Gaussian Mixture Models for Anomaly Detection

Karan Gandhi
IIT Gandhinagar
Gandhinagar, Gujarat, India
23110157@iitgn.ac.in

Anurag Signh
IIT Gandhinagar
Gandhinagar, Gujarat, India
23110035@iitgn.ac.in

Arjun Dikshit
IIT Gandhinagar
Gandhinagar, Gujarat, India
23110040@iitgn.ac.in

Abhinav Khot
IIT Gandhinagar
Gandhinagar, Gujarat, India
23110006@iitgn.ac.in

Abstract

Unsupervised anomaly detection on multi- or high-dimensional data is of great importance in both fundamental machine learning research and industrial applications, for which density estimation lies at the core. For instance, in fraud detection, identifying irregularities in high-dimensional transaction data can prevent financial losses. In healthcare, detecting rare patterns in patient records or medical imaging can lead to early diagnosis of critical conditions. These applications rely heavily on accurate density estimation to distinguish normal behavior from potential anomalies in complex data distributions. Gaussian Mixture Models (GMMs) are a probabilistic approach used effectively for this task. They model data as a mixture of several Gaussian distributions, capturing complex data distributions and enabling the identification of outliers. In our experiments detailed in this report, we modify the benchmark model Deep Autoencoding Gaussian Mixture Model(DAGMM) [1] and perform various experiments to help better detect anomalies.

Keywords

Gaussian Mixture mistakes, Anomaly detection, Sample energy, Reconstruction loss, Autoencoders

1 Introduction

Unsupervised anomaly detection is a key challenge in machine learning with important applications in fields like cybersecurity, system monitoring, and healthcare. At its core, it involves identifying data points that fall in low-density regions of the data distribution.

While significant progress has been made, detecting anomalies in high-dimensional data without supervision remains difficult. As dimensionality increases, accurate density estimation becomes harder because every data point might appear rare. To address this, many methods follow a two-step process: first reducing dimensionality, then estimating density in the lower-dimensional space. However, this can result in suboptimal outcomes, as the dimensionality reduction step may discard information that may be information for anomaly detection.

Some methods focused on training an autoencoder to compress data points to a lower-dimensional representation and classifying points which have high reconstruction error as anomalies. Reconstruction-based methods face performance limitations because they assess anomalies from only one perspective—reconstruction error. While anomalous samples often show distinct compression patterns compared to normal ones and may exhibit high reconstruction errors, many anomalies can still remain hidden with seemingly normal error levels. This typically occurs when the dimensionality reduction techniques used have high model complexity or when the anomalous samples are noisy and structurally complex.

A more effective strategy involves integrating dimensionality reduction and density estimation into a single unified model. While this approach is computationally intensive, deep learning methods have begun to explore it by leveraging the powerful representation capabilities of neural networks. However, their performance is often constrained—either because the reduced low-dimensional space fails to retain crucial information from the original data, the density estimation model lacks sufficient complexity, or the training process is not well-suited for the goals of density estimation.

DAGMM is a model that mixes and takes the best attributes out of all the approaches. In our experiments, we modify the DAGMM architecture to increase accuracy in detection of anomalous points in the dataset. As a broad overview, we have conducted five main experiments

(1) **Modifying the loss function:** The DAGMM model uses only non anomalous points for tarining. We modify the training process of DAGMM to include anomalous points as well and we introduce a loss term tot he original loss expression that encourages the learned mixture models to have lesser density in the regions where the anomalous points are present.
(2) **Modifying the Inference Criteria:** During Inference, we make incremental modifications the strategy used by DAGMM to classify points as anomalous.
(3) **Modifications in Mixture models:** We use Laplacian mixture models for the problem instead of Gaussian Mixture models
(4) **Modifications to the Model Architecture:** We replace a part of the DAGMM architecture with Variational Autoencoder (VAE's) and Vector-Quantized Variational Autoencoder (VQ-VAE).
(5) **Estimating the correct value of threshold, and using unlabeled data:** We modify the training setup to use very few labeled data and a lot of unlabeled data and estimate the threshold based on that.

The exact details of the modifications are explained in section 3.

2 Related Work

A huge leap in advancement for detection of anomalies in a dataset were made by the authors of DAGMM. Since our experiements mainly involve modifying different parts of the DAGMM architecture, we explain their architecture in brief. Interested readers are encouraged to read the entire paper linked in the references.

DAGMM makes use of the best parts of both the paradigms used above. They propose an end-to-end trainable model. The model mainly consists of two networks, the Compression Network and the Estimation Network.

