Anomaly Detection Based on Disentangled Representation Learning

by

Xiaoyan Li

Thesis submitted to the University of Ottawa
In partial Fulfillment of the requirements for the
M.A.Sc.

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

In the era of Internet of Things (IoT) and big data, collecting, processing and analyzing enormous data faces unprecedented challenges even when being stored in preprocessed form. Anomaly detection, statistically viewed as identifying outliers having low probabilities from the modelling of data distribution $p(\boldsymbol{x})$, becomes more crucial. In this Master thesis, two (supervised and unsupervised) novel deep anomaly detection frameworks are presented which can achieve state-of-art performance on a range of datasets.

Capsule net is an advanced artificial neural network, being able to encode intrinsic spatial relationship between parts and a whole. This property allows it to work as both a classifier and a deep autoencoder. Taking this advantage of CapsNet, a new anomaly detection technique named **AnoCapsNet** is proposed and three normality score functions are designed: prediction-probability-based (PP-based) normality score function, reconstruction-error-based (RE-based) normality score function, and a normality score function that combines prediction-probability-based and reconstruction-error-based together (named as PP+RE-based normality score function) for evaluating the "outlierness" of unseen images. The results on four datasets demonstrate that the PP-based method performs consistently well, while the RE-based approach is relatively sensitive to the similarity between labeled and unlabeled images. The PP+RE-based approach effectively takes advantages of both methods and achieves state-of-the-art results.

In many situations, neither the domain of anomalous samples can be fully understood, nor the domain of the normal samples is straightforward. Thus deep generative models are more suitable than supervised methods in such cases. As a variant of variational autoencoder (VAE), β-VAE is designed for automated discovery of interpretable factorised latent representations from raw image data in a completely unsupervised manner. The t-Distributed Stochastic Neighbor Embedding (t-SNE), an unsupervised non-linear technique primarily used for data exploration and visualizing high-dimensional data, has advantages at creating a single map that reveals local and important global structure at many different scales. Taking advantages of both disentangled representation learning (using β-VAE as an implementation) and low-dimensional neighbor embedding (using t-SNE as an implementation), another novel anomaly detection approach named **AnoDM** (stands for **Ano**maly detection based on unsupervised **D**isentangled representation learning and **M**anifold learning) is presented. A new anomaly score function is defined by combining (1) β-VAE's reconstruction error, and (2) latent representations' distances in the t-SNE space. This is a general framework, thus any disentangled representation learning and low-dimensional embedding techniques can be applied. AnoDM is evaluated on both image and time-series data and achieves better results than models that use just one of the two measures and other existing advanced deep learning methods.

Acknowledgements

I would like to express my sincere gratitude to my supervisors Dr. Tet Yeap and Dr. Iluju Kiringa for their consistent supports and guidance all through my Master research. They offered me a great opportunity to conduct such an interesting research in their group. Their beneficial mentoring and constructive feedback play a key role to not only enhance my academic knowledge but also forge my personalities as a good researcher.

I would also like to extend my thinks to my colleagues and friends for their help and encouragement in both study and life. I really appreciate the happy study time we spent together. Their positive and upward spiritual strength motivate me at all times.

I want to give many thanks to my husband for his considerate care and help. Whenever I face challenges, he always give me encouragement and useful suggestions. Thank him for making time to pick up our children from school. Thank him for giving up his rest time to correct my academic writing.

I wish to give big thanks to my parents and my children for their sacrifice and understanding. Since went back to university, I haven't visited my parents. Every time having a video chat with them, I am really touched by their understanding and encouragement.

Dedication

This is dedicated to the ones I love.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ADGAN: anomaly detection with generative adversarial network, 38
AnoCapsNet: anomaly detection based on CapsNet, 3
AnoDM: anomaly detection based on unsupervised disentangled representation learning and manifold learning, 3
CNN: convolutional neural network, 4
CPSs: cyber-physical systems, 15
CapsNet: Capsule neural network, 2
DBN: deep belief net, 8
DCGANs: deep convolutional generative adversarial networks, 13
DGMs: deep generative models, 2
ELBO: evidence lower bound, 9
EM: expectation maximization, 21
FCN: fullly connected network, 26
FN: false negatives, 30
FP: false positives, 30
FPR: false positive rate, 30
GAN-AD: generative adversarial networks-based anomaly detection, 9
GANs: generative adversarial networks, 13
GRUs: gated recurrent units, 9
IoTs: internet of things, 1
KDE: kernel density estimation, 1
LSTM: long short-term memory, 4
MIG: mutual information gap, 73
MIM: mutual information machine, 73
NSE: normalized squared error, 28
OCSVM: one-class support vector machine, 7
PCA: principal component analysis, 25
PP: prediction probabilities, 28
RBMs: restricted Boltzmann machines, 9
RE: reconstruction error, 29
ROC: Receiver operating characteristics, 30
SNE: stochastic neighbor embedding, 23
SR-VAE: sequential residual variational autoencoder, 73
SVDD: support vector domain description, 1
SVM: support vector machine, 9

TCN: temporal convolutional network, 4
TCVAE: total correlation variational autoencoder, 73
TN: true negatives, 30
TP: true positives, 30
TPR: true positive rate, 30
UMAP: uniform manifold approximation and projection, 73
VAE: variational autoencoder, 2
auROC: area under receiver operating characteristics curve, 30
t-SNE: t-distributed stochastic neighbor embedding, 4

Chapter 1

Introduction

1.1 Motivations

As real-time tracking & diagnosis systems and autonomous controlling devices are strongly demanded in various domains in the current era of Internet of Things (IoTs) , smart cities, big data, and deep learning, anomaly detection (also known as *outlier detection*) is becoming increasingly critical. It aims at uncovering abnormal data points which significantly deviate from a decision threshold. The abnormal data might be translated into either negative alarming events or positive novel observations. For instance multiple failed login attempts indicating the possibility of cyber intrusion. A animal with rare color or size (such as white tiger) might indicating new (tiger) species. Particularly, anomaly detection is of key importance in IoT systems, data centres, security platforms, and life science to diagnose system failure, detect intruders or attackers, and discover novel knowledge.

Prior to the use of deep learning approaches, statistical and heuristic methods were main tools for anomaly detection in restricted application domains. Kernel methods, such as kernel density estimation (KDE) [64, 43] and support vector domain description (SVDD) [73], were the most successful ones to deal with non-linearity of the input feature space through the kernel trick. However, nowadays the tremendous amount of data of various types (such as images, time-series, texts, omics data, etc.) poses new challenges to them to handle the scalability and complexity of such data. With billions of samples in modern datasets, traditional methods become less effective. For example, conventional feature encoding and extraction methods are incompetent to capture informative factors from the input feature space of complex data.

1.2 Challenges

From the data analytics perspective, anomaly detection is a very challenging task due to the following reasons. (1) Many forms of data, e.g., images, text, and other types of sequences, are often highly unstructured and complex. How can these data be well represented and high-level information be extracted by an algorithm? (2) The sample sizes of modern data sets are often

extremely large and most of them are unlabelled. Unfortunately, traditional methods do not scale and perform well on these data. (3) When data of multiple modalities are naturally available for same events in a system, a robust and precise algorithm needs to be designed to integrate these information for system diagnosis or decision making. (4) Many intelligent systems, such as IoTs, require real-time detection and reaction of abnormal events to avoid costly and irrevocable damages. Thus, anomaly monitoring algorithms to be designed in these platforms must be highly efficient. In summary, anomaly detection raises challenges in representability, scalability, multimodality, and time complexity.

1.3 Objective

Embracing the wealth of data, deep learning models have achieved significant successes in various discriminative and generative modellings of modern data [6, 33, 46, 24, 44, 21, 48], encouraging to explore deep anomaly detection solutions. It offers great potentials to overcome the above challenges [48]. (1) Representation learning mechanisms (such as convolution, word embedding, and recurrence for time-series) have been developed in supervised and unsupervised deep models to consider the nature of specific types of input samples and encode them into vectors of continuous values as corresponding latent representations. (2) Most deep learning models are trained using stochastic gradient descent that splits a giant training set into mini-batches. Thus, learning becomes unrestricted and blessed by a large sample size. Particularly, stochastic variational inference [35, 80] has successfully enabled scalable learning and inference for deep generative models (DGMs) on a vast amount of unlablled data. (3) The development of deep learning programming packages, such as PyTorch [65] and TensorFlow [1], greatly eases the assembly of multiple network components (corresponding to different modalities) together for multimodal representation learning [55]. (4) Once a deep model is learned, the inference or encoding step is very efficient, thanks to the highly paralleled computing architectures and techniques.

Therefore, inspired by these developments, the objective of this thesis is to design two highly efficient and robust anomaly detection algorithms via employing representation learning mechanisms of deep learning models, in both supervised and unsupervised manners.

1.4 Methodology

Disentangled feature representation learning is one approach that allows deep learning models to capture the factorized latent features of a given input. By Learning a disentangled posterior distribution of the generative factors of the observed input, anomalous unseen input can be easily distinguished if its features locates outside of this posterior distribution. As CapsNet is able to automatically learn affine transformations of a raw input via an iterative routing-by-agreement mechanism [29, 69, 32], it can be employed as an autoencoder to capture the interpretable latent representation of the input's generative factors. β-VAE [28, 8, 59], a modification of variational autoencoder (VAE) , is famous for its effectiveness of automated discovery of disentangled latent

encoding from raw image data in a completely unsupervised manner. Therefore, based on these properties of both models, two novel anomaly detection frameworks: AnoCapsNet and AnoDM are introduced.

1.4.1 Anomaly Detection Based on CapsNet

In fact, learning transformation is a more difficult task than other kinds of feature learning. Convolutional neural network (CNN) [50], a hierarchy of convolution operations, has been widely used as a highly effective technique in classifying images. However, one arguable key limitation of CNN is that the neurons do not sufficiently capture the properties of entities such as position, orientation, and size, as well as their part-whole relationship. The capsule network (CapsNet) [29, 69, 32] has been proposed and shown advantages in learning such information, which is a novel and promising structure that may be more closely related to biological neural organization. A capsule is a group of neurons whose activity vector represents the instantiation parameters of a specific type of entity such as an object or part. It has been demonstrated that CapsNet is capable of preserving hierarchical pose (position, size, and orientation) relationships between image features. For a given image, CapsNet can automatically and dynamically model affine transformations and part-whole relationships using an iterative routing-by-agreement mechanism (please see Section 2.4 for details).

CapsNet is essentially a disentangled representation learning framework, where a capsule can encapsulate pose information (such as position, orientation, scaling, and skewness) and instantiation parameters (such as color and texture) of a local part. In order to surrogate activation probability, CapsNet introduces a novel activation function called squash function. Unlike softmax probabilities in CNN, the activation probabilities produced by this function do not necessarily sum to one. Thus, the activation probabilities of digital capsules at the last layer correspond to the probabilities of the unseen input belonging to the target classes. This inspires me to design a probabilities-prediction-based anomaly detection method using CapsNet (PP-based AnoCapsNet).

From the other perspective, CapsNet can be considered as a deep autoencoder, in which classifier functions as an encoder that automatically learn latent encoding (or representation) of a raw input data and reconstruction part works as decoder that recovers the input based on its latent representation. Therefore, reconstruction error which indicates the difference between original input and the reconstructed one, can be used to measure anomaly of unseen data, as anomalous data can not be well recovered by a CapsNet that is trained by using normal samples. The reconstruction-error-based (RE-based) anomaly detection using CapsNet (RE-based AnoCapsNet).

As PP-based AnoCapsNet and RE-based AnoCapsNet have complementary strengths, another normality score based method is introduced by taking full advantages of both methods and simply name it as PP+RE-based AnoCapsNet.

1.4.2 Anomaly Detection Based on Unsupervised Disentangled Representation Learning and Manifold Learning

As a variant of variational autoencoder (VAE) [44], β-VAE [28] is designed for unsupervised discovery of interpretable factorized latent representations from raw image data. An adjustable hyperparameter β is introduced to balance the extent of learning constraints (a limit on the capacity of the latent information channel and an emphasis on learning statistically independent latent factors) and reconstruction accuracy. It was demonstrated that β-VAE with appropriately tuned value of β (when $\beta > 1$) qualitatively outperforms VAE (when $\beta = 1$, β-VAE is exactly VAE). Burgess et al. [8] proposed a modification to the training regime of β-VAE by progressively increasing the information capacity of the latent code during training. This modification facilitates the robust learning of disentangled representations in β-VAE, without the previous trade-off in the reconstruction accuracy. Hoffman et al. [36] introduced a reformulation of β-VAE for $0 < \beta < 1$. They argued that, within in this range, training β-VAE is equivalent to optimizing an approximate log-marginal likelihood bound of VAE under an implicit prior.

Manifold learning is a family of nonlinear dimensionality reduction techniques. The t-distributed stochastic neighbor embedding (t-SNE) [15] is an unsupervised manifold learning method primarily used for data exploration and visualization by approximating high-dimensional data distribution using a two or three-dimensional map that could preserve local and certain global structures of the data. The use of t-SNE for anomaly detection has been sceptical [15]. However, no comprehensive investigation has been made in this topic.

Taking advantages of both disentangled representation learning (using β-VAE as an implementation) and low-dimensional manifold learning (using t-SNE as an implementation), a new anomaly detection approach is introduced and named **AnoDM**, standing for ***Ano****maly detection based on unsupervised* ***D****isentangled representation learning and* ***M****anifold learning* is presented. An anomaly score function in which is defined by combining: (1) β-VAE's reconstruction error, and (2) distances between latent representations of test points and training points in t-SNE map. AnoDM is a general framework, thus any disentangled representation learning and manifold learning techniques can be applied. The choice of a lower-level encoding scheme in β-VAE depends on data type of interest. For image data, deterministic convolutional network (CNN) is used in the encoder.

In case of time series (sequence) data, an improved version of β-VAE is designed by replacing CNN with temporal convolutional network (TCN) [5], a generic architecture for convolutional sequence prediction, in the encoder. TCN is incorporated as part of the encoder, because [5] have shown that TCN outperforms canonical recurrent networks such as LSTMs [34] across a range of supervised learning tasks and recommended that CNN should be regarded as the first method to try for sequence modeling tasks. Regarding the decoding architecture, CNN is simply chosen, because by choosing a simpler CNN architecture as a part of the decoder, the model can achieve a comparable even better performance but takes much less running time.

1.5 Contributions

1. X. Li, I. Kiringa, T. Yeap, X. Zhu, and Y. Li, "Exploring Deep Anomaly Detection Methods Based on Capsule Net," ," Long version was accepted as a long paper by 33th Canadian Conference on Artificial Intelligence, paper ID 71, May 2020. (ICML 2019 Workshop on Uncertainty & Robustness in Deep Learning, paper ID 17, Jun. 2019)

2. X. Li, I. Kiringa, T. Yeap, X. Zhu, and Y. Li, "Anomaly Detection Based on Unsupervised Disentangled Representation Learning in Combination with Manifold Learning," Long version was accepted by the International Joint Conference on Neural Networks (IJCNN), paper ID 20702, July 2020. (NeurIPS 2019 Workshop on Learning with Rich Experience (LIRE): Integration of Learning Paradigms, paper ID 37, Dec. 2019.)

3. In anomaly detection methods based on Capsule Net (named as AnoCapsNet), I make contributions as follows.

   (a) Based on unique characteristics of CapsNet, three high-efficiency normality score functions is proposed: prediction-probability-based normality score, reconstruction-error-based normality score and a normality score that combines them together;

   (b) AnoCapsNet is compared with principled benchmark methods (such as deep belief net (DBN), variational autoencoder (VAE) and one-class support vector machine (OCSVM)) and assess their capacities on a range of image datasets.

4. In anomaly detection method based on unsupervised disentangled representation learning in combination with manifold learning (named as AnoDM), I make the following contributions.

   (a) The capacity of an unsupervised disentangled representation learning for anomaly detection, which uses β-VAE as an implementation, is comprehensively explored.

   (b) The potential of t-SNE in combination with β-VAE for outlier identification is thoroughly investigated by taking the latent representation from β-VAE as input to t-SNE. To the best of my knowledge, this is the first attempt to explore t-SNE in anomaly detection.

   (c) For sequence anomaly detection, instead of using prevailed recurrent networks (such as LSTM), as a practical contribution, an improved convolution architecture (TCN) is adopted to capture the temporal dependency in the encoder in unsupervised way.

   (d) AnoDM is evaluated on a range of image data, including: MNIST, Fashion-MNIST, CIFAR-10 and Small-Norb, and also tested on ECG heart beat signals for evaluating its effectiveness on time-series data.

1.6 Thesis Outline

Chapter 2 starts with a comprehensive review of anomaly detection problem including both classical techniques and deep-learning based models, in terms of two categories: boundary-based

and distribution-based. Then it brings us to analyze one of the most popular classical anomaly detection model (one-class support vector machine). After that it particularly includes several existing advanced deep learning models and algorithms that are related and employed by the proposed frameworks.

