Robust Novelty Detection with a Dynamic Safety Function

Nipun Agarwala and Gleb Shevchuk
Stanford University
{*nipuna1, glebs*}*@stanford.edu*

***Abstract*— Currently, novelty detection is done by using traditional clustering methods, outlier detection, or unsupervised neural networks. Though these methods have had some success, there lacks a robust method to do novelty detection with multiple feature spaces. Additionally, in robotics, novelty detection methods are not informative enough to influence a dynamic safety function. Hence, previous work only use a global, static safety function when novelty is detected. In this paper, we introduce a framework to perform more robust novelty detection and use novelty to guide a dynamic safety function. To perform novelty detection, We utilize Variational AutoEncoders, along with unsupervised clustering methods like K-means and HDBSCAN to collect relevant features about image data. By extracting PCA and SIFT features and Bhattacharya distance metrics, we then use novelty detection to deduce a dynamic safety function. In the course of our research, we found that our method robustly detects novelty and has promising preliminary results for a dynamic safety function.**

I. MOTIVATION

In navigation, robotic systems often encounter unknown or unrecognizable objects on previously traversed paths. It is often difficult to detect these and, if they are detected, it is difficult to adjust the navigation path to safely adjust to the obstacle. This adjusting problem is most relevant in robotic agents that move across environments, say a self driving car or a robotic arm. However, the problem can also be generalized to a problem of novelty. That is, when an agent is a new environment, how do they adjust their process, and do so in a safe way? Currently, if a system detects a novel object, it reverts to a safe prior, which is almost always a static-defined action, such as "Stop", "Wait for human input", or "Slow down". However, it seems like a more dynamic, environment-specific safety function would be needed in order for systems to work safely in real-world environments.

However, before we can begin to define what this more dynamic safety function looks like, we would like to comment on why novelty seems like the most reliable approach to understanding safety. Robotic systems' processes are built around environmental needs. A self driving car has to take into account the road, pedestrians, and surroundings. And, usually, it performs well because it has been trained in a known environment. In other words, it is safe because it has had previous exposure to elements in the environment. The real issue of safety, however, comes when the car faces an object or situation that it has never been trained on before. That is why novelty detection seems like the most viable method to improving safety. However, in previous work [1], novelty detection is only used to revert to a safe, static prior, and does not actually contribute much to the safety of the system. Therefore, the motivation for our research is to see if there exists a way to better integrate information from novelty detection into safety decision systems.

II. RELATED WORKS

A. Novelty Detection

Novelty detection involves learning the distribution of a dataset and classifying new instances as familiar or novel. More specifically, novelty detection of images uses image representations to learn the familiarity of new images. Initially, novelty detection used either probabilistic or distance-based methods like K Means Clustering [2]. However, most of these methods, like discordancy tests [3], rely on the training data being continuous and following a Gaussian distribution[4]. New methods that use Deep Neural Networks have improved the accuracy of image novelty detection systems in Robots. Recent work [1] has shown use of autoencoders to do novelty detection while having safety priors based on free-space distance and current velocity. But with these deep models being vulnerable to adversarial examples and finding it hard to generalize [5], there is a need to incorporate robustness into the predictions directly to allow for safe navigations of robotic systems. Similarly, subspace-sampling based methods, like PCA [6], project high-dimensional data into lower dimensions to capture features, but tend to capture local features over global features in a dataset.

B. Safe Navigation

Various approaches have been taken to ensuring agent and environment safety in robotic navigation tasks. In [1], the authors propose using a deep learning model and novelty detection to inform a robot's movement. More specifically, a perceptron model is used to guide an agent's actions but, if novelty is detected, control is modified by a safe prior. However, this prior is statically defined. Meanwhile, in [7], sensor data is used to compute safe regions from which a robot can make the most salient observations to map its environment. However, little is done to inform the robot's trajectory between safe regions. Meanwhile, in [8], the authors propose a safety framework based on Hamilton-Jacobi reachability analysis, but the complexity of reachability analysis makes it difficult to apply in complex systems. Lastly, [9] uses Gaussian Processes to approximate the robot's model and compute a safe set, but this safe set cannot be computed in real time.

III. METHODOLOGY

We will first introduce, in separate sections, each individual algorithmic component of our framework and then explain our proposed framework at the end.

A. Variational AutoEncoder

Variational AutoEncoders (VAE) [10] were introduced to learn the latent variables z from which the data $X = \{x^i\}_{i=1}^N$ were generated. Learning distributions for the dataset of a particular task at hand is desirable since distributions generalize better and capture more intricate properties about X due to the need for generative power. Specifically, given the conditional distribution $p_\theta(x|z)$, we want to approximate the posterior distribution $q_\phi(z|x)$ using the reparametrization trick of the variational bound. The reparametrization trick involves assuming that $q_\phi(z|x) \sim \mathcal{N}(\mu_z, \Sigma_z)$ and only learning μ_z and Σ_z as needed. We also assume that X has i.i.d. samples i.e. Σ_z is a diagonal matrix.