As mentioned before, identifying an anomaly in high-dimension space is hard as the probability of any of the point occurring is very low due to the curse of dimensionality. To mitigate this issue, DAGMM uses the Compression Network, which takes in an input vector x and outputs a lower dimension z, using a simple autoencoder. A decoder is used to reconstruct x from z_c. Let us call this reconstruction x'. This reconstruction is used to calculate the reconstruction loss for training the autoencoder.

Another vector z_r i calculated using the reconstruction error. In particular, z_r can be multi-dimensional, considering multiple distance metrics such as absolute Euclidean distance, relative Euclidean distance, cosine similarity, and so on. We use cosine similarity and euclidean distance in our implementation. In the end, the compression network concatenates both z_c and z_r and feeds the concatenated vector z to the subsequent estimation network.

The Estimation Network takes the low-dimensional representation z from the Compression Network and performs density estimation using a Gaussian Mixture Model (GMM). Rather than relying on traditional EM algorithms, DAGMM employs a neural network to predict soft assignments $\hat{\gamma}$ to the GMM components using a softmax over a learnable mapping. These soft assignments are then used to compute the mixture component parameters: mixture probabilities $\hat{\phi}_k$, means $\hat{\mu}_k$, and covariances $\hat{\Sigma}_k$ for each component k, as follows:

$$\hat{\phi}_k = \sum_{i=1}^{N} \frac{\hat{\gamma}_{ik}}{N}, \quad \hat{\mu}_k = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik}\mathbf{z}_i}{\sum_{i=1}^{N} \hat{\gamma}_{ik}}, \quad \hat{\Sigma}_k = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik}(\mathbf{z}_i - \hat{\mu}_k)(\mathbf{z}_i - \hat{\mu}_k)^T}{\sum_{i=1}^{N} \hat{\gamma}_{ik}}$$

where

$$\mathbf{p} = MLP(\mathbf{z}; \theta_m), \quad \hat{\gamma} = \text{softmax}(\mathbf{p}),$$

(θ_m is the model parameters for the Estimation network)

These parameters are then used to compute a sample's energy $E(z)$, which measures how well it fits the learned distribution. A high energy indicates an anomaly.

To train the overall DAGMM model, an objective function is used that balances three components: reconstruction loss, sample energy, and a regularization penalty on the covariance matrices to avoid degenerate GMM components:

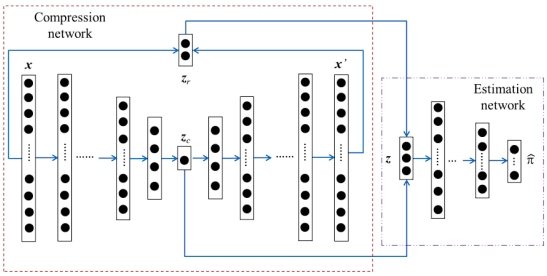

Figure 1: Overview of the DAGMM architecture

Loss Function:

$$J(\theta_e, \theta_d, \theta_m) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L(\mathbf{x}_i, \mathbf{x}'_i)}_{\text{reconstruction loss}} + \underbrace{\frac{\lambda_1}{N}\sum_{i=1}^{N} E(\mathbf{z}_i)}_{\text{sample energy}} + \underbrace{\lambda_2 P(\hat{\Sigma})}_{\text{penalize degeneracy of covariance}}.$$

This joint objective ensures that both the representation learning and density modeling are optimized together. The penalty term $P(\hat{\Sigma})$ is critical, as it helps avoid trivial solutions where covariance matrices become singular, thereby stabilizing GMM training.

During Inference, the points with sample energy in the top x percentile are classified to be anomalous.

It is important to note that during DAGMM training, only normal points are used. The intuition is that we fit our Gaussians on normal correct data, without considering the positions of the anomalous points. This ensures that the Gaussians represent the probability distribution of the actual correct data, without being skewed by adding anomalous points to skew the distribution. Infact, when DAGMM was processed in a dataset with only 5% anomalous points out of all anomalous points, we see a sharp decrease in accuracy. Training on only normal points tries to ensure that anomalies have very high reconstruction loss and in turn have a very high sample energy, which would aid us in seperating them from normal data.

3 Methodology

In this section we analyze our experiment methodologies in detail, as well as the leading intuition behind them.