The first part of Chapter 3 illustrates the detailed architecture and algorithm of the first novel anomaly detection framework (**AnoCapsNet**), which employs CapsNet as an autoencoder to learn disentangled representation of raw data, by taking advantages of CapsNet's unique property (automatically and dynamically model affine transformations and part-whole relationships). In the second part of this chapter, comprehensive experiments (including evaluation of anomaly score functions and comparison with other state-of-the-art anomaly detection methods) are presented. Then, the experiments' results are carefully analyzed.

Chapter 4 introduces the second new generic anomaly detection framework named as **AnoDM**, which the first time combines β-VAE (an unsupervised disentangled representation learning method) and t-SNE (low-dimensional manifold learning technique) for anomaly detection problem. In the first part, the architecture and algorithm of AnoDM are presented in details. In the second part, experiments such as investigating the strengths of AnoDM over other state-of-the-art anomaly detection methods and the evaluation of anomaly score function that is used to measure the anomality quality of data are thoroughly discussed.

Chapter 5 first draws the conclusion of this thesis. Then it focuses the discussion of both strengths and limitations of the two proposed frameworks. Finally, the future work is discussed for possible improvements of AnoCapsNet and AnoDM.

Chapter 2

Related Works

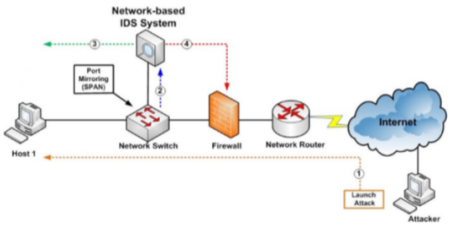
2.1 Anomaly Detection

Detecting anomalies in data flow of modern intelligent systems is an important but challenging problem. Formally speaking, anomaly detection problems can be statistically viewed as identifying outliers having low probabilities from the modelling of data distribution $p(\boldsymbol{x})$. Practically, since statistical modelling of the data is often difficult, it degenerates to domain description [73] or supervised prediction [23] problems in some cases. The exact explanation of an anomalous data point depends on the specific domain of focus. In data centers, it probably indicates an attempt of cyber intrusion (e.g. Figure 2.1a). In surveillance system, it could be an illegal traffic flow (e.g. Figure 2.1b). In biomedical information systems, it means possible onset of certain diseases (e.g. Figure 2.1c). In Internet of Things (IoT) systems (see Figure 2.1d), it may represent a hardware failure or alarming event captured by sensors. An anomalous sample is not always associated with negativity. Sometimes, it leads to novel discoveries in scientific explorations.

A classical viewpoint regarding anomaly detection is that, learning the boundary of data mass is more effective and straightforward than learning the density distribution of the data, because (1) available data were too few to cover the distribution in many cases, and (2) it was much more complicated and difficult to model data distribution using a generative model. However, in the big data era, massive amount of data become available in many domains; many of the data are structured (such as images, graphs, time-series, text, etc.); and deep generative models have achieved promising successes in modelling such modern data, offering a new avenue for exploring distribution-based methods for anomaly detection. These two categories, boundary-based and distribution-based, are respectively discussed below.

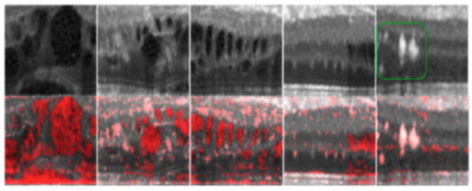
2.1.1 Boundary-Based Methods

Kernel based support vector domain description (SVDD) or one-class support vector machine (OCSVM) methods , including hypersphere [73] and hyperplane [71] models, is a successful family for anomaly detection in the pre-deep-learning era. The idea of the hypersphere-based SVDD is to map data points from input space to high-dimensional space and learn a hypersphere

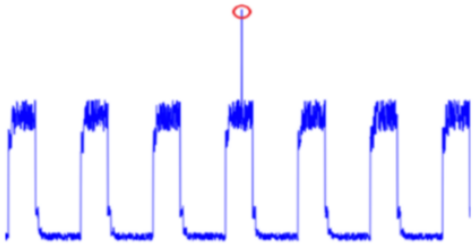


(a) Cyber-Network Intrusion detection, from [37]

(b) Illegal Traffic detection, from [79]



(c) Detecting Retinal Damage, from [70]



(d) Internet of Things (IoT) sensor data anomaly detection, from [62]

Figure 2.1: Applications of anomaly detection technique.

that capture the core mass of the data distribution. Any data point outside the hypersphere is viewed as an abnormal sample. Please see [53, 54] for a systematic discussion of SVDD methods. It is quite natural to think of deep extension of these methods to continue their success in the deep learning age. In pursuit of this aim, there are two efforts: deep hybrid methods and one-class neural network models. In deep hybrid methods (e.g. VAE+OCSVM [3] and DBN+OCSVM [16]), a supervised (e.g. CNN or recurrent net) or unsupervised (e.g. deep belief net (DBN) or varitiaonal autoencoder (VAE)) neural network is first employed to learn embedding representations of samples in the hidden space, then using these latent representations as inputs an SVDD method is used to detect abnormal data points. The one-class neural network models (e.g. one-class deep SVDD [68]) learn a neural network and SVDD together by maximising an adapted SVDD objective in the prime form. Both classes of methods have pros and cons. Deep hybrid methods are pipelines that are very flexible in choosing and combining different representation learning (or pretrained embeddings) and SVDD models. However, these methods face the challenge of scalability due to size of kernel matrices in dual form of SVDD. Deep SVDD models explicitly use deep neural networks as feature extractors in replacement of implicit kernel tricks, and are scalable to large data due to use of SVDD's prime forms and stochastic gradient descent. Nevertheless, this strategy is lack of flexibility in practice, that is a specific model needs to be built for each specific type of data.

2.1.2 Distribution-Based Methods

As mentioned above, deep generative models (DGMs), such as deep belief net (DBN) [30, 56] and variational autoencoder (VAE) [44], can be applied as unsupervised feature learning techniques. More importantly, since DGMs aim at modelling the joint distribution of visible and

latent variables (that is $p(\boldsymbol{x}, \boldsymbol{h})$), their likelihood $p(\boldsymbol{x})$ by marginalising out $\boldsymbol{h}$ may serve as an anomality describer. However, exact likelihood can only be obtained in quite few generative models, such as exponential family restricted Boltzmann machines (RBMs) [57]. In many cases when a recognition component is used for approximate inference, only the variational/evidence lower bound (ELBO) of log-likelihood is available. Unfortunately, ELBO may be too loose to be an normality indicator. Oftentimes, the ELBOs of normal and abnormal samples indistinguishably fall into the same range. Luckily, this is not the end of the story. In an architecture with encoder (recognition) component and decoder (generative) component, reconstruction error could serve as anomality measure based on the intuition that out-of-distribution samples can be reconstructed badly [2].

Furthermore, DGM-based algorithms are also devised to detect anomaly problem on sequence data (e.g. LSTM-VAE [63] and GAN-AD [52]). Conventionally, canonical recurrent networks (such as LSTM and GRUs [11]) are considered as the dedicated methods for sequence modeling. Some recent studies have also claimed that there was no architecture that could consistently beat LSTM in some typical sequence modeling tasks [38, 25, 61]. On the other hand, some other researchers insist that CNN [49] should be considered as more appropriate choice for sequences. Inspired by more recent CNN-based sequence modelling (such as machine translation [17, 18] and language modeling [12]), Bai et al. [5] conducted a systematic evaluation of generic convolutional and recurrent architectures for sequence modeling across a broad range of tasks that are commonly used to benchmark recurrent networks, and concluded that convolutional networks, rather than recurrent networks, should be respected as a "natural starting point for sequence modeling tasks".

2.2 One-class Support Vector Machine

2.2.1 Support Vector Machine

Support vector machine (SVM) was first introduced by Vapnik et al. [9, 7, 74, 75] which is primarily used for classification and regression tasks by constructing a hypersphere or hyperplane that divides training sample points into separate categories with as wide a gap as possible. It then makes predictions by assigning points to one side of the gap or the other. In Figure 2.2, the hyperplane $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ separates data points into two classes, with a margin defined by positive border $\boldsymbol{w}^T\boldsymbol{x} + b = +1$ and negative border $\boldsymbol{w}^T\boldsymbol{x} + b = -1$.

2.2.2 One-class Support Vector Machine

One-class Support Vector Machine (OCSVM) is a natural extension of SVM [71, 72]. It is to identify outliers or novelties given limited anomalous training examples. Instead of estimating the distribution of the data, OCSVM simply estimates the boundary of the distribution which captures the main mass of the data. For this reasion, OCSVM is also called *support vector domain description* (SVDD) [73]. SVDD finds the support (the set of support vectors lying on and outside a desired boundary) of a multivariate distribution. The border in the model can either be a hypersphere [73] or a hyperplane [71].

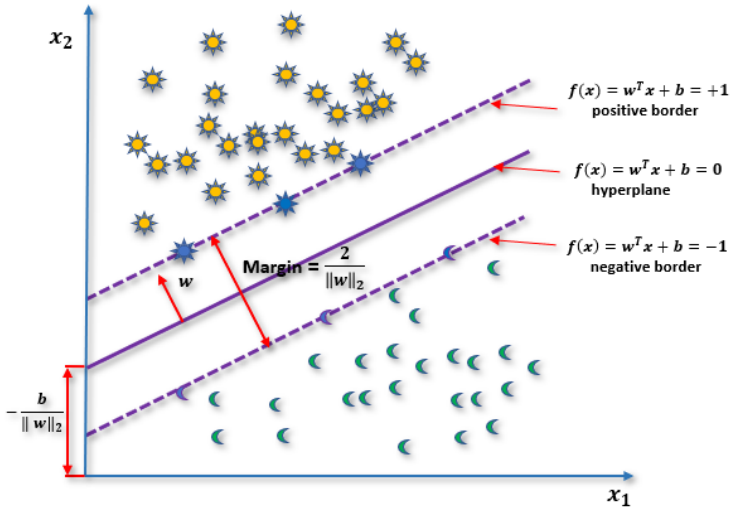


Figure 2.2: Illustration of support vector machine.

Hypersphere OCSVM

The intuition behind the hypersphere border is by implicitly mapping points to a higher-dimensional feature space its volume becomes as small as possible . A hypersphere is defined by its center and (squared) radius. An indicator function is learned such that the data points inside are positive (core data points), and these outside are negative (outlieres). This idea can be formulated, in following primal form [53].

$$\begin{aligned} &\min_{R,\xi,\upsilon} \boldsymbol{C}^T\boldsymbol{\xi} + \nu R && (2.1)\\ &s.t. \|\phi(\boldsymbol{x}_i - \boldsymbol{c})\|_2^2 \leq R + \xi_i, && i = 1, \cdots, n \\ &\boldsymbol{\xi} \geq 0 \\ &R > 0, \end{aligned}$$

where $\boldsymbol{c}$ is the center of the hypersphere, R is its squared radius, ξ_i is a slack variable representing error, $\nu \in (0, 1]$, and $\boldsymbol{C}$ is constant with $C_i = \frac{1}{n}$. The Lagrange function is

$$\begin{aligned} &L(R, \boldsymbol{\xi}, \boldsymbol{c}, \boldsymbol{\mu}, \boldsymbol{\eta}, \delta) \\ &= \boldsymbol{C}^T\boldsymbol{\xi} + \boldsymbol{\nu}R + \boldsymbol{k}^T\boldsymbol{\mu} - 2(\phi(\boldsymbol{X})\boldsymbol{\mu})^T\boldsymbol{c} + s\boldsymbol{c}^T\boldsymbol{c} - \boldsymbol{\mu}^T\boldsymbol{R} - \boldsymbol{\mu}^T\boldsymbol{\xi} - \boldsymbol{\eta}^T\boldsymbol{\xi} - \delta R, \end{aligned} \quad (2.2)$$

where k = diag$(\phi(\boldsymbol{X})^T\phi(\boldsymbol{X}))$, and $s = \mathbf{1}^T\boldsymbol{\mu}$. Then, the KKT conditions [53] are

$$\begin{cases} \nu - \mathbf{1}^T\boldsymbol{\mu} - \delta = 0 \\ \boldsymbol{C} - \boldsymbol{\mu} - \boldsymbol{\eta} = \mathbf{0} \\ -\boldsymbol{\phi}(\boldsymbol{X})\boldsymbol{\mu} + s\boldsymbol{c} = \mathbf{0} \\ \mu_i(\|\phi(\boldsymbol{x_i}) - \boldsymbol{c}\|_2^2 - r - \xi_i) = 0 \\ \boldsymbol{\eta} * \boldsymbol{\xi} = \mathbf{0} \\ \delta R = 0 \\ \boldsymbol{\mu} \geq 0 \\ \boldsymbol{\eta} \geq 0 \\ \delta \geq 0 \\ \|\phi(\boldsymbol{x_i}) - \boldsymbol{c}\|_2^2 - R - \xi_i \leq 0 \\ \xi_i \geq 0 \\ R > 0 \end{cases} \tag{2.3}$$

The dual form becomes

$$\begin{aligned} &\min_{\boldsymbol{\mu}} \frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{K}\boldsymbol{\mu} - \frac{\nu}{2}\boldsymbol{k}^T\boldsymbol{\mu} \\ &s.t. \mathbf{1}^T\boldsymbol{\mu} = \nu \\ &0 \leq \boldsymbol{\mu} \leq C, \end{aligned} \tag{2.4}$$

where $\boldsymbol{K} = \phi(\boldsymbol{X})^T\phi(\boldsymbol{X})$, an $\boldsymbol{k} = \text{diag}(\boldsymbol{K})$.

$S = \{i|\mu_i > 0, i = 1, \cdots, n\}$ is denoted as the set of indices of nonzero multipliers which correspond to points on or outside the border. From the KKT conditions, one know that the centroid of hypersphere is a sparse non-negative linear combination of the training data points, that is $\boldsymbol{c} = \frac{1}{s}\phi(\boldsymbol{X})\boldsymbol{\mu} = \frac{1}{\nu}\phi(\boldsymbol{X})_S\boldsymbol{\mu}_S$. $B = \{i|0 < \mu_i < C, i = 1, \cdots, n\}$ as the subset of indices of pointson th hypersphere. Then

$$R = \frac{1}{|B|}\sum_{b\in B}\|\phi(\boldsymbol{x})_b - \boldsymbol{c}\|_2^2 = \frac{1}{|B|}(\text{trace}(\boldsymbol{K}_B) - \frac{2}{\nu}\text{sum}(\phi(\boldsymbol{X})_B^T\phi(\boldsymbol{X})_S\boldsymbol{\mu}_S)) + \frac{1}{\nu^2}\boldsymbol{\mu}_S^T\boldsymbol{K}_S\boldsymbol{\mu}_S \tag{2.5}$$

The decision function is the following indicator function:

$$d(\boldsymbol{x}) = \text{sign}[f(\boldsymbol{x})] \tag{2.6}$$

where $f(\boldsymbol{x})$ is defined as below:

$$\begin{aligned} f(\boldsymbol{x}) &= R - \|\phi(\boldsymbol{x}) - \boldsymbol{c}\|_2^2 \\ &= R - (\phi(\boldsymbol{x})^T\phi(\boldsymbol{x}) - \frac{2}{\nu}\phi(\boldsymbol{x})^T\phi(\boldsymbol{X})_S\boldsymbol{\mu}_S + \frac{1}{\nu^2}\boldsymbol{\mu}_S^T\boldsymbol{K}_S\boldsymbol{\mu}_s). \end{aligned} \tag{2.7}$$

From the KKT conditions, the following important properties can be obtained [53].

1. If $\mu_i > 0$, the data point $\boldsymbol{x}_i$ resides either on or outside of the hypersphere.
2. If $0 < \mu_i < C$, then data point $\boldsymbol{x}_i$ is on the hypersphere. However, the reverse is not true.
3. If $\boldsymbol{x}_i$ resides outside of the hypersphere, then $\mu_i = C$. $\boldsymbol{x}_i$ is called an *outlier*. However, the reverse is not true.
4. As in the two-class ν-SVM, ν is a lower bound of the fraction of support vectors, and an upper bound of the fraction of outliers.

