and Gustavo K Rohde. 2019. Gat: Generative adversarial training for adversarial example detection and robust classification. *arXiv preprint arXiv:1905.11475* (2019). https://arxiv.org/pdf/1905.11475