(1) **Modifying the loss function:** The current DAGMM model only uses non anomalous data points for training. The objective function of DAGMM incentivizes the sample energy of non anomalous points to be low such that during inference, anomalous points are easily identifiable as they have high sample energies. However, we are not really taking into consideration all of the data present. Also there is no guarantee that reducing the sample energy for normal points necessarily means that the sample energy of the anomalous points during inference will be high. To mitigate this issue, we propose a slight modifaction to the loss function and the training strategy and shift to a more supervised model. We will train DAGMM on all points (anomalous and

non anomalous). However, we will incentivize the model to give higher sample energy to anomalous points as well as decrease sample energy of normal points by changing the objective function just for anomalies. We use the following objective function:

$$J_1(\theta_e,\theta_d,\theta_m) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L(\mathbf{x}_i,\mathbf{x}_i')}_{\text{reconstruction loss}} + \underbrace{\frac{\lambda_1}{N}\sum_{i=1}^{N}\left(\underbrace{E(\mathbf{z}_i)}_{\text{normal data}} \quad OR \quad \underbrace{\frac{1}{E(\mathbf{z}_i)}}_{\text{anomalous data}}\right)}_{\text{sample energy}}$$

$$+ \quad \underbrace{\lambda_2 P(\hat{\Sigma})}_{\text{penalise degeneracy of COV}} \quad .$$

(2) **Modifying the Inference criteria:** in the DAGMM architecture, only sample energy is being used during inference. In inference, it could be the case the during compression, anomalous data may get compressed to look similar to normal data and hence get lower sample energy. So we include even reconstruction loss during inference.

(3) **Modification in mixture models:** We replace Gaussian mixture model update equations with the Laplacian mixture model(LMM) update equations to try fit a LMM on the dataset for anomaly detection.
A Laplacian Mixture Model (LMM) assumes each data point x_i comes from one of K Laplace components:

$$p(x_i) \;=\; \sum_{k=1}^{K} \pi_k \,\text{Laplace}(x_i \mid \mu_k, b_k),$$

where

$$\text{Laplace}(x \mid \mu, b) = \frac{1}{2b}\exp\Big(-\frac{|x-\mu|}{b}\Big), \quad \sum_{k=1}^{K}\pi_k = 1.$$

Parameter update for LMMs:

$$\mathbf{p} = MLP(\mathbf{z};\theta_m), \quad \gamma = \text{softmax}(\mathbf{p})$$

$$\pi_k \leftarrow \frac{1}{N}\sum_{i=1}^{N}\gamma_{ik}, \quad \mu_k \leftarrow \frac{\sum_i \gamma_{ik}\, x_i}{\sum_i \gamma_{ik}}, \quad b_k \leftarrow \frac{\sum_i \gamma_{ik}\,|x_i-\mu_k|}{\sum_i \gamma_{ik}}.$$

(4) **Modifications in DAGMM Architecture:** In the current DAGMM, we are using autoencoders for encoding the data to lower dimensional representations. So, we replaced the autoencoder with Variational Autoencoder (VAE) and Vector Quantized Variational Autoencoder (VQ-VAE).
VAE are a type of autoencoder that learns a probability distribution over the data. Instead of mapping each input to a single point, VAEs map it to a range of possible values (mean and variance). This helps in learning more relevant lower dimensional representation.

DAGMM–VAE Loss:

$$\mathcal{L}_{\text{VAE}} = \frac{1}{N}\sum_{i=1}^{N}\Big[-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)}\big[\log p_{\theta_d}(\mathbf{x}_i \mid \mathbf{z})\big] + KL\big(q_\phi(\mathbf{z}\mid\mathbf{x}_i)\,\|\,p(\mathbf{z})\big)\Big]$$

$$J(\phi,\theta_d,\theta_m) = \underbrace{\mathcal{L}_{\text{VAE}}}_{\substack{\text{recon.+}\\ \text{KL reg.}}} + \underbrace{\frac{\lambda_1}{N}\sum_{i=1}^{N}E(\mathbf{z}_i)}_{\text{sample energy}} + \underbrace{\lambda_2\, P(\{\Sigma_k\})}_{\text{covariance penalty}}$$

VQ-VAE use a fixed set of learned discrete codes (like a dictionary) to represent inputs. Each input is mapped to the closest code in this dictionary, making the representation compact and discrete. By using VQ-VAE, the latent $\mathbf{z}_i = e_{k_i}$ takes only K possible values, so the GMM fits a mixture over discrete clusters. Outliers—which map to rarely used or poorly reconstructed codes—produce high sample energy and stand out better.