Hyperplane OCSVM

The hyperplane proposed by Schölkopf et al [71]. is defined as $f(\boldsymbol{x}) = \boldsymbol{w}^T\phi(\boldsymbol{x}) - b(b \geq 0)$. The indicator function $g(\boldsymbol{x}) = \text{sign}[f(\boldsymbol{x})]$ takes $+1$ for the region that captures most of the target data, and -1 elsewhere (where the outliers are located). because the distance from the origin to the hyperplane is $\frac{-b}{\boldsymbol{w}}$, therefore minimizing the negative distance is equivalent to maximizing the absolute distance (That is maximizing the margin). The objective task is therefore to minimize $\frac{1}{2}\|\boldsymbol{w}\|_2^2 - b$ as well as the loss. It can be formulated as below:

$$\begin{aligned} &\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \boldsymbol{C}^T\boldsymbol{\xi} - \nu b \\ &s.t. \phi(\boldsymbol{X})^T\boldsymbol{w} - b\mathbf{1} + \boldsymbol{\xi} \geq 0 \\ &\boldsymbol{\xi} \geq 0 \\ &b > 0, \end{aligned} \tag{2.8}$$

where C is a constant vector with elements equal to $\frac{1}{n}$. The Lagrange function is

$$L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\mu}, \boldsymbol{\eta}, \delta) = \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \boldsymbol{C}^T\boldsymbol{\xi} - \nu b - \boldsymbol{\mu}^T(\phi(\boldsymbol{X})^T\boldsymbol{w} - b\mathbf{1} + \boldsymbol{\xi}) - \boldsymbol{\eta}^T\boldsymbol{\xi} - \delta b. \tag{2.9}$$

The KKT conditions [53] are

$$\begin{cases} \boldsymbol{w} = \phi(\boldsymbol{X})\boldsymbol{\mu} \\ \mathbf{1}^T\boldsymbol{\mu} = \nu + \delta \\ \boldsymbol{C} - \boldsymbol{\mu} - \boldsymbol{\eta} = \mathbf{0} \\ \boldsymbol{\mu} * (\phi(\boldsymbol{X})\boldsymbol{w} - b\mathbf{1} + \boldsymbol{\xi}) = \mathbf{0} \\ \boldsymbol{\eta} * \boldsymbol{\xi} = \mathbf{0} \\ \delta b = 0 \\ \boldsymbol{\mu} \geq 0 \\ \boldsymbol{\eta} \geq 0 \\ \delta \geq 0 \\ \phi(\boldsymbol{X})^T\boldsymbol{w} - b\mathbf{1} + \boldsymbol{\xi} \geq 0 \\ \boldsymbol{\xi} \geq 0 \\ b > 0. \end{cases} \tag{2.10}$$

We can express the dual form as

$$\min_{\boldsymbol{\mu}} \frac{1}{2}\boldsymbol{\mu}^T \boldsymbol{K}\boldsymbol{\mu} \tag{2.11}$$

$$s.t. \mathbf{1}^T\boldsymbol{\mu} = \nu$$

$$0 \leq \boldsymbol{\mu} \leq C. \tag{2.12}$$

Thus the decision function is

$$g(\phi(\boldsymbol{x})) = \text{sign}[\boldsymbol{w}^T\phi(\boldsymbol{x}) - b] = \text{sign}[\boldsymbol{\mu}_S^T\phi(\boldsymbol{X}_S)\phi(\boldsymbol{x}) - b], \tag{2.13}$$

where S is the set of indices of nonzero multipliers. If $0 < \mu_i < C$, data point $\boldsymbol{x}_i$ is on the bound and has $f(\boldsymbol{x}_i) = 0$. Thus, a set B that includes indices satisfying $0 < \mu_i < C$ can be found. b is then computed as

$$b = \text{mean}(\boldsymbol{X}_B^T\boldsymbol{X}_S\boldsymbol{\mu}_S). \tag{2.14}$$

The hyperplane based OCSVM has the following important characteristics [53]:

1. The relations between multiplier and position in hypersphere-based SVDD also apply to hyperplane-based SVDD.
2. As in the two-class SVM, it can be proven that ν is an upper bound of the fraction o outliers, and a lower bound of the fraction of support vectors, that is $\frac{n_e}{n} \leq \nu \leq \frac{n_s}{n}$.
3. ν equals $\frac{n_e}{n}$ and $\frac{n_s}{n}$ asymptotically with probability 1.
4. Based on the dual forms of both hypersphere and hyperplane formulations, one can conclude that the hypersphere formulation is equivalent to the hyperplane formulation in the case of constant $K(\boldsymbol{x}, \boldsymbol{x})$ (the linear term in the objective becomes constant).

From Figure 2.3, we can summarize that hypersphere OCSVM determines a hypersphere of radius R around the most similar target samples, excluding the outliers outside of the ball. Hyperplane OCSVM aims at determining a hyperplane that separates normal samples with a maximal margin $\frac{-b}{\|w\|}$ from the origin. They both delimit the same regions of normal samples and outliers except for the light blue region.

2.3 Generative Adversarial Nets

Generative adversarial networks (GANs) [21] are a fairly successful deep-learning-based generative model introduced by Goodfellow et al. A standardized approach called deep convolutional generative adversarial networks (DCGAN) . Later on Alec Radford et al. presented a refined version [66]. “Most GANs today are at least loosely based on the DCGAN architecture.” [20].

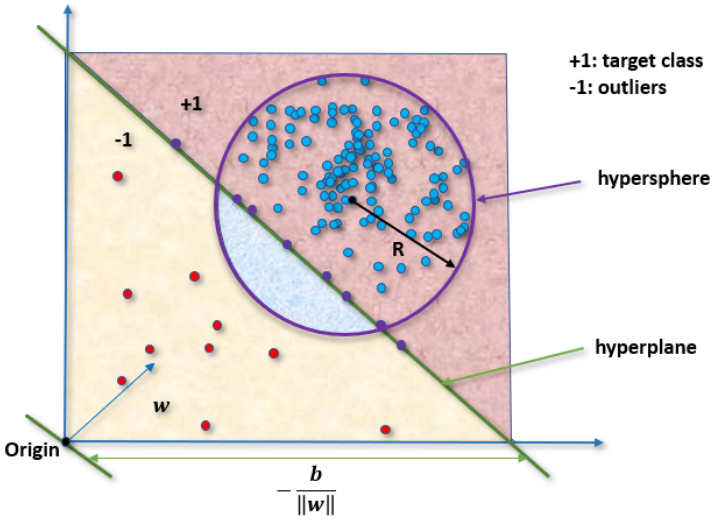


Figure 2.3: Illustration of one class support vector machine.

2.3.1 GAN Model

The GAN model architecture involves two sub-models: a generator model for generating new plausible examples that look real and a discriminator model learning to tell real images apart from fake images generated by generator. More specifically, generator takes fixed-length random vectors in multidimensional latent space as inputs and generates some fake samples, along with real examples from the domain, are provided to the discriminator. The discriminator model (a normal classification model) then classifies these images as real or generated (see Figure 2.4). During training generator model and discriminator model simultaneously, generator progressively improves in creating fake images that look real, while the discriminator becomes more professional at distinguishing real images from fake ones. The training process reaches equilibrium when the discriminator can no longer differentiate between real images from fakes.

2.3.2 GAN Loss

GAN has two loss functions that derive from a single formula (Eq. 2.15; a single measure of distance between the probability distribution of the fake data generated by the GAN and the distribution of the real data) [21]: one for generator training and one for discriminator learning. The goal of generator is trying to minimize Eq. 2.15 while discriminator aims to maximize it.

$$E(G, D) = E_{\boldsymbol{x}}[\log(D(\boldsymbol{x}))] + E_{\boldsymbol{z}}[\log(1 - D(G(\boldsymbol{z})))], \tag{2.15}$$

where real data instance is denoted as $\boldsymbol{x}$ and latent vector (noise) is denoted as $\boldsymbol{z}$; $D(\boldsymbol{x})$ denotes the discriminator's estimate of the probability that $\boldsymbol{x}$ is real; $E_{\boldsymbol{x}}$ presents the expected value over all real data instances; $G(\boldsymbol{z})$ is the generator's output when given $\boldsymbol{z}$; $D(G(\boldsymbol{z}))$ represents the discriminator's estimate of the probability that a fake instance is real; $E_{\boldsymbol{z}}$ is the expected value over all generated fake instances $G(\boldsymbol{z})$. The generator can't directly affect the $\log(D(\boldsymbol{x}))$ term

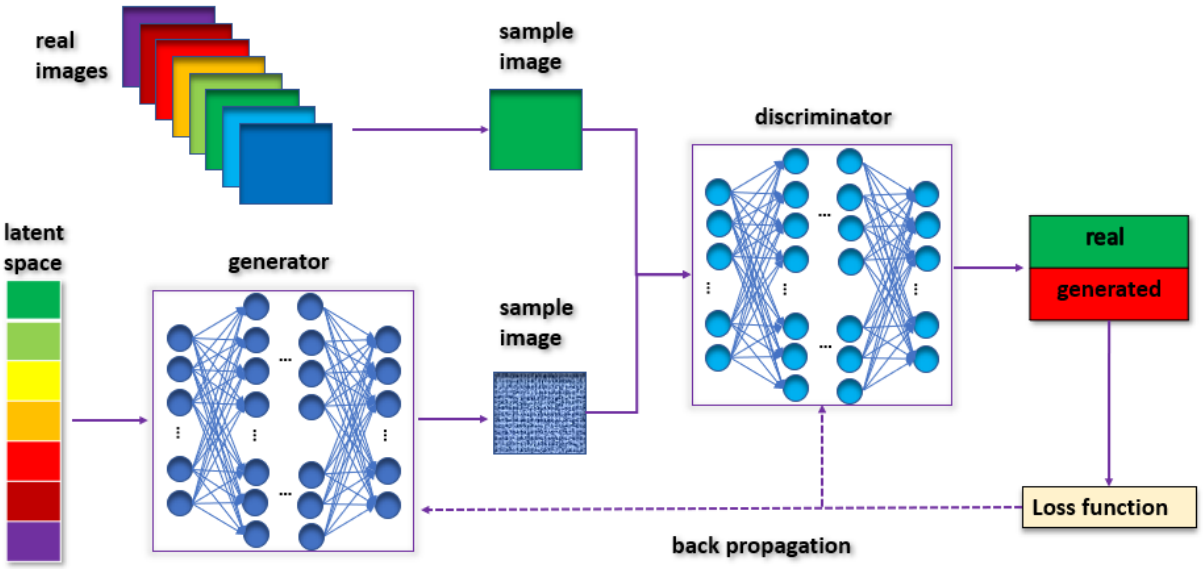


Figure 2.4: Architecture of generative adversarial net.

in the function, so, for the generator, minimizing the loss is equivalent to minimizing $\log(1 - D(G(\boldsymbol{z})))$.

GAN has also been applied for anomaly detection [70]. Since there is no encoder in GAN, Deecke et al. [13] presented the ADGAN algorithm based on the availability of a good representation of a sample in latent space (of its generator) by assuming that the generator is able to effectively capture the distribution of the training data. Li et al. [52] proposed the GAN-AD method for cyber-physical systems (CPSs) . It distinguishes fake data from actual data by taking into consideration of both discrimination loss calculated by the trained discriminator and residual loss between reconstructed and actual test data.

2.4 Capsule Nets

The concept of capsule was first introduced in transforming autoencoder for image modelling [29]. But its potential capacity was not demonstrated until the publications of vector capsule net (CapsNet) [69] and matrix capsule net [32]. Since both models are conceptually very similarly, vector CapsNet is used in my studies. The normality score functions described in Section 3.1 should also work for matrix CapsNet. Figure 2.5 shows an example of such network for the MNIST data where each capsule is a vector of 8 neurons and a digit capsule has 16 neurons. Each capsule is a sparse linear combination of all transformed vectors of lower capsules. A capsule can encapsulate pose information (such as position, orientation, scaling, and skewness) and instantiation parameters (such as color and texture). The connections and activations between a parent capsule and child capsules are dynamically determined, thus transformations and part-whole relationships can be modeled, implying a revolutionary potential for structured data modeling (such as images, videos, and documents).

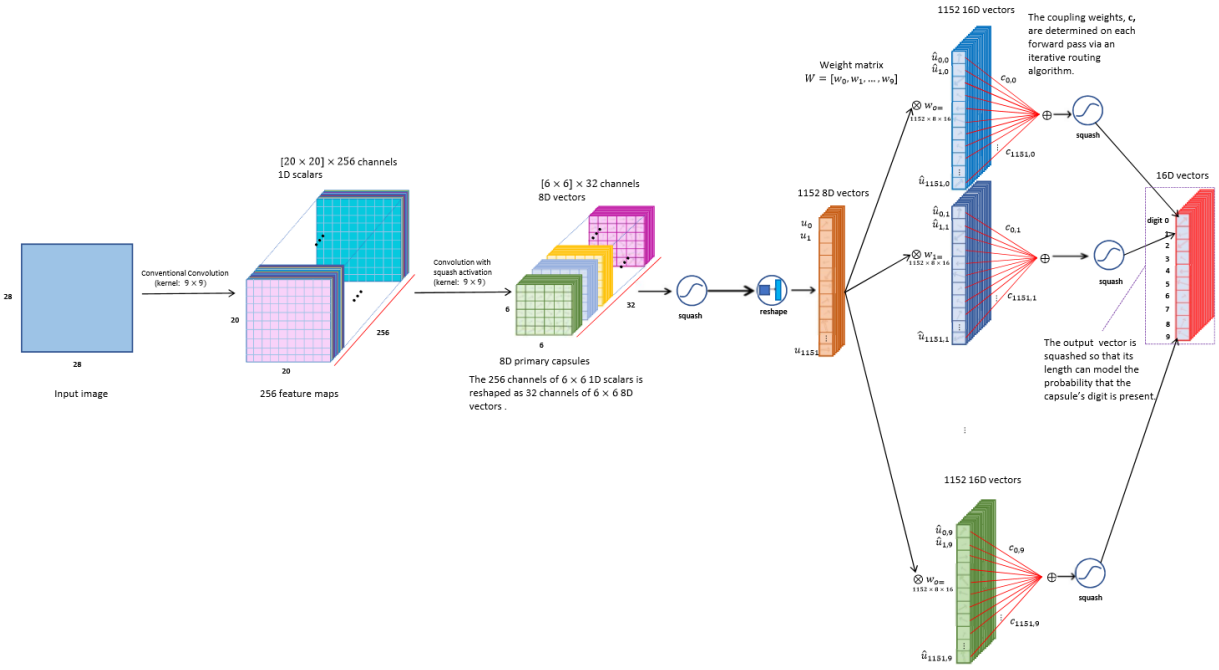


Figure 2.5: Architecture of the CapsNet.

Formally, the j-th capsule at layer $l+1$ can be computed as

$$\boldsymbol{h}_j^{(l+1)} = \frac{\|\boldsymbol{s}_j^{(l+1)}\|_2^2}{1+\|\boldsymbol{s}_j^{(l+1)}\|_2^2} \frac{\boldsymbol{s}_j^{(l+1)}}{\|\boldsymbol{s}_j^{(l+1)}\|_2}, \tag{2.16}$$

where $\boldsymbol{s}_j^{(l+1)}$ is a sparse linear combination of pose vectors from the lower layer:

$$\boldsymbol{s}_j^{(l+1)} = \sum_{i=1}^{K_l} c_{i,j} \boldsymbol{h}_{i,j}^{(l)}, \tag{2.17}$$

where K_l is total number capsules at level l, and a pose vector $\boldsymbol{h}_{i,j}^{(l)}$ is transformed from a capsule:

$$\boldsymbol{h}_{i,j}^{(l)} = \boldsymbol{W}_{i,j} \boldsymbol{h}_i^{(l)} \quad \forall i \text{ in } \{1, \cdots, K_l\}, \tag{2.18}$$

where $\boldsymbol{W}_{i,j}$ is a transformation matrix to be learned using back-propagation, $\boldsymbol{h}_i^{(l)}$ is the i-th capsule at level l, and $c_{i,j}$ is called a coupling coefficient and is computed using a softmax function

$$c_{i,j} = \frac{e^{b_{i,j}}}{\sum_{j'=1}^{K_{l+1}} e^{b_{ij'}}}, \tag{2.19}$$

where $b_{i,j}$ is determined using a dynamic routing algorithm (please see Algorithm 1 for detail) based on inner product (cosine) $\boldsymbol{h}_j^{(l+1)^T} \boldsymbol{h}_{i,j}^{(l)}$.

The squash function in Equation 2.16 allows the length of a capsule, $\|\boldsymbol{h}_j^{(l+1)}\|_2$, to serve as its activation probability. Thus, there is no explicit activation unit in vector CapsNet.

The objective function is defined by a margin loss which uses l_2 norms of digit capsules as prediction probabilities:

$$L = \sum_{c=1}^{C} L_c, \tag{2.20}$$

where C is the total number of classes, and

$$\begin{aligned} L_c &= \delta(y-c)\max(0, m_+ - \|\boldsymbol{h}_c\|_2)^2 \\ &+ \lambda(1-\delta(y-c))\max(0, \|\boldsymbol{h}_c\|_2 - m_-)^2, \end{aligned} \tag{2.21}$$

where y is actual class label, $\boldsymbol{h}_c$ is a digit capsule, $\delta(y-c) = 1$ if and only if $y = c$, $m_+ = 0.9$, $m_- = 0.1$, and $\lambda = 0.5$.

Dynamic routing is used between the primary capsule layer and the digital capsule layer. Thus, in Figure 2.5, the $6 \times 6 \times 32$ lower-level capsules dynamically connect to the 10 higher-level digit capsules. Since bottom-up activation paths form a tree-like structure and a very deep tree is impractical, CapsNet should not have many capsule layers.