The μ_z and Σ_z are learnt using a Deep Neural Network called an AutoEncoder. The autoencoder will reduce high dimensional $x^i \in R^n$ to a much lower dimensional $\mathcal{N}(\mu_z, \Sigma_z)$, where $\mu_z \in R^m$ and $\Sigma_z \in R^{m \times m}$ for $m \ll n$ and $m, n \in \mathbb{N}$, using an encoder network. There is also a decoder network to reconstruct the image. A KL divergence regularization is needed to learn the appropriate distribution, apart from an $L2$ reconstruction loss. VAEs have become extensively popular recently for their ability to efficiently learn distributions and provide generative power to usually discriminative neural networks. Traditionally, they have involved fully connected layers and non-linearities in an alternating and dimension reduction fashion. But in recent times, with growing usage of images, convolutional layers and batchnorm is also being used to extract spatial features and prevent covariate shift before learning the corresponding μ_z and Σ_z. A diagram of our architecture is given in Fig1

B. K-means Clustering

K-means is a traditional clustering method used to partition n points in a vector space $\mathcal{R}^n$ into k clusters $S = \{S_1, S_2.., S_k\}$, $k \in \mathbb{N}$ and $k \leq n$ so that the following objective is minimized:

$$\arg\min_{S} \sum_{i=1}^{k} \frac{1}{S_i} \sum_{x,y \in S_i} \|x - y\|_2^2$$

Here, k is a hyper-parameter. Being an NP hard problem, iterative heuristic methods are used to solve this optimization objective, potentially leading to different solutions everytime (based on the initialization) and convergence into local optima. Intuitively, each point p is assigned to some cluster S_i. As a result, doing anomaly detection gets slightly tricky since it would required manual definition of boundaries or areas denoting anomalies and thus adding another hyper-parameter. In other words, K-means is an effective partitioning algorithm but may not always be effective in novelty detection. But it has been shown to be effective for various tasks, lending it to be considered here. We will perform comparisons to see whether this hypothesis is true . Additionally, the Euclidean L2 distance metric can potentially be replaced by other metrics like the Manhattan L1 distance, cosine similarity or Chebyshev distance.

C. HDBSCAN Clustering

Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [11] is a novel clustering method proposed based on density estimates of core and exemplar points rather than simple partitioning of the feature space. As part of the algorithm, a mutual reachability metric between point a and b is defined which is as follows:

$$d_{\text{mreach-k}} = \max\{\text{core}_k(a), \text{core}_k(a), d(a, b)\}$$

where k is the kth nearest neighbor or the core distance, and $d(a, b)$ is the distance metric chosen, like Euclidean $L2$ norm or the Manhattan $L1$ norm. A larger core distance forces the algorithm to form clusters that are denser and larger. Now, considering these mutual reachability distances, we can create a graph weighted inversely with the distance : the larger the distance, the less important it is. This graph is created using a graph algorithm like Prim's. This might result in a large highly connected tree, but it might also lead to a highly fragmented tree with points connected to only a few others. Thus, we need a hyper-parameter called minimum cluster size to enforce how big we really want the clusters to be to be even considered a cluster. Finally, we want the clusters to be robust to noise i.e. be persistent for small changes. Thus, using a persistence metric measured by the number of leaves the points in the cluster have and how deep the tree is below those points, the algorithm makes relevant changes to ensure that the cluster is actually robust and valid.

Such a method will only create clusters $C = \{c_1, ..., c_r\}$ that are similar and dense under the distance metric d in some feature representation $f : R^n \to R^m$. The points inside the cluster are given the cluster label $l \in \{1, 2, ..., r\}$, and those not assigned to any cluster are labeled as -1, lending the algorithm naturally to anomaly detection. Additionally, each cluster has a cluster exemplary which acts as the representative point of the cluster, thus allowing new points to be placed onto this clustering and compared using the appropriate distance metric. We can also measure the probability of each point being assigned to a cluster c_i, whether the point p is an anomaly or not. Due to the robustness and natural linkage of this method to our framework, this method was an additional avenue we wanted to explore.

D. PCA and SIFT

To enable effective clustering, we needed to perform dimensionality reduction and extract features which were then used in the respective algorithms. Principal Component Analysis (PCA) has shown to be simple and effective for doing so. In can be formulated in terms of a Singular Value Decomposition problem such that $X = \frac{1}{N} U \Sigma V^T$, where $U\Sigma$ are the principal components, V is the principal directions and N is the number of samples (for normalization sake,

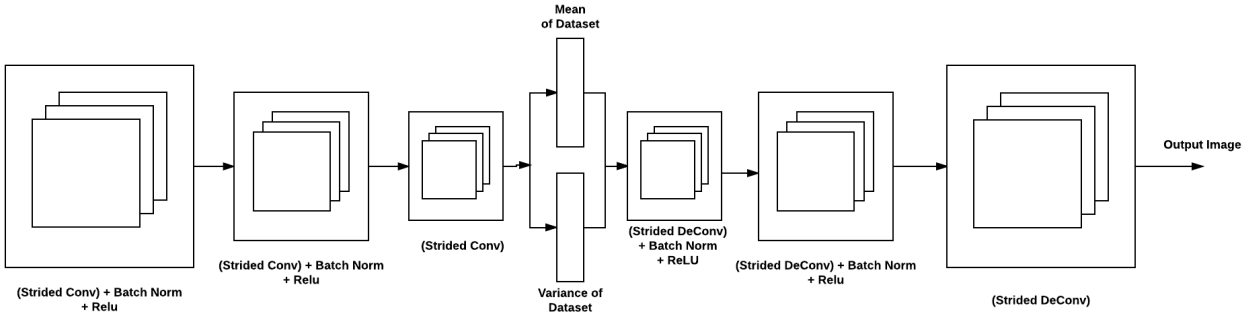


Fig. 1: Convolutional Variational AutoEncoder

if needed). Our reduced dimensionality dataset becomes $X_r = \frac{1}{N} U_r \Sigma_r V_r^T$.