DAGMM–VQ-VAE Loss:

$$\mathcal{L}_{\text{VQ}} = \frac{1}{N}\sum_{i=1}^{N}\Big[\underbrace{\|\mathbf{x}_i-\mathbf{x}_i'\|^2}_{\text{reconstruction}} + \underbrace{\|\,\text{sg}[z_e(\mathbf{x}_i)]-e_{k_i}\|^2}_{\text{codebook update}} + \underbrace{\beta\|\,z_e(\mathbf{x}_i)-\text{sg}[e_{k_i}]\|^2}_{\text{commitment}}\Big]$$

$$J = \mathcal{L}_{\text{VQ}} + \underbrace{\frac{\lambda_1}{N}\sum_{i=1}^{N}E(e_{k_i})}_{\text{sample energy}} + \underbrace{\lambda_2\, P(\{\Sigma_k\})}_{\text{covariance penalty}}$$

(5) **Estimating the correct value of threshold, and using unlabeled data:** In the original setup, the threshold was treated as a hyperparameter that could be manually specified. Additionally, only positive samples were utilised during training. However, in real-world applications—particularly within medical datasets—the available positive samples are often limited and manually curated. Training a model exclusively on such a small dataset increases the risk of overfitting. Therefore, incorporating additional unlabeled data into the training process may be advantageous for improving the model's generalisation capability.
So now we try to modify our setup into the following setup: Given a partially labelled dataset containing only positive. More formally: Our dataset is $\{(x^{(i)}, t^{(i)}, y^{(i)})\}_{i=1}^{m}$, where $t^{(i)} \in \{0,1\}$ is the "true" label, and where

$$y^{(i)} = \begin{cases} 1 & x^{(i)} \text{ is labeled} \\ 0 & \text{otherwise.} \end{cases}$$

All labeled examples are positive, which is to say $p(t^{(i)} = 1 \mid y^{(i)} = 1) = 1$, but unlabeled examples may be positive or negative. Our task is to correctly predict the true labels of the datapoints.
It can be shown that if the model is trained on the entire dataset, the following relationship holds under the assumption that the number of anomalous points is negligible:

$$p(t^{(i)} = 1 \mid x^{(i)}) = \frac{p(y^{(i)} = 1 \mid x^{(i)})}{p(y = 1 \mid t = 1)}.$$

Additionally, it can be demonstrated that:

$$p(y = 1 \mid t = 1) = \mathbb{E}_{v\in V^+}[h(v)],$$

where $h(v)$ denotes the predicted logits, and V^+ represents the subset of positively labeled data points. Consequently, the threshold for predicting probabilities can be set to

$$\frac{\mathbb{E}_{v \in V^+}[h(v)]}{2}.$$

And we can use this threshold to classify the points as anomalous or non-anomalous.

4 Results

All experiments were performed on the KDDCUP dataset. DAGMM reports its accuracies on the 20 percentile threshold and for comparing our models with DAGMM, we will use the same threshold. We have included the other threshold results just for completeness. All models are trained for 200 epochs.

(1) **Modifying the loss function:** The following are the results for modifying the loss function

Percentile Threshold	Precision	Recall	F-score
10	0.5820	0.2977	0.3939
20	0.5072	0.5171	0.5121
30	0.3694	0.5641	0.4465
40	0.4207	0.8556	0.5641
50	0.3806	0.9573	0.5446
60	0.3807	0.9576	0.5448

Table 1: Peformance metrics on modifying the loss function ROC AUC score: 81.80.

Overall, we see a large decrease in the accuracy in comparison with the DAGMM model. We infer that adding anomalous points during training confuses the model, reducing its precision and recall drastically.

(2) **Modifying the Inference Criteria:** The following are the results for modifying the inference criteria:

Percentile Threshold	Precision	Recall	F-score
10	0.9464	0.4557	0.6152
20	**0.9505**	**0.9200**	**0.9350**
30	0.7919	0.9995	0.8837
40	0.6590	0.9996	0.7943
50	0.5952	0.9999	0.7462
60	0.5952	0.9999	0.7462

Table 2: Performance metrics using reconstruction loss + sample energy as inference criterion (ROC AUC score: 98.62).

We see a 2% increase in Precision score, and maintaing a similar F-score to the DAGMM model. However, We observe a 2% decrease in the Recall in comparison to DAGMM.