Algorithm 1: Dynamic Routing Between Capsules

Result: $\boldsymbol{h}_j^{(l+1)}$
: output from high-level Capsule
Inputs: $\boldsymbol{h}_{i,j}^{l}$: input from low-level Capsule, r: the number of iterations, l: the number of input (low-level) layer
for *all capsule* i *in layer* l *and capsule* j *in layer* $(l+1)$ **do**
  $b_{i,j} \leftarrow 0$;
end
for r *iterations* **do**
  for *all capsule* i *in layer* l **do**
    $\boldsymbol{c}_i \leftarrow \text{softmax}(\boldsymbol{b}_i)$; // ▷ see Eq. (2.19)
  end
  for *all capsule* j *in layer* $(l+1)$ **do**
    $\boldsymbol{s}_j^{(l+1)} \leftarrow \sum_i c_{i,j} \boldsymbol{h}_{i,j}^{(l)}$;
  end
  for *for all capsule* j *in layer* $(l+1)$ **do**
    $\boldsymbol{h}_j^{(l+1)} \leftarrow \text{squash}(\boldsymbol{s}_j^{(l+1)})$; // ▷ see Eq. (2.16)
  end
  for *all capsule* i *in layer* l *and for all capsule* j *in layer* $(l+1)$ **do**
    $b_{i,j} \leftarrow b_{i,j} + \boldsymbol{h}_{i,j}^{l} \cdot \boldsymbol{h}_j^{(l+1)}$;
  end
end

The main differences between Capsules and traditional neurons are summarized in Figure 2.6. One can more clearly see that in traditional neural networks both input and output neurons are scalars, while in CapsNets both input and output Capsules are vectors. Capsules are able to

capture affine transformation of inputs via transformation matrix ($\boldsymbol{w}_{i,j}$) multiplication, whereas traditional neurons have no such quality. CapsNets also introduce a novel nonlinear activation function called squash function. Scalar weightings ($c_{i,j}$) of input vectors in CapsNets are updated by a new routing-by-agreement mechanism, while scalar weightings (w_i) of input scalars in traditional neural networks are tuned by back-propagation algorithm. Furthermore, the bias term ($\boldsymbol{b}$) is not included in calculating output vectors in CapsNets, whereas traditional neural nets usually have that.

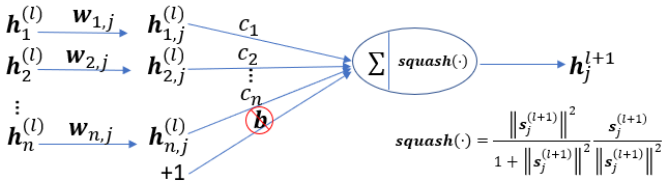
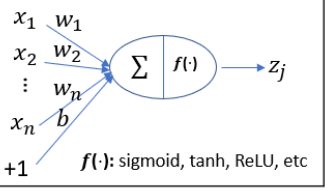
Capsule			Traditional neuron
Input from low-level neuron/capsules		Vector($\boldsymbol{h}_i^{(l)}$)	Scalar(x_i)
Operations	Affine transformation	$\boldsymbol{h}_{i,j}^{(l)} = \boldsymbol{w}_{i,j}\,\boldsymbol{h}_i^{(l)}$	—
	Weighting	$\boldsymbol{s}_j^{(l+1)} = \sum_i c_{i,j}\,\boldsymbol{h}_{i,j}^{(l)}$	$a_j = \sum_i w_i x_i + b$
	Summation		
	Non-linearity Activation	$\boldsymbol{h}_j^{l+1} = \boldsymbol{squash}(\boldsymbol{s}_j^{(l+1)})$	$z_j = \boldsymbol{f}(a_j)$
Output		Vector($\boldsymbol{h}_j^{l+1}$)	Scalar(z_j)
			

Figure 2.6: Main differences between capsules and traditional neurons.

2.5 Variational Lower Bound and Variational Autoencoder

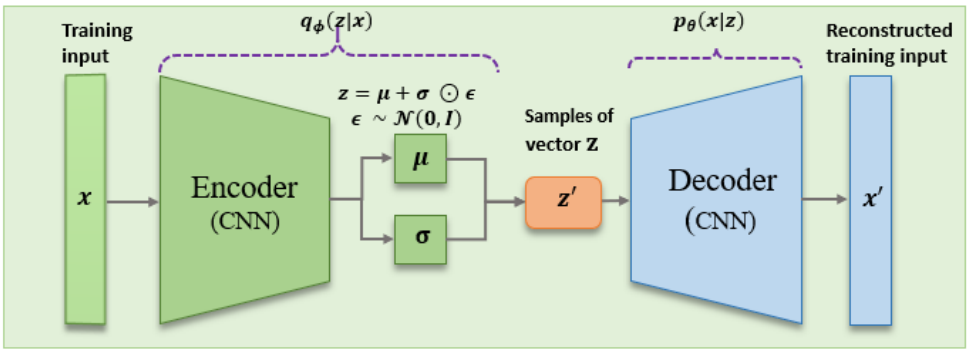


Figure 2.7: Architecture of variational autoencoder.

A directed generative model such as variational autoencoder (VAE) [44, 67] concerns the joint distribution $p_\theta(\boldsymbol{x}, \boldsymbol{z})$ where $\boldsymbol{x}$ is a random vector variable representing an observation in

d-dimension space, $\boldsymbol{z}$ is the vector of latent variables in k-dimensional space, and θ is the set of model hyperparameters. Inference $p_\theta(\boldsymbol{z}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{x})}$ is generally intractable, with few exceptions such as exponential family RBM [57]. An easy parametric distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is introduced to approximate the intractable posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$. Distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ should be as close to $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ as possible by estimating inference parameters ϕ. Based on mean-field variational Bayes, reverse Kullback-Leibler divergence is employed to measure the distance between these two distributions, $D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x})\big)$, with respect to ϕ. After being expanded, the following formula is obtained (p_θ is fixed with respect to q_ϕ),

$$D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x})\big) = \log p_\theta(\boldsymbol{x}) + D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})\big) - \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})]. \tag{2.22}$$

After being rearranged, we can obtain an objective with respect to both generative θ and inference parameter ϕ:

$$\begin{aligned}
\mathcal{L}(\theta,\phi) &= \log p_\theta(\boldsymbol{x}) - D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x})\big) && (2.23)\\
&= \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x},\boldsymbol{z}) - \log q_\phi(\boldsymbol{z}|\boldsymbol{x})] && (2.24)\\
&= \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})\big). && (2.25)
\end{aligned}$$

Since the KL divergence in Equation (2.23) is always non-negative and thus $\mathcal{L}$ is the varitional lower bound (or evidence lower bound, ELBO) of $\log p_\theta(\boldsymbol{x})$:

$$\mathcal{L} = \log p_\theta(\boldsymbol{x}) - D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x})\big) \leq \log p_\theta(\boldsymbol{x}). \tag{2.26}$$

Equation (2.25) implies that maximizing the ELBO is to maximize the reconstruction error and also minimize the difference between the inference distribution and generative prior distribution. From the variational Bayes perspective, the aim is to obtain the optimal generative and inference parameters through maximizing the ELBO:

$$\hat{\theta},\hat{\phi} = \underset{\theta,\phi}{\operatorname{argmax}}\,\mathcal{L}. \tag{2.27}$$

The main challenge in learning of directed generative models is to derive the gradient of Equation (2.24) with respect to the inference parameter. Therefore, the reparameterization trick is introduced in VAE such that backpropagation can be used through expressing the random variable $\boldsymbol{z}$ as a deterministic variable $\boldsymbol{z} = \tau_\phi(\boldsymbol{x},\epsilon)$, where ϵ is an auxiliary independent random variable, and the transformation function τ_ϕ parameterized by ϕ converts ϵ to $\boldsymbol{z}$. For example, when $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is a multivariate Gaussian with a diagonal covariance structure ($q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z};\boldsymbol{\mu},\boldsymbol{\sigma}^2\boldsymbol{I})$), sampling from this distribution is equivalent to:

$$\begin{aligned}
\boldsymbol{\epsilon} &\sim \mathcal{N}(\boldsymbol{0},\boldsymbol{I}), && (2.28)\\
\boldsymbol{z} &= \boldsymbol{\mu}(\boldsymbol{x}) + \boldsymbol{\sigma}(\boldsymbol{x}) \odot \boldsymbol{\epsilon}, && (2.29)
\end{aligned}$$

where $[\boldsymbol{\mu}(\boldsymbol{x}),\boldsymbol{\sigma}(\boldsymbol{x})] = f_\phi(\boldsymbol{x})$ is a deterministic two-head multilayer perceptrons parameterized by ϕ, and $\odot$ denotes element-wise product. The reparametrization trick enables $\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x},\boldsymbol{z}) -$

$\log q_\phi(\boldsymbol{z}|\boldsymbol{x})] = \mathbb{E}_{\boldsymbol{\epsilon}}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z}) - \log q_\phi(\boldsymbol{z}|\boldsymbol{x})]$, such that the gradient of Equation (2.24) with respect to ϕ can be reformulated to

$$\frac{\partial \mathcal{L}(\theta, \phi)}{\partial \phi} = \frac{\partial\, \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z}) - \log q_\phi(\boldsymbol{z}|\boldsymbol{x})]}{\partial \phi} \tag{2.30}$$

$$= \frac{\partial\, \mathbb{E}_{\boldsymbol{\epsilon}}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z}) - \log q_\phi(\boldsymbol{z}|\boldsymbol{x})]}{\partial \phi} \tag{2.31}$$

$$= \mathbb{E}_\epsilon \Big[\frac{\partial\big(\log p_\theta(\boldsymbol{x}, \boldsymbol{z}) - \log q_\phi(\boldsymbol{z}|\boldsymbol{x})\big)}{\partial \phi}\Big] \tag{2.32}$$

$$= \mathbb{E}_\epsilon \Big[\frac{\partial\big(\log p_\theta(\boldsymbol{x}, \tau_\phi(\boldsymbol{x}, \boldsymbol{\epsilon})) - \log q_\phi(\boldsymbol{z}|\boldsymbol{x})\big)}{\partial \phi}\Big]. \tag{2.33}$$

The gradient with respect to generative parameters θ is easier, because it does not rely on the reparameterization trick:

$$\frac{\partial \mathcal{L}(\theta, \phi)}{\partial \theta} = \frac{\partial\, \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z}) - \log q_\phi(\boldsymbol{z}|\boldsymbol{x})]}{\partial \theta} \tag{2.34}$$

$$= \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})} \Big[\frac{\partial\big(\log p_\theta(\boldsymbol{x}, \boldsymbol{z}) - \log q_\phi(\boldsymbol{z}|\boldsymbol{x})\big)}{\partial \theta}\Big] \tag{2.35}$$

$$= \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})} \Big[\frac{\partial \log p_\theta(\boldsymbol{x}, \boldsymbol{z})}{\partial \theta}\Big]. \tag{2.36}$$

2.6 Beta-VAE and Beyond for Disentangled Representation Learning

Higgins et al. [28] proposed a novel deep generative model, named β-VAE, a modification of VAE by introducing an adjustable hyperparameter β to learn an interpretable disentangled representation of the data generative latent factors. Specifically, β functions as a controller to trade off between the extent of learning constraints and reconstruction accuracy. The constraints impose a limit on the capacity of the latent information channel and an emphasis on learning statistically independent latent factors. The authors of [28] demonstrated that β-VAE with appropriately tuned β ($\beta > 1$) qualitatively outperforms VAE ($\beta = 1$) as well as state of the art unsupervised (InfoGAN) and semi-supervised (DC-IGN) approaches to disentangled factor learning on a variety of datasets (celebA, faces and chairs).

Higgins et al. [28] assumed that an image $\boldsymbol{x}$ is generated by the true world simulator using ground truth data generative factors: $p(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{w}) = \text{Sim}(\boldsymbol{v}, \boldsymbol{w})$, where $\boldsymbol{v}$ is set of conditionally independent factors and $\boldsymbol{w}$ is set of conditionally dependent factors. Therefore, the joint distribution of the data $\boldsymbol{x}$ and a set of generative latent factors $\boldsymbol{z}$ is: $p(\boldsymbol{x}|\boldsymbol{z}) \approx p(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{w}) = \text{Sim}(\boldsymbol{v}, \boldsymbol{w})$. The aim of this generative model is then to ensure that the inferred latent factors from $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ capture the generative factors $\boldsymbol{v}$ in a disentangled manner. The conditionally dependent data generative factors $\boldsymbol{w}$ can remain entangled in a separate subset of $\boldsymbol{z}$ that is not entangled with $\boldsymbol{v}$. Considering the prior $p(\boldsymbol{z})$ is set to be an isotropic unit Gaussian $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, a constraint

δ is introduced to encourage the matching between $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{z})$ such that the disentangling property in the inferred $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ can be realized.

Following the same incentive as in VAE: maximizing the probability of generating real data, while minimizing the distance between the generative and approximate posterior distributions, as formulated below

$$\max_{\phi,\theta} \mathbb{E}_{\boldsymbol{x}\sim\mathbb{X}}[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})]] \tag{2.37}$$

$$\text{s.t. } D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big) < \delta, \tag{2.38}$$

where $\mathbb{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ is the training data set. Rewriting this equation as a Lagrangian under the KKT conditions [47, 40], the following equation $\mathcal{F}(\theta, \phi, \beta)$ is obtained:

$$\begin{aligned}
&\mathcal{F}(\theta, \phi, \beta) \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \beta(D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) - \delta\big) \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \beta D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big) - \beta\delta \\
&\geq \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \beta D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big).
\end{aligned} \tag{2.39}$$

The objective function to be maximized in β-VAE is thus defined as:

$$\mathcal{L}_\beta(\theta, \phi) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \beta D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big), \tag{2.40}$$

where the Lagrangian multiplier β is the regularisation coefficient that constrains the capacity of the latent information channel $\boldsymbol{z}$ and puts implicit independence pressure on the learnt posterior due to the isotropic nature of the Gaussian prior $p(\boldsymbol{z})$. When $\beta = 1$, β-VAE corresponds to the original VAE formulation of [44]. When $\beta > 1$, it applies stronger constraint which limits the capacity of $\boldsymbol{z}$ and encourages the model to learn the most efficient representation of the data. Theoretically, a higher β encourages more efficient latent encoding and further encourages the disentanglement. However, a higher β may lead to poorer reconstructions due to the loss of high frequency details when passing through a constrained latent bottleneck.

Burgess et al. [8] proposed an improvement to β-VAE, by progressively increasing the information capacity of the latent code during training. It facilitates the robust learning of disentangled representations in β-VAE without the previous trade-off in reconstruction accuracy. The objective function to be maximized in β-VAE is redefined as:

$$\mathcal{L}_\beta(\theta, \phi) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \beta|D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) - C|, \tag{2.41}$$

where the hyperparameter β controls how heavily to penalise the deviation of $D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big)$ and a controllable value C. By gradually increasing C from zero to a large value, good-quality reconstruction can be obtained.

Hoffman et al. [36] had also made a deeper research on β-VAE considering $\beta < 1$. They argued that optimizing this partially regularized ELBO is equivalent to performing variational expectation maximization (EM) with an implicit prior $r(\boldsymbol{z})$ ($r(\boldsymbol{z}) \propto q_\phi(\boldsymbol{z})^{(1-\beta)}p(\boldsymbol{z})^\beta$) that depends on the marginal (aggregate) posterior $q(\boldsymbol{z}) \triangleq \frac{1}{N}\sum_{i=1}^{N} q(\boldsymbol{z}|\boldsymbol{x}^{(i)})$, and further derived some approximations to examine this prior.

Mathieu et al. [59] explained that most recent work for learning disentangled representations of data with deep generative models has focused on capturing purely independent factors of variation by employing regularizers explicitly encouraging independence in the representations. They argued that such an approach is not generalisable, and quite restrictive for complex models where the true generative factors are not independent, very large in number, or where a set of "true" generative factors cannot be well defined. Overly large β is not universally beneficial for disentanglement, Since this in turn causes a mismatch between marginal posterior $q_\phi(\boldsymbol{z})$ and the prior $p(\boldsymbol{z})$. Thus they proposed a generalization of disentanglement in VAE by explicitly separating such a decomposition as two tasks: a) the latent encoding of data should achieved an appropriate level of non-negligible overlap in aggregate encoding $q_\phi(\boldsymbol{z})$, and b) the aggregate encoding of data $q_\phi(\boldsymbol{z})$ should match the prior $p(\boldsymbol{z})$ which demonstrates the desired dependency structure between latent variables. By improving the match of $q_\phi(\boldsymbol{z})$ and $p(\boldsymbol{z})$, the overlap is only up to an appropriate level. Figure 2.8 illustrates an example of such decomposition where the desired structure is a cross shape, expressed through the prior $p(z)$ as shown on the left of the figure. In the case of insufficient overlap (on the top part), points that are close in the data space are not close in the latent space and so the latent space loses meaning. In the case of too much overlap (on the bottom part), the latent variable and observed datapoint convey little information about one another, such that the latent space again loses meaning. Note that if the distributional form of the latent distribution does not match that of the prior, as is the case here, this can also prevent the aggregate encoding matching the prior when the level of overlap is large.

The authors of [59] developed a new objective that incorporates both a) and b) by introducing an additional divergence term $\mathbb{D}\left(q_\phi(z), p(\boldsymbol{z})\right)$. The objective of this improved β-VAE is defined as below:

$$\mathcal{L}_{\alpha,\beta}(\boldsymbol{x}) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \beta KL(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) - \alpha\, \mathbb{D}(q_\phi(\boldsymbol{z}), p(\boldsymbol{z})). \tag{2.42}$$

By appropriately setting β and α, it allows direct control over the level of overlap and the regularization between the marginal posterior and the prior. Since this loss function contains two hyperparameter α and β, it is simply called $\alpha\beta$-VAE. However, a practical challenge of this method is how to define a proper structured prior when the structure of real hidden factors is poorly known. For this reason, our computational experiment in this thesis is based on [8]'s β-VAE.