Scale-Invariant Feature Transforms (SIFT) is an algorithmic unsupervised method in traditional computer vision to extract keypoints and then features from images. Keypoint descriptors are first extracted from the image based on histogram and binning of pixels techniques, resulting in each keypoint being $K_i \in \mathcal{R}^{128}$, for $1 \leq i \leq n$. These features tend to be sparse, and hence computing distances, measuring differences and similarities is easier.

E. Bhattacharya Distance

Our VAE learns the μ_z and Σ_z corresponding to the latent variable z of an input dataset X. In reality, when we encode two separate inputs, x_1 and x_2 for an appropriately trained network on dataset X, then $\mu_1 \sim \mu_2$ but not necessarily equal. This applies to Σ_1 and Σ_2 also. These differences might be magnified when compared to unseen data. Since we only have the μ and Σ of the normal distribution, we can use the following distance metric called the Bhattacharya Distance:

$$D_b = \frac{1}{8}(\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2) + \frac{1}{2} \ln \left(\frac{\det \Sigma}{\sqrt{\det \Sigma_1 \det \Sigma_2}} \right)$$

where $\Sigma = \frac{1}{2}(\Sigma_1 + \Sigma_2)$. This metric lends itself more naturally than, say, a KL divergence does, though either should suffice.

F. Proposed Framework

Given the above algorithmic tools, we developed a framework to do novelty prediction and predict the safety action dynamically in the novel cases. Let $X = \{x_i\}_{i=1}^N$ be the training set for a task T. Now, let $X' = \{x'_i\}_{i=1}^N$ be the test set, which can either be pre-collected, or seen sequentially in a streaming fashion. We train our VAE on X and find μ_a and Σ_a, corresponding to the average mean and variance of input X. Each example x_i will now give a μ_i and Σ_i for the encoder part of the VAE. We can define the average Bhattacharya Distance for X as

$$D_b^a = \frac{1}{N} \sum_{i=1}^{N} D_b(\mu_i, \mu_a, \Sigma_i, \Sigma_a)$$

Now, extract features for X using a feature extractor function $f : R^{m \times n} \rightarrow R^w, m, n, w \in \mathbb{N}$ (we use SIFT and PCA, but others could work depending on the domain) . We also cluster X using K-Means or HDBSCAN into $k \in \mathbb{N}$ clusters $S = \{S_1, S_2, ..., S_k\}$ with exemplars (or centroids) $C = \{C_1, C_2, ..., C_k\}$ and a distance metric d. On seeing S, we define safety actions $A = \{a_1, a_2, ..., a_k\}$, a_i corresponding to cluster S_i. This is acceptable since the task is known most of the times before hand along with the training set, and hence we can always pre-computer the clustering and define the safety functions. Assigning a_i to S_i acts as an expert mapping such that $g(S_i) = a_i$ for some highly non-linear $g : R^w \rightarrow R$ that is defined implicitly by the expert annotator. The expert annotator also defines a_s as the absolute safety function in the worst case uncertainty event.

Now, let us took at a training example x'_i. We first run inference on the VAE to find μ'_i and Σ'_i. Using these, we can find $D'_b = D_b(\mu'_i, \mu_a, \Sigma'_i, \Sigma_a)$. D'_b should be larger for novel scenarios and smaller for similar ones. Additionally, x'_i is clustered using K-Means or HDBSCAN using centroids C and distance metric d to get a label $l_i \in \{-1, 1, 2, ...l_k\}, 1 \leq i \leq k$ and probabilities of similarity to each cluster $p_i = \{p_i^1, ..., p_i^k\}$. With all this, we can define our new dynamic safety action as:

$$a'_{s_i} = \sum_{r=1}^{k} p_i^r a_r + \mathbb{1}(l(x'_i) = -1) \frac{|D_b^a - D'_b|}{D_b^a} a_s \text{ or}$$

$$a'_{s_i} = \sum_{r=1}^{k} p_i^r g(S_r) + \mathbb{1}(l(x'_i) = -1) \frac{|D_b^a - D'_b|}{D_b^a} a_s$$

We use $\frac{|D_b^a - D'_b|}{D_b^a}$ to normalize the difference in the Bhattacharya constants in order to align it with $0 \leq p_i^r \leq 1$, which are also normalized.

IV. Datasets

In the course of our research, we used three datasets to test various aspects of our proposed method. First, we used the MNIST dataset to evaluate our novelty detection system. Then, we used a Toy dataset to apply novelty detection to action decision. Finally, we used a FETCH dataset to test our proposed method on a real-world robot. We will briefly discuss these three datasets below.