(3) **Modifications in Mixture models:** The results in Table 3 are the results we get using Laplacian mixture models instead of Gaussian Mixture models for anomaly detection. We consistently get worser results than DAGMM more than 10 % decrease in precision and 20% reduction in recall. This is representative of the distribution of the underlying data.

Percentile Threshold	Precision	Recall	F-score
10	0.9786	0.4868	0.6502
20	0.8532	0.7554	0.8014
30	0.6700	0.7680	0.7156
40	0.6183	0.9086	0.7358
50	0.5653	0.9326	0.7039
60	0.5217	0.9327	0.6691

Table 3: Performance metrics for LMM (constant ROC AUC = 88.91%).

(4) **Modifications to the Model Architecture:** The results for using the VAE are give in Table 4

Percentile Threshold	Precision	Recall	F-score
10	0.9686	0.4776	0.6398
20	**0.9642**	**0.9461**	**0.9551**
30	0.7767	0.9684	0.8620
40	0.6526	0.9846	0.7849
50	0.5616	0.9916	0.7171
60	0.4930	0.9975	0.6599

Table 4: Performance metrics while using VAE (ROC AUC score: 98.28).

We notice a significant increase in Precision in comparison to DAGMM with a 4% increase. We also see 2% increase in F-Score while recall remains almost the same. This marks a good improvement from DAGMM
The results for using the VQ-VAE are give in Table 5

Percentile Threshold	Precision	Recall	F-score
10	0.9720	0.4804	0.6430
20	**0.9663**	**0.9499**	**0.9581**
30	0.7869	0.9887	0.8763
40	0.6552	0.9893	0.7883
50	0.5759	1.0000	0.7309
60	0.5336	1.0000	0.6959

Table 5: Performance metrics while using VQ-VAE (ROC AUC score: 98.93).

We again notice a 4% increase in Precision in comparison to DAGMM. We also see 2% increase in F-Score while recall sees a 0.5% increase. This marks a good improvement from DAGMM

(5) **Estimating the correct value of threshold, and using unlabeled data** This method achieves an F-score of 0.8434, with a precision of 0.7411 and a recall of 0.9785. It is important to note that this performance is obtained under a modified training setup, where only a very limited number of positive samples are known. Despite this constraint, the method maintains a high F-score, indicating its robustness.

Method	Precision	Recall	F-score
DAGMM (Baseline)	0.9297	0.9442	0.9369
DAGMM with VAE	**0.9642**	**0.9461**	**0.9551**
DAGMM with VQVAE	**0.9663**	**0.9499**	**0.9581**
DAGMM with modified inference	**0.9505**	0.9200	0.9350
DAGMM with Laplacian	0.8532	0.7554	0.8014
DAGMM with modified Loss	0.4207	0.8556	0.5641
DAGMM with modified training setup*	0.7411	**0.9785**	0.8434

Table 6: Performance metrics using across all modifications

5 Conclusion

Our extensive experimentation with various modifications to the Deep Autoencoding Gaussian Mixture Model (DAGMM) framework demonstrates several significant findings in the field of unsupervised anomaly detection. The baseline DAGMM model already performs well with precision of 0.9297 and recall of 0.9442, achieving an F-score of 0.9369. However, our proposed modifications yield notable improvements.

The integration of Variational Autoencoders (VAE) into the DAGMM architecture produces the most substantial performance gain, with an F-score of 0.9551 and improved precision (0.9642) while maintaining high recall (0.9461). The Vector Quantized VAE variant (VQVAE) further enhances precision to 0.9663 with competitive recall at 0.9499, resulting in an F-score of 0.9581.

Our modified inference criterion shows particular promise at lower threshold values, increasing precision by 2% while maintaining comparable F-scores to the baseline. However, the Laplacian mixture model substitution performed worse than the Gaussian approach, indicating that Gaussian distributions better represent the underlying data structure for anomaly detection tasks.

The modified training setup incorporating both anomalous and normal data points yields strong results with precision of 0.7411 and recall of 0.9785, which is impressive considering the limited availability of labeled anomaly data in real-world scenarios. This semi-supervised approach offers a practical advantage for applications where some anomaly examples are available but scarce.

Overall, our modifications demonstrate that architectural improvements to autoencoder components and objective functions can significantly enhance anomaly detection performance. Future work should focus on further exploring the integration of mixture of multiple methods tested above and extending these approaches to other domains beyond the KDDCUP dataset tested in this research.

References

[1] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. 2018. Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection. In *International Conference on Learning Representations*. https://openreview.net/forum?id=BJJLHbb0-