Instead of employing the regularization controller β to disentangle the independent data factors by putting implicit independence pressure in the representation, Hamaguchi et al. [26] presented a new method to learn disentangled representation by introducing two additional loss functions $\mathcal{L}_{\text{sim}}$ (a similarity loss function) and $\mathcal{L}_{\text{act}}$ (an activation function). This technique aims to detect trivial events in an image resulting from environmental changes (such as illumination changes, background motions and shadows) by disentangling each image into two kinds of features, specific and common. The functionality of $\mathcal{L}_{\text{sim}}$ constrains common features to represent invariant factors between two paired images. $\mathcal{L}_{\text{act}}$ encourages activation of common features to avoid a trivial solution. After common features of paired images are separated, the means of common features from two images are fed into event detector (a classifier) for training. As mentioned in [26], this method cannot achieve good disentanglement when dealing with complicated scenes, since the activation of the units in the common features are degenerated a certain value.

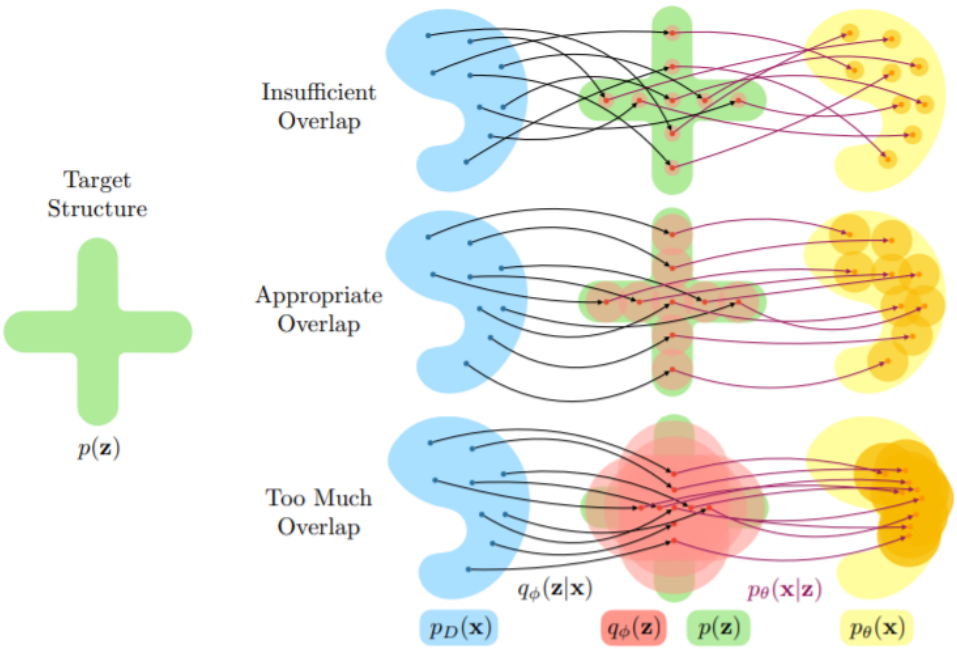


Figure 2.8: Illustration of decomposition where the desired structure is a cross shape. From [59]

2.7 t-SNE for Manifold Learning

The t-distributed stochastic neighbor embedding (t-SNE), introduced by Maaten and Hinton [15], is a popular technique for nonlinear dimensionality reduction that is particularly well suited for visualizing similarity of high-dimensional data that lie on several different, but related, low-dimensional manifolds. It is an improvement of stochastic neighbor embedding (SNE) presented by Hinton and Roweis [31] with easier optimization and better visualization via replacing the SNE cost function with a symmetrized version and using a Student-t distribution instead of a Gaussian to compute the similarity between two points in the low-dimensional space.

2.7.1 Stochastic Neighbor Embedding (SNE)

SNE measures similarities (in both original high-dimensional space and mapped low-dimensional space) between points by converting Euclidean distances between data points into conditional probabilities under Gaussian distributions. In original high-dimensional space, for example, the similarity of a pair of data points x_i and x_j is represented by $p_{j|i}$ under a Gaussian centered at x_i. High $p_{j|i}$ means x_i and x_j are close to each other, and vice versa. $p_{j|i}$ is defined as:

$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-||x_i - x_k||^2/2\sigma_i^2))}, \tag{2.43}$$

where σ_i is the variance of the Gaussian that is centered on data point x_i. It is influenced by the perplexity $\text{Perp}(P_i)$, which is defined as follows,

$$\text{Perp}(P_i) = 2^{H(P_i)}, \tag{2.44}$$

where P_i corresponds particular σ_i and $H(P_i)$ is the Shannon entropy of P_i measured in bits.

For the low-dimensional counterparts y_i and y_j of the high-dimensional data points x_i and x_j, the similarity of these two points is denoted as $q_{j|i}$, and similarly defined as:

$$q_{j|i} = \frac{\exp(-||y_i - y_j||^2)}{\sum_{k \neq i} \exp(-||y_i - y_k||^2)}, \tag{2.45}$$

where the variance of the Gaussian is set to $\frac{1}{\sqrt{2}}$. Since the focus is modeling pairwise similarities, both $p_{i|i}$ and $q_{i|i}$ are set to zero.

2.7.2 Cost Function of t-SNE

Kullback-Leibler divergence is utilized to measure the difference between the low-dimensional data representation distribution $q_{j|i}$ and high-dimensional data distribution $p_{j|i}$. The cost function of SNE, which is sum of Kullback-Leibler divergences over all data points, to be optimized by a gradient descent method, is given by

$$C = \sum_i D_{\text{KL}}(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}. \tag{2.46}$$

Aiming to alleviate the difficult optimization problem of the cost function above and "crowding problem" (comparing nearby data points, moderate distant data points will not occupy reasonably large area in the low-dimensional map), Maaten and Hinton [15] introduced symmetric SNE and employed a Student t-distribution with one degree of freedom (which is the same as a Cauchy distribution) as the heavy-tailed distribution in the low-dimensional map. The cost function is redefined as:

$$C = D_{\text{KL}}(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}, \tag{2.47}$$

where again, p_{ii} and q_{ii} are also set to zero, and

$$p_{ij} = \frac{\exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} \exp(-||x_k - x_l||^2/2\sigma^2)}, \tag{2.48}$$

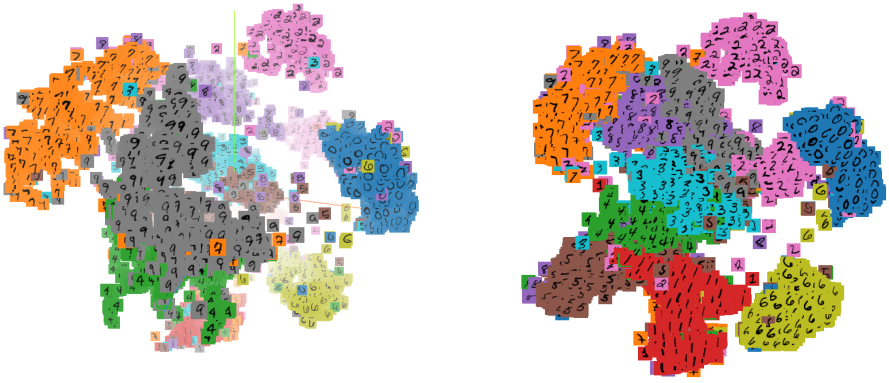
$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}. \tag{2.49}$$

The gradient of the Kullback-Leibler divergence between P and the Student-t based joint probability distribution Q is given by

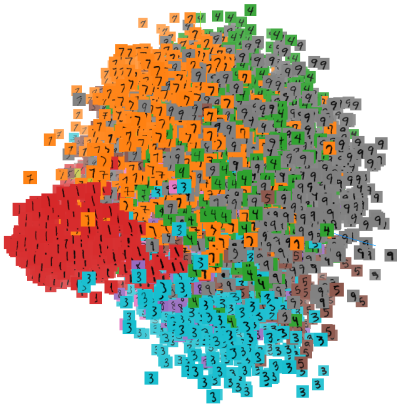
$$\frac{\delta C}{\delta y_i} = 4 \sum_i (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}. \tag{2.50}$$

2.7.3 t-SNE versus PCA

Principal component analysis (PCA) is a linear feature extraction algorithm, which maps high-dimensional data to low-dimensional space in a linear way by using an orthogonal transformation technique. Both PCA and t-SNE have their own advantages and disadvantages, such as t-SNE is computationally expensive algorithm, whereas PCA is fairly fast technique; PCA is not able to interpret the complex polynomial relationship between features while t-SNE is made to capture exactly that. This can be graphically illustrated from Figure 2.9. By comparing t-SNE maps with PCA plot of MNIST, one can clearly see that t-SNE has obvious advantages over PCA for visualization of MNIST data by maintaining some degree of local geometrical non-linear relationships between data, projecting data from the same class to the same cluster.



(a) 3D (left) and 2D (right) t-SNE maps of MNIST.



(b) PCA plot (3 components) of MNIST.

Figure 2.9: t-SNE maps and PCA plot of MNIST (Produced using Embedding Projector [22]).

2.8 Temporal Convolutional Networks

Bai et al. [5] employed a basic architecture which is essentially same as the time delay neural network proposed by Waibel et al. [76] to ensure outputs of same length as inputs and no leakage from the future into the past.

TCN = 1D FCN + causal convolutions.

However, since a simple causal convolutions is not able to achieve a long effective history size, The authors of [5] employed dilated causal convolutions (same architecture as WaveNet [14]) that enable an exponentially large receptive field (please refer to Figure 4.5a). The dilated convolution operation F on element s of the sequence is defined as:

$$F(s) = (\boldsymbol{x} *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot \boldsymbol{x}_{s-d \cdot i}, \tag{2.51}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is a 1-D sequence input, $f : 0, \ldots, k-1 \to \mathbb{R}$ is a filter, d is the dilation factor, k is the filter size, and $s - d \cdot i$ accounts for the direction of the past. When $d = 1$, a dilated convolution reduces to a regular convolution. By choosing larger filter sizes k and increasing the dilation factor d, the receptive field of TCN can be enlarged.

Bai et al. [5] also utilized a generic residual module [27] in place of a convolutional layer. A residual block is defined as:

$$\text{o} = \text{Activation}(\boldsymbol{x} + \mathcal{F}(\boldsymbol{x})).$$

The outputs of a series of transformations $\mathcal{F}$ are added to the input $\boldsymbol{x}$ of the block. To tackle with discrepancy of input and output widths in a standard residual block, an additional 1×1 convolution is used to ensure that element wise addition $\oplus$ receives tensors of the same shape (see Figure 4.5b).

Bai et al. [5] further discussed the advantages of TCN (including parallelism, flexible receptive field size, stable gradients, low memory requirement for training and variable length inputs) and its disadvantages (including possibly high memory requirement for evaluation and potential parameter change for a transfer of domain).

Chapter 3

Anomaly Detection Based on CapsNet

3.1 Architecture and Algorithms

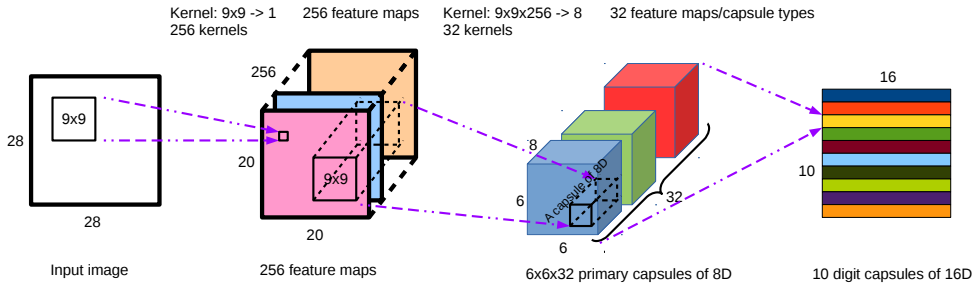
The technique in [19] is based on the insightful observation: learning to discriminate between many types of geometrically transformed images encourages learning of features that are useful for detecting novelties. Among all geometric transformations, the authors of [19] only considered compositions of horizontal flipping, translations, and rotations, resulting in 72 distinct transformations. Their main focus was tackling the problem of identifying anomalous images in pure single class setting, even though it was mentioned their method may also be effective at distinguishing out-of-distribution samples from multiple-class data.

Inspired by the developments of CapsNet, CapsNet is proposed to be employed to automatically learn transformations from the training examples instead of using geometric transformations to learn distinct features of images, such that a test example that cannot be explained by the network should be viewed as anomaly.

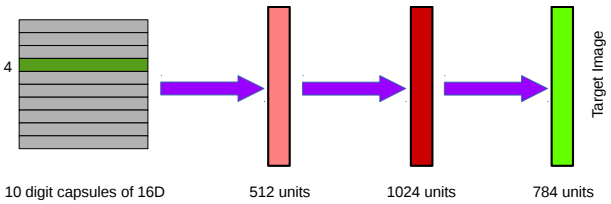
Unlike the method in [19], manually transforming each training image actually is not needed. Using CapsNet, transformations ought to be automatically learned via an iterative routing-by-agreement mechanism. Thus the stage of labeling each transformed image can be ignored. After a CapsNet classifier is trained, unseen images (either normal or out-of-distribution) could be directly fed into the learned model. Aiming to determine the *outlierness* of these images, two normality score functions (prediction-probability-based and reconstruction-error-based) are presented and discussed as following.

3.1.1 Prediction-Probability-Based Normality Score

The activation probabilities of digital capsules at the last layer of a CapsNet indicate the probabilities of the input sample belonging to the classes. However, unlike softmax probabilities in CNN, the activation probabilities of all digit capsules do not necessarily sum to one. Assuming the network is trained sufficiently well, for a normal test image, there should be one and only one probability being close to "1", representing the possibility of this image belonging to its true class. How much the probability is close "1" depends on the classification ability of this



(a) Encoder as classifier.



(b) Decoder as regularizer.

Figure 3.1: The main structure of CapsNet is displayed in (a) is a classifier but can also be viewed as an encoder. The reconstruction regularizer can be viewed as a decoder as displayed in (b).

well-trained CapsNet on test input samples. However, when an anomalous sample cannot be explained by the network, all activation probabilities of digital capsules would be very low. For example, if a normal digit sample is fed into a well-trained CapsNet, the activation probability from the output of squash function corresponding to class 2 will be close to "1" (e.g. 0.98) while the other activation probabilities might be all lower than 0.5. If an anomalous test sample is fed into a well-trained CapsNet, all the activation probabilities from output of squash function might all be lower than 0.5. Therefore, this unique characteristic inspires me to define a normality score function based on prediction probabilities (PP) :

$$\mathcal{N}_{PP}(\boldsymbol{x}) \triangleq \max_{c=1,\cdots,C}(\|\boldsymbol{h}_c\|_2), \tag{3.1}$$

where $\boldsymbol{x}$ is an input image, $\boldsymbol{h}_c$ represents the c-th digit capsule, $\|\boldsymbol{h}_c\|_2$ denotes the probability of $\boldsymbol{x}$ belonging to the c-th class. Hereafter, Equation (3.1) is simply called *PP normality score function*.

3.1.2 Reconstruction-Error-Based Normality Score

In CapsNet, reconstruction error is used as a regularization term through a decoder component (Figure 3.1b). The classifier is also an encoder (Figure 3.1a) for disentangled representation learning. In this perspective, a CapsNet thus meanwhile function as a deep autoencoder, offering an idea to score normality based on reconstruction error.

In this method, normalized squared error (NSE) is used to measure the quality of reconstructed images. The reason of using NSE instead of MSE (mean squared error) is that, different objects in different images (e.g. MNIST digits) can have different numbers of nonzero pixels in

contrast with pure background. Using MSE will significantly weaken the actual difference between the input image and the reconstructed image. For example, as digit image "1" takes much less numbers of pixels than digit "8", using MSE the reconstruction loss of image "1" will be reduced at a greater extent than that of image "8", even though image "1" may be reconstructed worse than image "8". The reconstruction error (RE) based normality score can thus be defined as:

$$\mathcal{N}_{RE}(\boldsymbol{x}) \triangleq -\text{NSE}(\boldsymbol{x}, \boldsymbol{x}') = -\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{\|\boldsymbol{x}\|_2} \tag{3.2}$$

where $\boldsymbol{x}$ represents an actual image and $\boldsymbol{x}'$ the reconstructed image. When the background in an image takes a large portion of the space, and the values of background pixels are near zeros, the advantage of NSE will be more obvious. Hereafter, Equation (3.2) is referred as *RE normality score function*.