A. R-MNIST Dataset

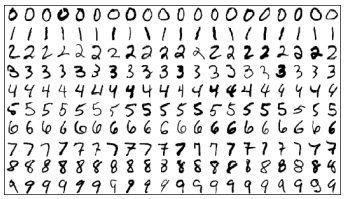


Fig. 2: R-MNIST Dataset

The MNIST dataset is composed of 60,000 examples of 28 pixel by 28 pixel, centered, one channel, images of handwritten letters from 0 through 9. Roughly 50,000 images were used for training and roughly 6,000 for testing. In our experiment, we used a reduced-MNIST, or R-MNIST, dataset, where we trained the VAE model on all classes of images except one. More specifically, we trained the model on images of 0-8 handwritten digits. Then, we ran images of the handwritten letter 9 and used the output of the model to test the hypothesis that images of the letter 9 would be classified as novel.

B. Simulated Dataset (Toy)

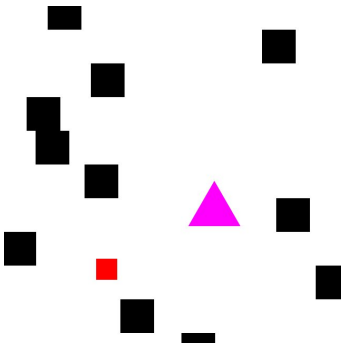


Fig. 3: Toy Dataset

The toy dataset is composed of 3-channel, 640 pixel by 640 pixel images and corresponding x positions from 0 to 640, discretized in interavls of 10, that is of the form $\{0, 10, 20, 30, .., 640\}$, from a basic simulation of an agent moving along a path from a start point to a finish point while avoiding obstacles. Above in 3, we see that the agent is represented as a red box and obstacles are represented as either black boxes or pink triangles. In the simulation, the agent moves left to right to avoid obstacles as it approaches them at a constant velocity. Then, once it reaches a final state, denoted as a bright green box, the simulation terminates. In the training phase of this experiment, a human controlled the x position of the red agent as it moved at constant velocity. In this initial training configuration, the environment only had black, square obstacles. Then, as the human maneuvered the agent to the finish state without colliding with the obstacles, the system captured each frame of the simulation along with the corresponding x position to that frame. This initial run served as training data, effectively creating an expert policy for the robot to look to in future runs through the same environment. Then, for the testing portion of this dataset, we used a similar environment, but inserted pink, triangular obstacles at random positions in the environment. Therefore, this dataset contains images and corresponding action pairs of two runthroughs of this environment, one with standard black obstacles and one with novel obstacles.

C. Fetch Robot Dataset

Fig. 4: Fetch Dataset

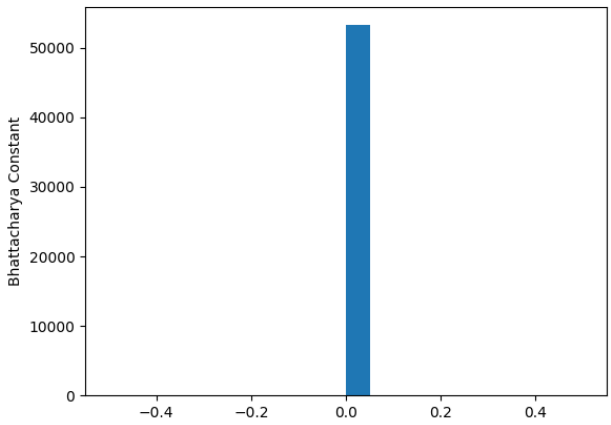
The Fetch dataset is composed of videos and corresponding robot state information of four runthroughs of a Fetch Research Edition Robot along a pre-assigned trajectory. More specifically, each dataset contains 640 pixel by 480 pixel RGB frames from a Primesense Carmine 1.09 short-range RGBD sensor calibrated in the 0.35-1.4m range and corresponding forward velocity and z-rotational velocity at each time frame. The forward velocity was captured in meters per second and z-angular velocity was captured in radians per second. In each of the four runthroughs, the robot was controlled by a human operator using a joystick controller. The simplest runthrough involves the robot moving from one edge of a room to the opposite edge, rotating 90 degrees clockwise and moving forward until it reaches a wall. The three other runthroughs build on this same general path, but introduce obstacles. These are either chairs, boxes, or both inserted randomly in the same path, such that the robot must adjust its path to avoid said obstacles. In 4, we see a sample frame of a video collected during the first run-through of the Fetch robot. Because of lack of time, however, we were unable to analyze this dataset, as we did with the Simulation dataset.

V. Experimental Setup

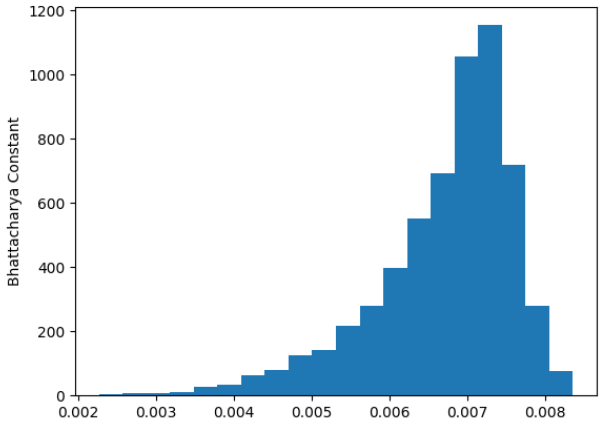
A. Variational AutoEncoder

The encoder of the VAE had 2 Convolutional layers, each having 3×3 filters, stride of 2 and a zero padding of 1 throughout the border. The first convolutional layer has 64 filters while the next has 128 filters. Each Convolutional layer is followed by a BatchNorm and ReLU layer. This is followed by 2 fully connected layers, each having 64 units and an accompanying ReLU layer. Finally we encode our μ and Σ using separate fully connected layers, each of size 16.

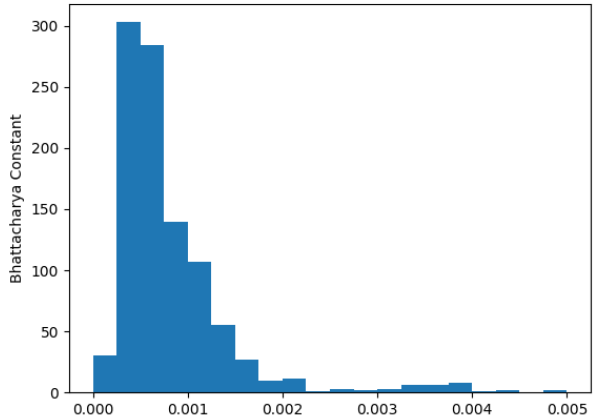
Our decoder inverts the encoder. We have 2 separate fully connected layers converting $\mu \in R^{16}$ and $\Sigma \in R^{16}$ to



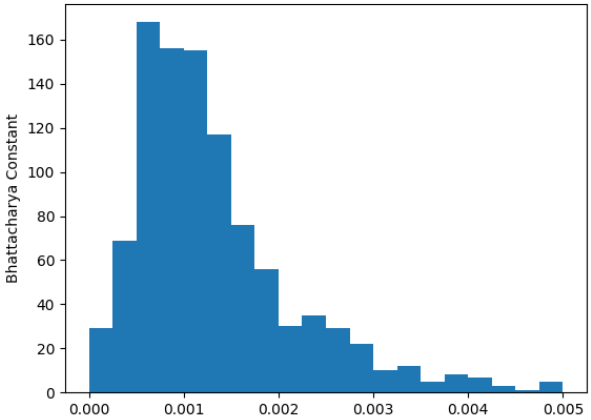
(a) MNIST Train Distributions



(b) MNIST Test Distributions

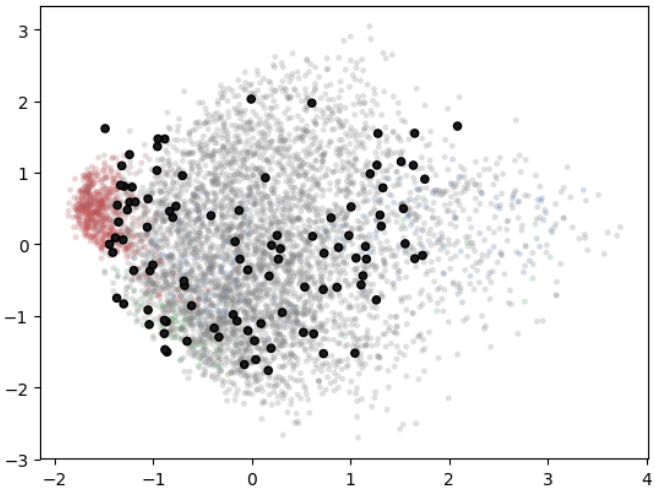


(c) Toy Train Distributions

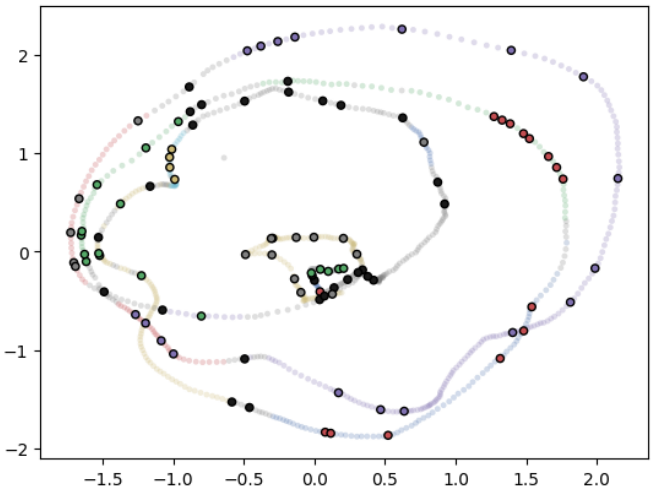


(d) Toy Test Distributions

Fig. 5: Bhattacharyya Distributions for MNIST/Toy Train/Test Datasets



(a) MNIST HBDSCAN



(b) Simulation HBDSCAN

Fig. 6: HBDSCAN Plots

R^{64} space. Then we have 2 fully connected layers, each of 64 units and a RELU non-linear activation. Finally, we use deconvolutional layers of the same parameters as the convolutional layers to reconstruct the image. We use Adam as our optimizer with learning rate 10^{-3}.