3.1.3 Combined Normality Score

Since prediction-probability-based and reconstruction-error-based normality score functions demonstrate their own strengths on a range of datasets (see Section 3.2). Why don't we combine them together so that they can complement each other? Therefore, the combined normality score function (denoted as $\mathcal{N}_{PP+RE}$) is introduced and defined as below:

$$\mathcal{N}_{PP+RE}(\boldsymbol{x}) \triangleq \alpha\mathcal{N}_{RE}(\boldsymbol{x}) + (1-\alpha)\mathcal{N}_{PP}(\boldsymbol{x}), \tag{3.3}$$

where $\alpha \in [0, 1]$ is the combination hyperparameter such that the two terms can effectively complement each other. And the experiments' results in Section 3.2.10 verify the effectiveness of this normality score function.

Algorithm 2: Anomaly detection using CapsNet.

Inputs: $\boldsymbol{X}_{tr}$ set of labeled training images, $\boldsymbol{X}_{te}$ set of unseen test images, $\boldsymbol{f}_\theta$ a CapsNet classifier parametered by θ

Result: normality scores of test images

while *epoch no more than training iterations* **do**
  train $\boldsymbol{f}_\theta$ on the labeled set $\boldsymbol{X}_{tr}$;
end
for $\boldsymbol{x}_{te}^{(i)}$ *within* $\boldsymbol{X}_{te}$ **do**
  $\mathcal{N}_{PP}(\boldsymbol{x}_{te}^{(i)}) \triangleq \max_{c=1,\cdots,C}(\|\boldsymbol{h}_c\|_2)$; // where $\|\boldsymbol{h}_c\|_2$ denotes the probability of $\boldsymbol{x}_{te}^{(i)}$ belonging to the c-th class.
  $\mathcal{N}_{RE}(\boldsymbol{x}_{te}^{(i)}) \triangleq -\text{NSE}(\boldsymbol{x}_{te}^{(i)}, \boldsymbol{x}_{te}'^{(i)}) = -\frac{\|\boldsymbol{x}_{te}^{(i)} - \boldsymbol{x}_{te}'^{(i)}\|_2^2}{\|\boldsymbol{x}_{te}^{(i)}\|_2}$; // where $\boldsymbol{x}_{te}^{(i)}$ represents an actual image and $\boldsymbol{x}_{te}'^{(i)}$ the reconstructed image.
  $\mathcal{N}_{PP+RE} = \alpha\mathcal{N}_{RE}(\boldsymbol{x}) + (1-\alpha)\mathcal{N}_{PP}(x)$; // where $\alpha \in [0, 1]$.
end

3.2 Experiments and Results

3.2.1 Area Under Receiver Operating Characteristics

In this thesis, the performance of all experiments are measured by area under receiver operating characteristics curve (called auROC for short). Receiver operating characteristics curve (ROC curve) is a probability curve and auROC represents degree or measure of separability. It is one of the most important evaluation metrics for measuring any classification model's performance. In case of anomaly detection problem, auROC metric can be employed to evaluate the model's separability of a normal class and an abnormal one, after the possible outliers are distinguished from normal samples by a trained anomaly detection model.

ROC curve plots two parameters: true positive rate (TPR) and false positive rate (FPR) . TPR is a synonym for recall and sensitivity. it is defined as follows:

$$TPR = \frac{TP}{TP + FN}. \tag{3.4}$$

And FPR is defined as:

$$FPR = \frac{FP}{FP + TN}. \tag{3.5}$$

The ROC curve is plotted with TPR against the FPR at different thresholds. Lowering the threshold more items are classified as positive, thus increasing both false positives (FP) and true positives (TP) . AuROC ranges from 0 to 1. A model whose predictions are 100% wrong has an auROC of 0.0; one whose predictions are 100% correct has an auROC of 1.0.

3.2.2 Datasets

Two proposed methods and the three benchmarks were evaluated on three image datasets with increasing difficulty: MNIST, Fashion-MNIST and Small-Norb. In learning stages, all methods were trained using same normal training examples. In test processes, same normal samples and anomalous data were used. Normal and anomalous test data is kept balanced, so that we can focus on comparison using auROC and leave imbalanced issue for future investigation.

- **MNIST**: It contains a training set of 60000 28×28 grayscale digit images and a test set of 10000 same resolution grayscale examples from approximately 500 different writers [50].
- **Fashion-MNIST**: It is a dataset of Zalando's article images, comprising 70000 28×28 MNIST-like labeled fashion images with 7000 images per category [78]. The training set has 60000 images and the test set has 10000 images. The samples come from 10 classes: T-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot.
- **Small-Norb**: It contains 24300 96×96 grayscale images pairs of 50 toys belonging to 5 generic categories: four-legged animals, human figures, airplanes, trucks, and cars [51]. The objects were imaged by two cameras under 6 lighting conditions, 9 elevations , and 18 azimuths. As in [69], the images were resized to 48×48; random 32×32 crops of them

were obtained during training process. Central 32×32 patches of test images were used during test.

- **CIFAR-10**: It consists of 60000 32×32 colour images in 10 classes, with 6000 images per class. There are 10000 test images which include exactly 1000 randomly-selected images from each class and 50000 training images (with 5000 images from each class) are randomly grouped into 5 batches [45].

3.2.3 Benchmarks

CapsNet based anomaly detection methods were compared with three principled methods from the two family of methods discussed in Chapter 2. These benchmarks are described as below. Please note that the method presented in [19] is unavailable for multi-class normal data, thus it is unable to be compared with AnoCapsNet methods.

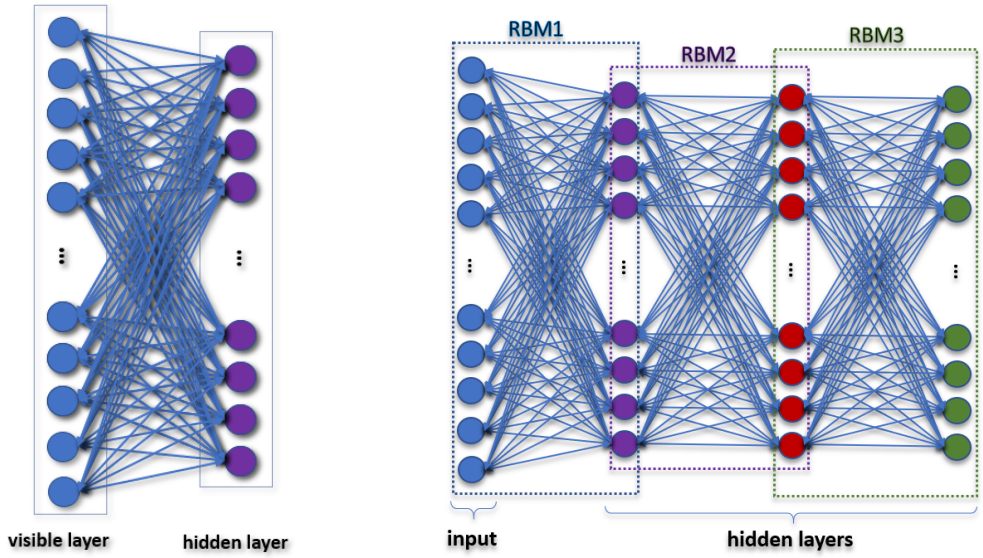
- A deep hybrid method named CNN+OCSVM is implemented with CNN being used to learn the latent representations and OCSVM being used for anomaly detection. The CNN component has two convolutional layers (3×3 receptive fields in both layers, 32 and 64 feature maps respectively for the first and second layers, max-pooling with pooling size of 2×2 after the second layer, ReLU activation function), one fully connected layer (128 units), and a softmax layer for class labels of normal samples. The outputs of the fully connected layer were latent representations extracted for searching the hyperparameters and optimizing the model parameters of OCSVM.

- Three-layer DBN [30, 56] is employed to model data distribution and then reconstruction error is measured to score abnormality. There are 500 units in each hidden layer. The model was layer-wise pretrained by RBMs [57]. Bernoulli distribution was assumed for both visible and hidden layers. The pixel values were scaled to interval $[0, 1]$ as input to DBN. Reconstruction error of an inquiry sample was used to detect anomalous samples.

- Similarly, a convolutional VAE was also applied to capture data distribution. The inference component (encoder) has the same architecture as the CNN component (disregarding the output layer) in CNN+OCSVM. The latent space size was set to 64. The structure of the inference component was mirrored for the structure of the generative component. Reconstruction error was used in determination of anomality.

3.2.4 Experimental Setup

The architecture of CapsNet used in this thesis is identical to that of [69]. The encoder network includes three layers: convolutional layer, primaryCaps layer and digitCaps layer. The decoder part consists of 3 fully connected layers. The convolutional layer in the CapsNet has 256 kernels with size of $9 \times 9 \times 1$ and stride 1, followed by ReLU activation. The primaryCaps has 32 Capsules with eight $9 \times 9 \times 256$ convolutional kernels (with stride 2) on each Capsule. DigitCaps has 10 digit capsules corresponding to 10 digits. The learning rate was chosen as 0.001 for all



(a) A restricted Boltzmann machine (RBM). (b) A deep belief network (DBN) with three hidden layers.

Figure 3.2: Structures of RBM and DBN.

experiments; the number of epochs was set to 20 for experiments on MNIST dataset, and 50 for Fashion-MNIST, Small-Norb and CIFAR-10 datasets; batch size was chose as 100 for MNIST, Fashion-MNIST and CIFAR-10 datasets and 64 for Small-Norb dataset. Algorithm 2 shows the process of calculating normality scores of unseen test images. A CapsNet firstly is trained using labeled training images. An unseen test image is then fed into the well-trained CapsNet to calculate its three normality scores.

3.2.5 Computational Complexity

Same as CNN, for evaluation of a single sample, the computational complexity of CapsNet is decided by the total number of weights and neurons. Given that every neuron has one bias parameter, it could be ignored. Therefore, the computational complexity becomes $O(w)$, where w is the number of total weights (including weights related to fully connection, convolution and affine transformation). Back-propagation has the same complexity as the forward evaluation. Thus, the complexity for training a CapsNet using m samples with n iterations will be $O(m \times n \times w)$. In this proposed method, normality score functions are calculated through simple arithmetic operations (such averaging and squaring) on each test sample, hence they could also be ignored. Finally, the computational complexity of AnoCapsNet will be $O(m \times n \times w)$.

3.2.6 Comparison on MNIST Dataset

Both RE and PP methods and three benchmarks: CNN+OCSVM, DBN, and VAE were compared on MNIST data by respectively treating each digit class as anomalous class and the rest

as normal. Their performance in terms of auROC is displayed in Figure 3.3. Evidently the CapsNet-based PP method works consistently the best, while CapsNet-based RE method in general is slightly inferior to the PP method, but has comparable results as CNN+OCSVM. The two DGMs (DBN and VAE) are not competitive to the CapsNet-based and deep hybrid methods.

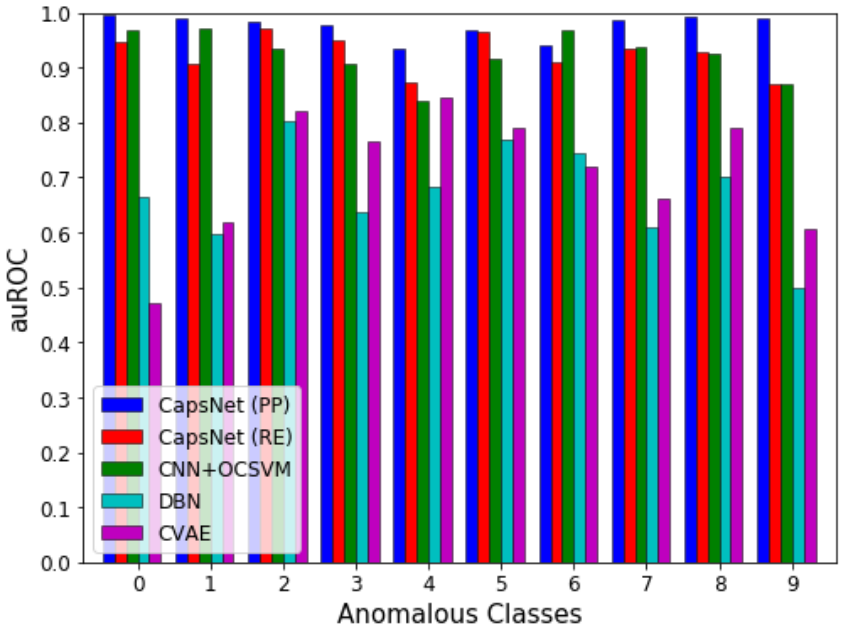


Figure 3.3: Performance of five methods on the MNIST data.

3.2.7 Comparison on Fashion-MNIST Dataset

When comparing all five methods on the Fashion-MNIST data (see Figure 3.4), generally speaking all methods tended to get lower results, which is reasonable because Fashion-MNIST samples is more complicated than MNIST samples. When using footwear (Sandals, Sneakers, or Ankle boots) as anomalous samples, the CapsNet(RE) method outperformed the CapsNet(PP) method. It is because the CapsNet classifier could carry some wrong confident information of classifying a footwear anomalous sample (e.g. Sneaker) to the normal footwear classes (e.g. Sandal or Ankle boots), while reconstruction errors can pick up differences in details. In the case of using data from a topwear class as anomalous samples, CapsNet(PP) worked better than CapsNet(RE). When trousers and bags were viewed as abnormal samples, both methods worked quite well without big differences in performance. The performances of CNN+OCSVM and VAE vary largely. In few cases, they can obtain similar results as CapsNet-based methods. DBN did not behave impressively on the data.

3.2.8 Comparison on Small-Norb Dataset

From Figure 3.5, one can see that Small-Norb is a challenging data for all five methods. When using animals and cars as anomalies, only CapsNet(PP) performed reasonably good, the other

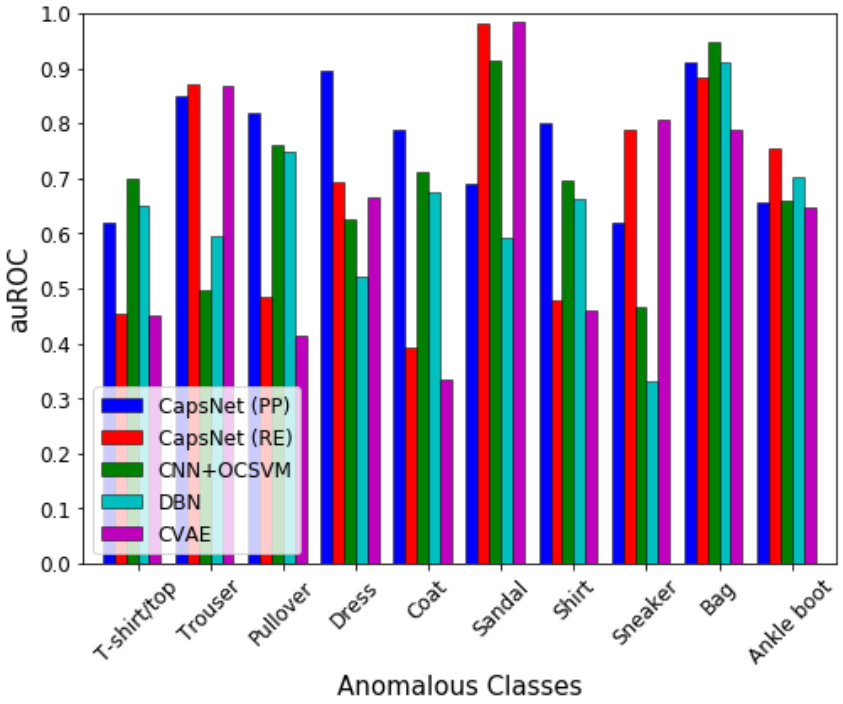


Figure 3.4: Performance of five methods on the Fashion-MNIST data.

methods behaved randomly. When trucks were used as anomalous samples, only CapsNet(PP) and CapsNet(RE) were able to behave non-randomly. In the case of planes as abnormal data, both CapsNet(RE) and VAE worked the best. Only in the case of humans as anomalies, CNN-OCSVM reached 0.7 auROC.

3.2.9 Insight into CNN-OCSVM Method

From the above comparisons, it seems that except CapsNet-based methods, CNN-OCSVM is the model that achieved impressive results, especially when doing anomaly detection on MNIST dataset. During these experiments, the main hyperparameters of CNN-OCSVM method (e.g. ν and γ) is carefully tuned, in order to allow it achieves its potential. The performance of CNN-OCSVM with tuned hyperparameters are depicted by Figures 3.6, 3.7 and 3.8. The bar chart of each figure represents the results corresponding to each anomalous class and the bottomed table illustrates three hyperparameters' values corresponding to each anomaly case.

3.2.10 Case Studies in Normality Score Functions

The performance of three normality score functions is evaluated through two case studies on the MNIST data.