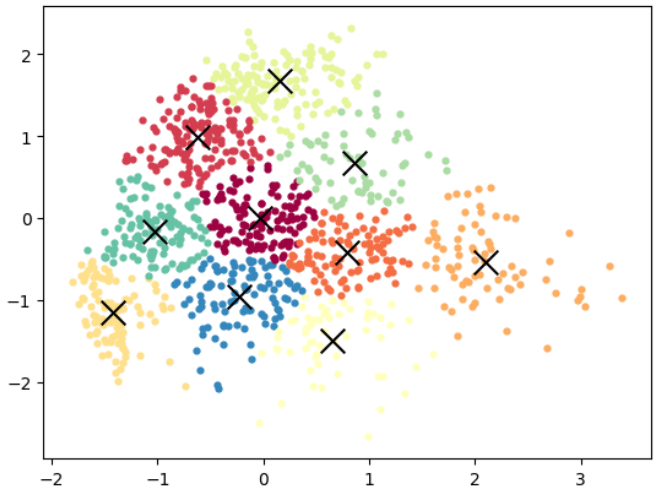


Fig. 9: KMeans Clustering Test

B. Clustering

Our PCA dimensions range from 2 to 50, depending on the clustering algorithm used. SIFT features for $3-5$ keypoints were used and concatenated into a single vector. For our HDBSCAN algorithm, we use minimum cluster sizes between 5 and 20. The number of clusters for our K-means algorithm ranged from $2-18$. We only used a few distance metrics like L1, L2 and cosine distance for both the algorithms.

VI. Results and Analysis

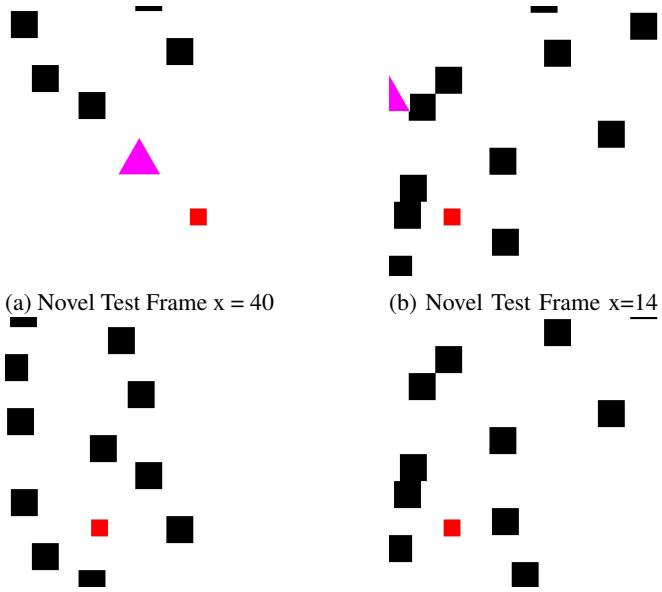
A. Novelty Detection : VAE

We first ran our proposed framework on the MNIST dataset. Though there were no actions associated with the dataset, we wanted to use it as a proof-of-concept for only the novelty detection aspect. For the VAE, we get D_b^a for the train to be $D_b^a = 0$ and the average D_b' across all test examples (having only label 9) to be 5.5×10^{-4}. This shows that our VAE successfully detected novel objects only using D_b.

To see whether this was robust in general, we plotted the distribution of D_b and D_b', as seen in fig.5(a) and fig.5(b). We see that all images with label 9 have a much larger D_b' and can be easily detected simply by comparing it to the base D_b.

Let us look at the Bhattacharya distances for the Simulated toy dataset. We see that the average D_b for the train is 1.86×10^{-6} and the average D_b' for the test is 1.27×10^{-5}, showing about an order of magnitude difference and thus detecting the novelty of the new environment.

Again, to understand the robustness, we see the distribution of D_b and D_b' as provided in fig.5(c) and fig.5(d). We observe an overlap between the distributions here (as expected) but the means of the distributions are markedly different and the D_b' distribution has a long and distinct tail. We can make a few inferences from these histograms. There are similarities between the train and test dataset i.e. some frames are highly similar if not the same, resulting in an overlap between the distributions. But the frames having the pink triangles do have distinctly higher and more diverse D_b', thus showing that our novelty detection using VAE does work.