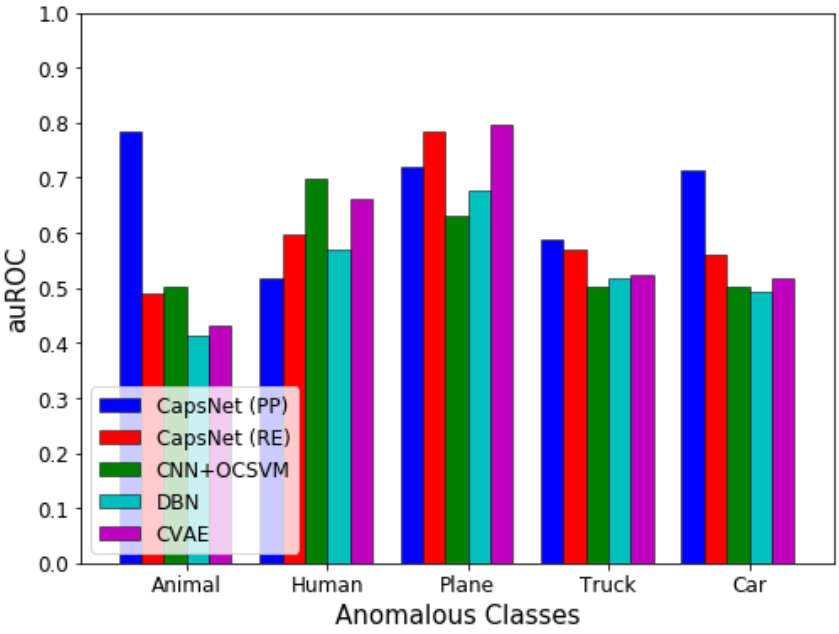
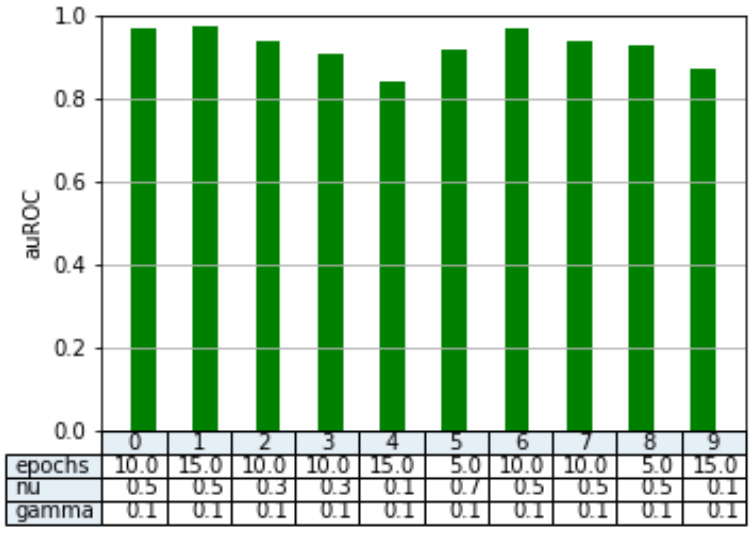


Figure 3.5: Performance of five methods on the Small-Norb data.

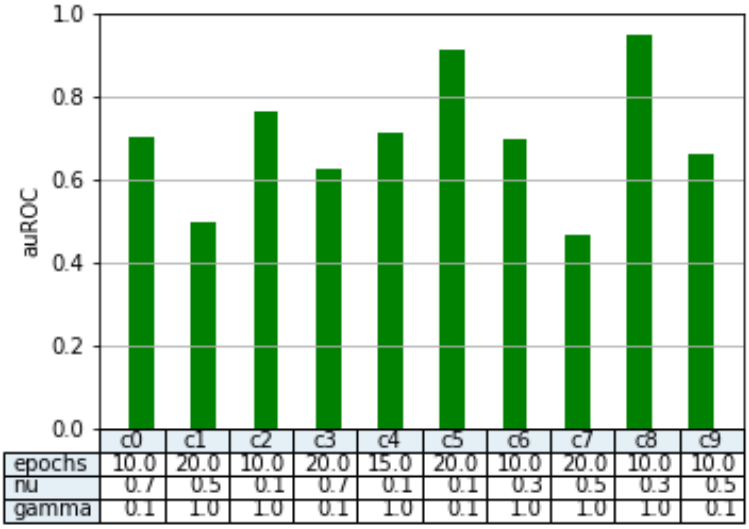


s

Figure 3.6: Performance of CNN-OCSVM on the MNIST data.

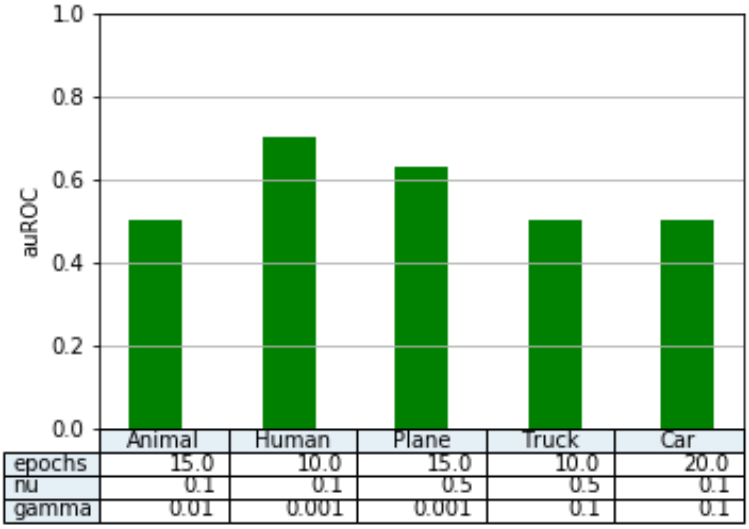
Multi-Class Training Data and Single-Class Anomalous Digits

1. **PP-based Normality Score VS. RE-based Normality Score**
   Figures 3.9a and 3.9b shows the performance of two methods when selecting digits "2" and "9" as abnormal samples, respectively, and the rest digits as normal samples. When digit "2" was treated as anomaly class, both PP and RE score functions achieved near



	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
epochs	10.0	20.0	10.0	20.0	15.0	20.0	10.0	20.0	10.0	10.0
nu	0.7	0.5	0.1	0.7	0.1	0.1	0.3	0.5	0.3	0.5
gamma	0.1	1.0	1.0	0.1	1.0	0.1	1.0	1.0	1.0	0.1

Figure 3.7: Performance of CNN-OCSVM on Fashion-MNIST data. "c0, c1, c2, c3, c4, c5, c6, c7, c8, c9" represent 10 classes: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot.

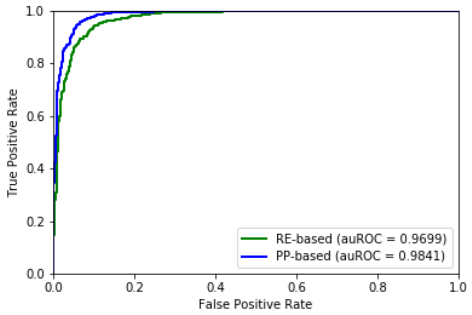


	Animal	Human	Plane	Truck	Car
epochs	15.0	10.0	15.0	10.0	20.0
nu	0.1	0.1	0.5	0.5	0.1
gamma	0.01	0.001	0.001	0.1	0.1

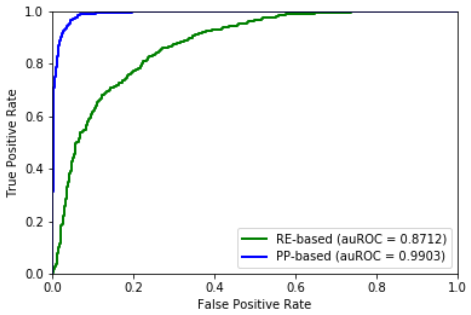
Figure 3.8: Performance of CNN-OCSVM on the Small-Norb data.

perfect auROCs (0.9841 and 0.9699). When "9" was viewed as anomaly class, however, the PP score function outperformed the RE score function by auROC of 0.12. The reason for this can be found from Figures 3.9c and 3.9d, which depicts 50 real digits and their reconstructed ones for both digits. From Figures 3.9c and 3.9d, one can easily notice that anomalous digit "9" was mostly reconstructed as digit "4", while anomalous digit "2" was

reconstructed as several different digits such as “1”, “3”, “6”, and “7”. As images “4” and images “9” are quite similar, the normality scores of images “9” become very high, even though the true digits “9” and their reconstructed versions “4” are two different numbers. Similar situations can be observed when considering other digits as anomaly, like “4” and “1”. For detailed auROCs and reconstructed images of all anomalous digits, please see Figures 3.11 and 3.12.



(a) Digit 2 (as anomaly).



(b) Digit 9 (as anomaly).

(c) Digit 2 (as anomaly).

(d) Digit 9 (as anomaly).

Figure 3.9: ROC curves and images (Original digits (upper half) and reconstructed digits (lower half)) when detecting anomalous digits: “2” and “9” using AnoCapsNet.

2. **Performance of PP+RE-based Normality Score**
   From the aforementioned experiments, one can be convinced that both CapsNet-based PP and RE methods displayed their own strengths on specific anomalous classes. Not surprisingly, the combined method (PP+RE) performed the best among all three normality score functions by making optimum use of advantages of PP and RE methods. Table 3.1 depicts the results of three normality score functions on four image datasets. From Table 3.1, one can easily notice that outstanding averaged 0.981 auROC was achieved by PP+RE-based method when detecting anomalous data on MNIST. In cases of Fashion-MNIST, Small-Norb and CIFAR-10, the averaged auROCs were also improved by around 7%, 4% and 6% of the highest auROCs respectively. For detailed auROCs of these three methods on all datasets, please see Figures 3.11, 3.13, 3.15 and 3.16. It can be concluded that (1) PP-based method had consistently better performance than RE-based method when evaluating AnoCapsNet on MNIST. And it outperformed RE method mostly on Fashion-MNIST, Small-Norb and CIFAR-10. (2) PP+RE method sometimes overlapped with PP

method (when PP outperformed RE at a great degree) and occasionally overlapped with PP method (considering RE had much better performance than PP). In other cases (PP and RE had close performance), PP+RE method improved both methods at some degrees.

Three normality scores of AnoCapsNet are also compared with other advanced models (AnoGAN, ADGAN and AnoDM) in the Chapter 4. The results displayed in Table 4.2, prove that the proposed PP+RE-based AnoCapsNet framework improved current successful image anomaly detection techniques at a great degree. It accomplished the state-of-art performance on a range of image datasets, even for some challenging datasets such as Small-Norb and CIFAR-10.

Multi-Class Training Data and Multiple Anomalous Digits

In this case, the three CapsNet-based normality score functions are tested by considering digits "0", "3" and "5" as abnormal digits, and the rest as normal. The results is displayed in Figure 3.10. Both RE and PP methods achieved similar auROCs. By combining them together (using PP+RE normality score function), the state-of-the-art result (0.9919 auROC) is achieved by AnoCapsNet framework.

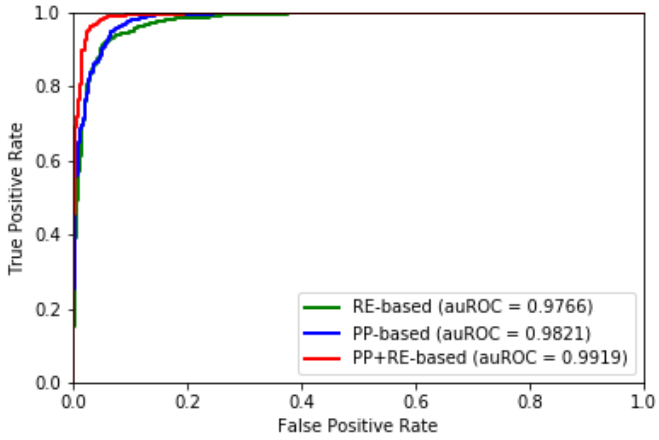
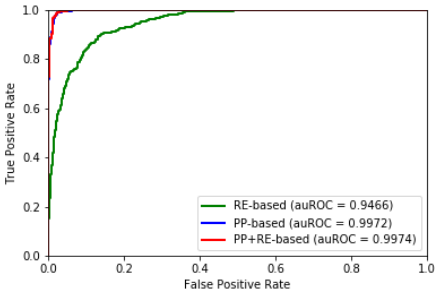


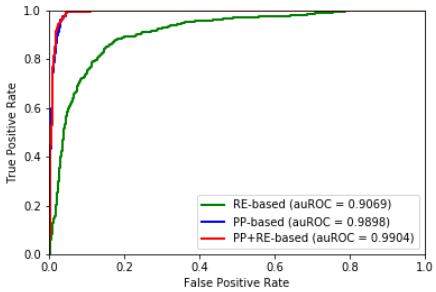
Figure 3.10: ROC curves of detecting abnormal digits "0", "3", and "5" using three CapsNet-based score functions.

Table 3.1: Performance of three normality score (PP, RE and PP+RE) based AnoCapsNet.

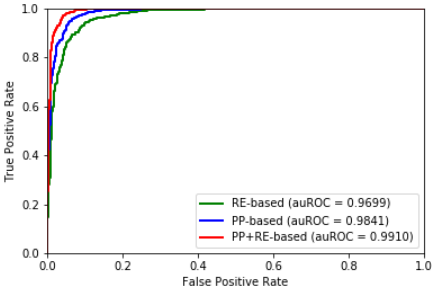
Dataset	Class	PP-based	RE-based	PP+RE-based
MNIST	0	0.997	0.947	**0.998**
	1	**0.990**	0.907	**0.990**
	2	0.984	0.970	**0.991**
	3	0.976	0.949	**0.984**
	4	0.935	0.872	**0.939**
	5	0.970	0.966	**0.983**
	6	0.942	0.909	**0.959**
	7	**0.987**	0.934	**0.987**
	8	**0.993**	0.929	**0.993**
	9	**0.990**	0.871	**0.990**
	avg.	0.977	0.925	**0.981**
Fashion-MNIST	0	**0.620**	0.454	**0.620**
	1	0.851	0.871	**0.915**
	2	**0.818**	0.486	**0.818**
	3	0.895	0.693	**0.898**
	4	**0.790**	0.394	**0.790**
	5	0.691	**0.982**	**0.982**
	6	**0.801**	0.480	**0.801**
	7	0.619	**0.787**	**0.787**
	8	0.912	0.885	**0.960**
	9	0.656	**0.754**	**0.754**
	avg.	0.765	0.679	**0.833**
Small-Norb	0	**0.785**	0.491	**0.785**
	1	0.519	**0.598**	**0.598**
	2	0.718	0.785	**0.812**
	3	0.589	0.570	**0.605**
	4	0.713	0.562	**0.714**
	avg.	0.665	0.601	**0.703**
CIFAR-10	0	**0.645**	0.377	**0.645**
	1	0.452	**0.736**	**0.736**
	2	**0.646**	0.413	**0.646**
	3	0.666	0.597	**0.686**
	4	**0.670**	0.390	**0.670**
	5	0.645	0.590	**0.669**
	6	**0.723**	0.486	**0.723**
	7	0.704	0.628	**0.733**
	8	0.477	0.403	**0.477**
	9	0.504	0.688	**0.788**
	avg.	0.613	0.531	**0.677**



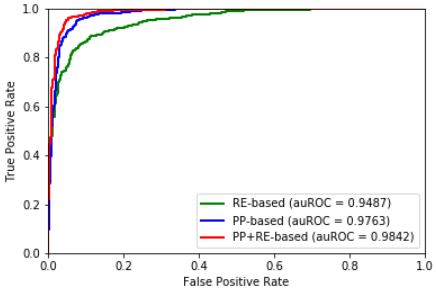
(a) digit 0 (as anoamly)



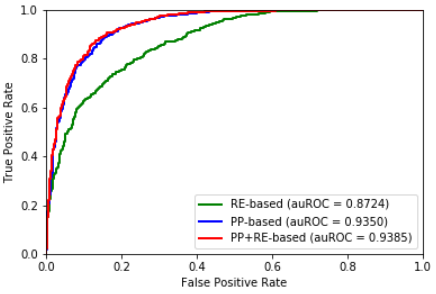
(b) digit 1 (as anoamly)



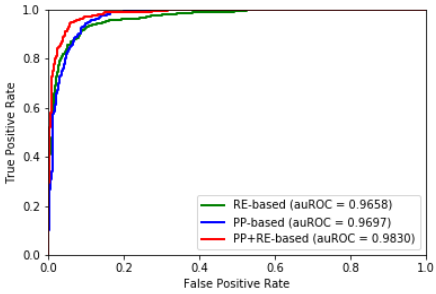
(c) digit 2 (as anoamly)



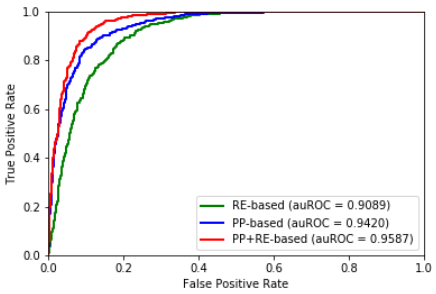
(d) digit 3 (as anoamly)



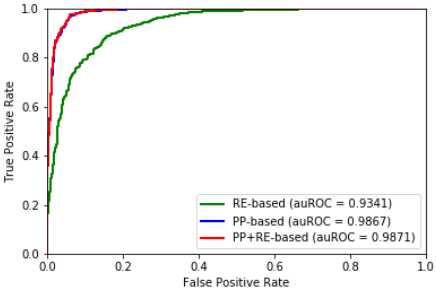
(e) digit 4 (as anoamly)



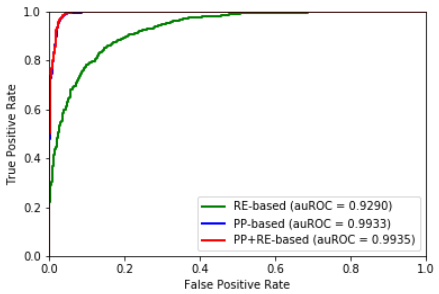
(f) digit 5 (as anoamly)



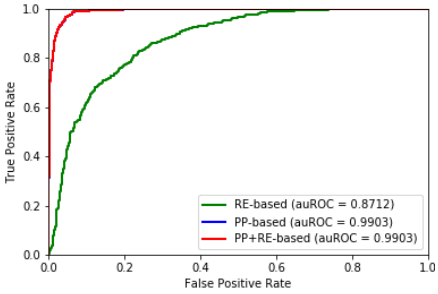
(g) digit 6 (as anoamly)



(h) digit 7 (as anoamly)

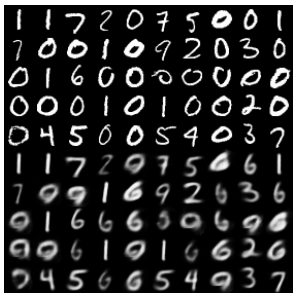


(i) digit 8 (as anoamly)



(j) digit 9 (as anoamly)

Figure 3.11: ROC plots of AnoCapsNet on MNIST.