(a) Novel Test Frame x = 40

(b) Novel Test Frame x=14

(c) Not Novel Test Frame x = 20

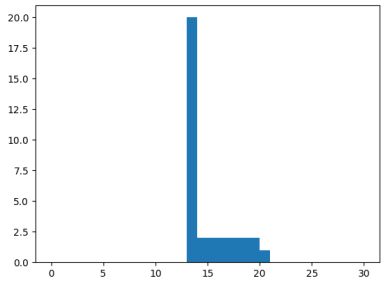
(d) Not Novel Train Frame x = 14

Fig. 10: Toy Dataset Frames with Novelty and Action

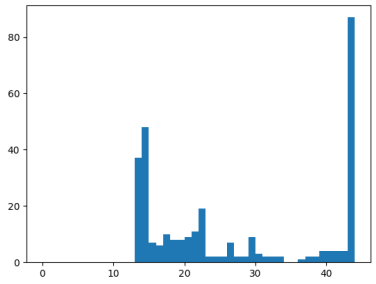
B. Novelty Detection : Clustering

Neural networks can be susceptible to adversarial examples and hence we would like to have an unsupervised method to also perform novelty detection. Thus, we evaluate using K-means clustering and HDBSCAN to see whether they go along with the VAE.

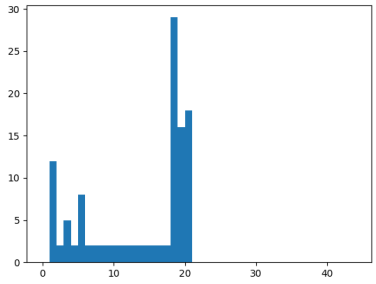
When using HDBSCAN with the training MNIST dataset, we get clustering plots in fig 6(a). We can see how there are multiple clusters, shown in colors, while the outliers are gray. Additionally, the test set points are in black and are all outside clusters. To actually see this, we plotted histograms to visualize the distributions of labels across the different clusters, as seen in fig 8. We chose one hard cluster with a label and also the generic cluster of outliers. We notice that the cluster labels as 3 has only images labeled as 8 while the cluster of outliers during the training has a distribution of labels, indicating that the algorithm only chooses the most representative images in the clusters while leaving out those not as representative (more on this later). We can also see the predicted outlier cluster for the test set, when run on the centroids of the train set, is exclusively for images of label 9. In fact, none of the other clusters had an image of label



(a) Toy Training Cluster 1

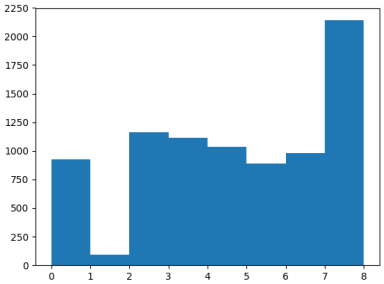


(b) Toy Testing Outliers - Novelty Detected

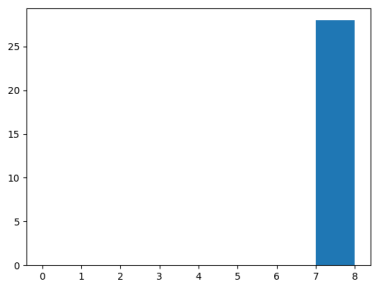


(c) Toy Testing Cluster 1 - Not Novel

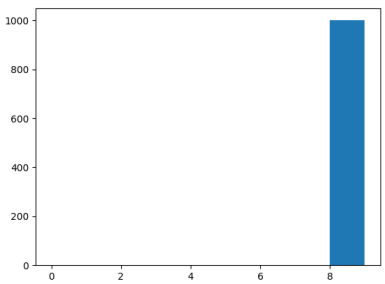
Fig. 7: Toy Cluster Labels



(a) RMNIST Training Outliers



(b) RMNIST Training Cluster 3 - Corresponds to Label 8



(c) RMNIST Testing Outliers - Corresponds to Label 9

Fig. 8: RMNIST Cluster Labels

9. We can already get some confirmation of the robustness of this algorithm.

When we ran this on the Simulated toy dataset, our results can be seen in fig 7. Our testing set outlier cluster (images with label $l = -1$) were dominated by actions needing the agent to be move to location > 40 or to location 18-20, as seen in fig 10(a) and fig 10(b). These were reasonable because these actions corresponded to the presence of the the pink triangles, which were novel objects. Some of the frames were put into the clusters defined during training, and we wanted to see whether those were valid. As seen in fig. 10(c) and 10(d), the frames have very similar structure and paths, making it perfectly reasonable to be clustered together.

All these experiments were performed with PCA features for 30 principal components. We got similar results for 50 components but not as good for below 20. The minimum cluster size hardly affected the type and nature of clusters, showing that the method was robust to noise and hyper-parameters. What was interesting was the type of distance metric did make a difference. Our results use L2 distance everywhere. On using L1, cosine or Chebyshev distance, results varied significantly and often much worse for our case.

We wanted to do a comparison with K-means and hence ran K-means for a variety of distance metrics and k, both being hyper-parameters. The clustering on MNIST we got was best with PCA using 2 components and 10 clusters, as seen in fig 9. On running on RMNIST with 9 clusters and then doing adding a test point force the test image (label 9) to be in either cluster with image label 1 or image label 7, thus not giving us what we needed. We thought of defining a radial distance around the each centroid beyond which points had to be classified as an outlier, but this would require much change from our current implementation and HDBSCAN already takes care of it for us. SIFT features were again not as useful for clustering in this case.

C. Safety Function

Having got the clusters, we wanted to see the safety action that we predicted given our formulation. In the toy dataset, the safety action $a_i = g(S_i)$ for each cluster S_i was easy to get, and it happened to be just the average of x coordinates of the dominant frame in that cluster. We defined our absolute safety function a_s to be completely stopping to velocity $v = 0$.

For the simulated toy dataset, in the case of the test image clustered in cluster with label $l = 1$, our probabilities to each

cluster, in order of their label, were:

$$p = \{5.45 \times 10^{-5}, 8.74 \times 10^{-1}, 5.29 \times 10^{-5}, \\ 5.82 \times 10^{-5}, 5.97 \times 10^{-5}, 4.99 \times 10^{-6}, \\ 7.13 \times 10^{-6}, 8.04 \times 10^{-6}, 7.27 \times 10^{-6}, \\ 7.26 \times 10^{-6}, 7.52 \times 10^{-6}, 7.11 \times 10^{-6}\}$$

Thus we see that the probabilities predominately indicate that the test image should go take the safety action as denoted by C_1 i.e. $x = 13$. This location was indeed a safe action for the agent to take, when we crossed checked this with our dataset. Additionally, since this test frame was not an outlier, we did not use the absolute safety metric a_s.

We apply our framework to an image which was classified as an outlier. Our probabilities to each cluster are:

$$p = \{3.43 \times 10^{-2}, 3.67 \times 10^{-2}, 5 \times 10^{-2}, \\ 5.10 \times 10^{-2}, 2.3 \times 10^{-2}, 2.58 \times 10^{-2}, \\ 2.47 \times 10^{-2}, 2.54 \times 10^{-2}, 2.6 \times 10^{-2}, \\ 2.58 \times 10^{-2}, 2.37 \times 10^{-2}, 5.63 \times 10^{-2}\}$$

We see that our probabilties to each cluster are really small, but our percentage deviation of the D_b' of this image from D_b^a is almost 1. Thus, our absolute safety function dominates for the a_s' equation, leading our agent to stop completely, as is safe in a completely unstructured environment or large x coordinate location and a new pink triangle object.

VII. Limitations

Although we believe this proposed method to be more robust than previous work, there are several key limitations to the formulation. First, depending on how diverse an agent's environment is, we cannot guarantee that the clustering method will produce clusters that are dense enough such that a safety action is associated with each cluster. Second, because we produce a dynamic safety function, we are not able to guarantee that the agent can always end up in a safe state, as it would if it uses a static safety prior. Specifically, having some theoretical properties and insights into such a method would be useful. Furthermore, because novelty detection is prone to changes in environment, like rain or snow on a road, more work would have to be done to ensure that the novelty detection aspect of this work still identifies fundamental obstacles in volatile environments. As an extension of this, because the system was not tested in a non-static environment, we have little guarantee as to how the system will classify and react to moving objects.

VIII. Future Work

First, we would like to analyze how our method works on the Fetch dataset using similiar metrics as those analyzed in the Simulation dataset. For future work, we hope to further analyze the benefits and disadvantages that the different features used (PCA, SIFT etc), different distance metrics in th clustering algorithms, and different clustering algorithms themselves. Furthermore, we hope to analyze whether a dynamic safety function such as the one proposed can provide a theoretical safety guarantee for robotic systems. We hope to analyze how this system can work in more complex environments and experiments, and whether novelty detection can be used to find and adjust to volatility in environments. Additionally, we would like to examine how to formulate this problem for non-static environments and how to test the proposed approach in non-static environments. Finally, we would like to further examine what edge cases would lead non-novel objects to be classified as novel, and whether they would pose a substantial risk to the reliability of the method.

IX. Acknowledgements

We would like to thank Prof. Dorsa Sadigh for all her help, guidance and resources (including Fetch) that she provided. We would also like to thank the students in the CS333 class for providing us feedback on our ideas, presentations and thoughts which helped us improve them to get them to a more clear and useful state.

References

[1] C. Richter and N. Roy, "Safe visual navigation via deep learning and novelty detection."

[2] S. Na, L. Xumin, and G. Yong, "Research on k-means clustering algorithm: An improved k-means clustering algorithm," in *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, April 2010, pp. 63–67.

[3] R. Butler, "Outlier discordancy test in the normal linear model," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 45, no. 1, pp. 120–132, 1983. [Online]. Available: http://www.jstor.org/stable/2345631

[4] M. A. F. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "Review: A review of novelty detection," *Signal Process.*, vol. 99, pp. 215–249, June 2014. [Online]. Available: http://dx.doi.org/10.1016/j.sigpro.2013.12.026

[5] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *CoRR*, vol. abs/1312.6199, 2013. [Online]. Available: http://arxiv.org/abs/1312.6199

[6] F. Kahl, R. I. Hartley, and V. Hilsenstein, "Novelty detection in image sequences with dynamic background." Springer.

[7] H. Gonzalez-Banos and J.-C. Latombe, *Robot Navigation for Automatic Model Construction using Safe Regions*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 405–415.

[8] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *arXiv preprint arXiv:1705.01292*, 2017.

[9] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, "Reachability-based safe learning with gaussian processes," in *53rd IEEE Conference on Decision and Control*, Dec 2014, pp. 1424–1431.

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

[11] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2013, pp. 160–172.