(a) digit 0 (as anoamly)

(b) digit 1 (as anoamly)

(c) digit 2 (as anoamly)

(d) digit 3 (as anoamly)

(e) digit 4 (as anoamly)

(f) digit 5 (as anoamly)

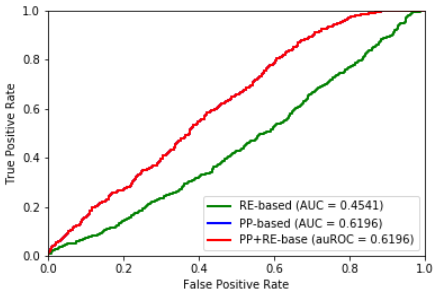
(g) digit 6 (as anoamly)

(h) digit 7 (as anoamly)

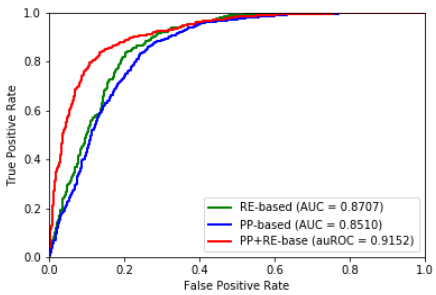
(i) digit 8 (as anoamly)

(j) digit 9 (as anoamly)

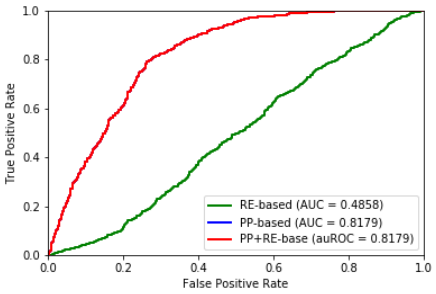
Figure 3.12: Original images (upper half) and reconstructed images (lower half) of AnoCapsNet on MNIST.



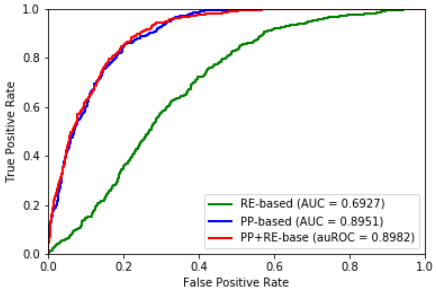
(a) “T-shirt/top” (as anoamly)



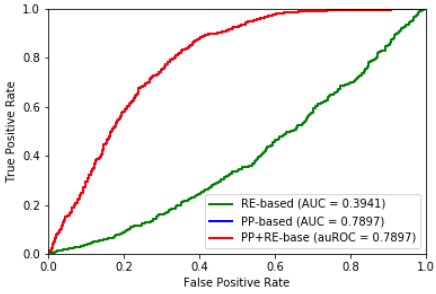
(b) “Trouser/pants” (as anoamly)



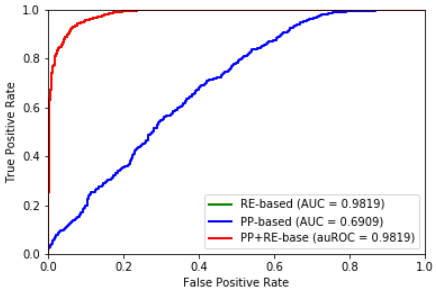
(c) “Pullover shirt” (as anoamly)



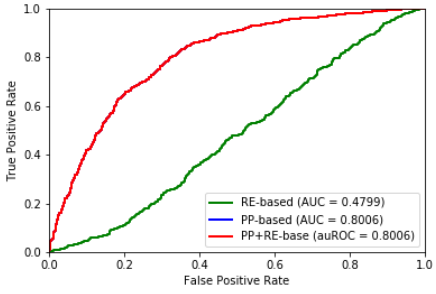
(d) “Dress” (as anoamly)



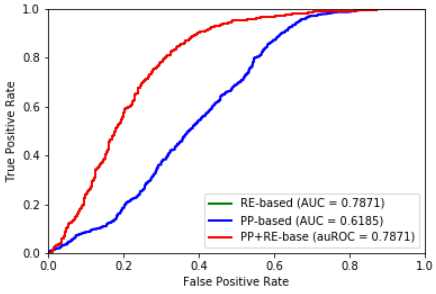
(e) “Coat” (as anoamly)



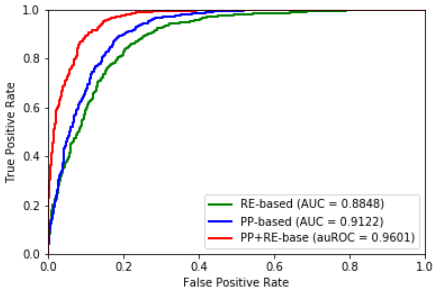
(f) “Sandal” (as anoamly)



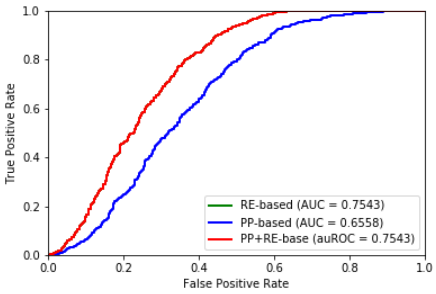
(g) “Shirt” (as anoamly)



(h) “Sneaker” (as anoamly)

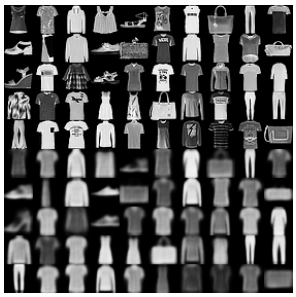


(i) “Bag” (as anoamly)

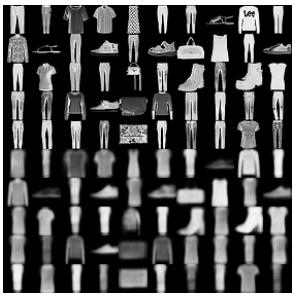


(j) “Ankle boot” (as anoamly)

Figure 3.13: ROC plots of AnoCapsNet on Fashion-MNIST.

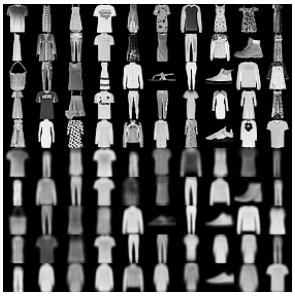


(a) “T-shirt/top” (as anoamly)

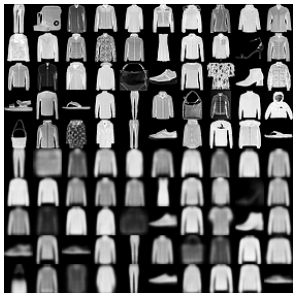


(b) “Trouser/pants” (as anoamly)

(c) “Pullover shirt” (as anoamly)

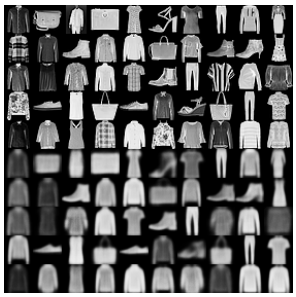


(d) “Dress” (as anoamly)



(e) “Coat” (as anoamly)

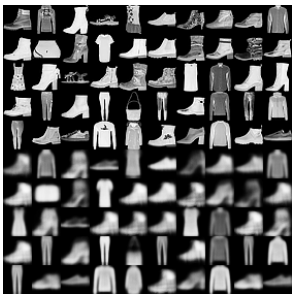
(f) “Sandal” (as anoamly)



(g) “Shirt” (as anoamly)

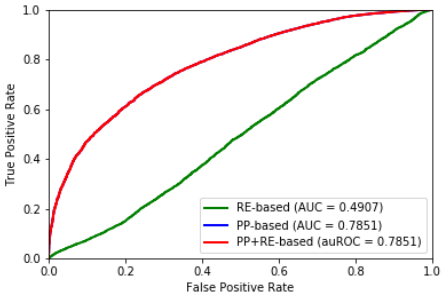
(h) “Sneaker” (as anoamly)

(i) “Bag” (as anoamly)

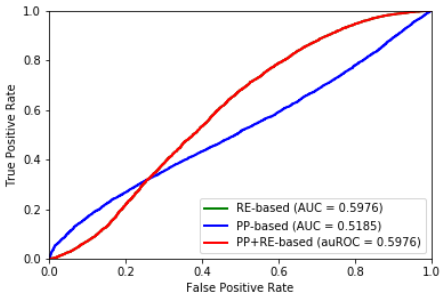


(j) “Ankle boot” (as anoamly)

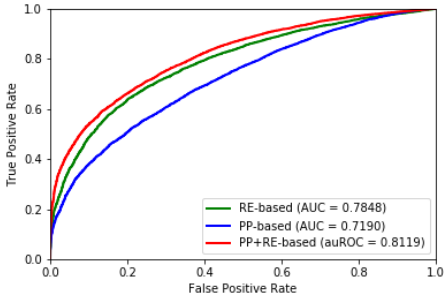
Figure 3.14: Original images (upper half) and reconstructed images (lower half) of AnoCapsNet on Fashion-MNIST.



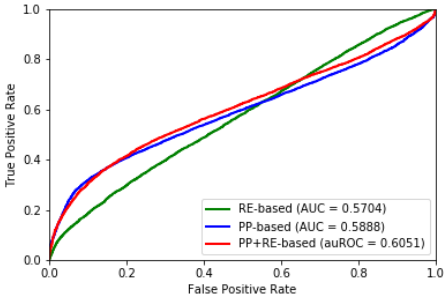
(a) “Four-legged animals” (as anoamly)



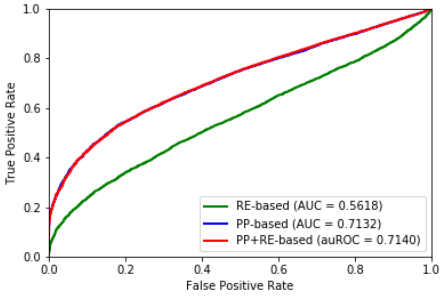
(b) “Human figures” (as anoamly)



(c) “Airplanes” (as anoamly)

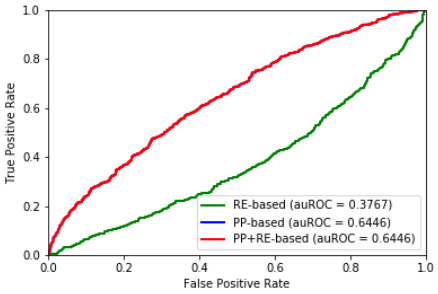


(d) “Trucks” (as anoamly)

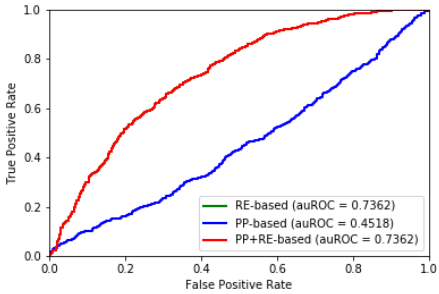


(e) “Cars” (as anoamly)

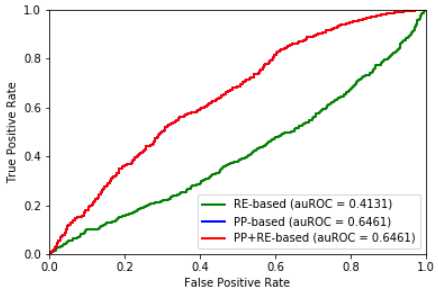
Figure 3.15: ROC plots of AnoCapsNet on Small-Norb.



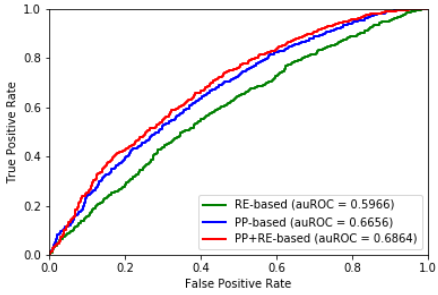
(a) “Airplane” (as anoamly)



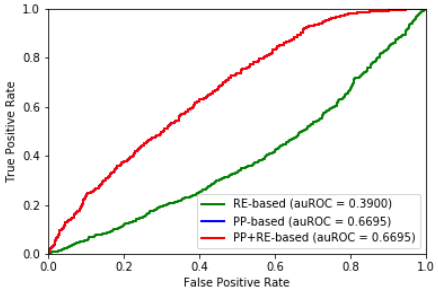
(b) “Automobile” (as anoamly)



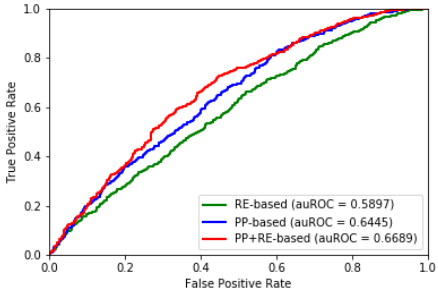
(c) “Bird” (as anoamly)



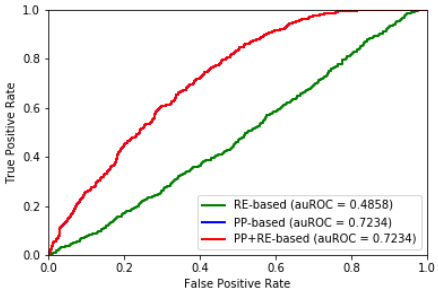
(d) “Cat” (as anoamly)



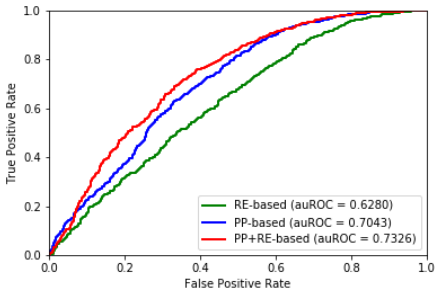
(e) “Deer” (as anoamly)



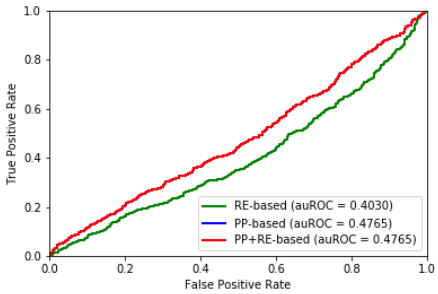
(f) “Dog” (as anoamly)



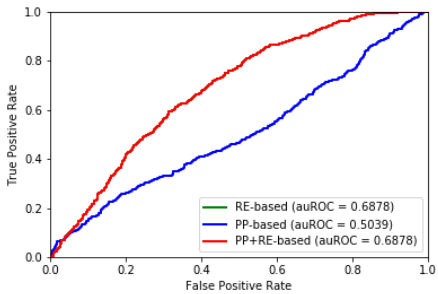
(g) “Fog” (as anoamly)



(h) “Horse” (as anoamly)



(i) “Ship” (as anoamly)



(j) “Truck” (as anoamly)

Figure 3.16: ROC plots of AnoCapsNet on CIFAR-10.

Chapter 4

Anomaly Detection Based on Unsupervised Disentangled Representation Learning and Manifold Learning

4.1 Architecture and Algorithm

In this chapter, the second novel generic anomaly detection framework (named AnoDM) is presented, which the first time combines unsupervised disentangled representation learning (implemented using β-VAE as an example) and low-dimensional manifold learning (currently using t-SNE as implementation) together to detect outliers via effectively taking the advantages of reconstruction at raw feature space and disentangled latent distribution in t-SNE map. Figure 4.1 shows the architecture of AnoDM which includes two main phases: (1) unsupervised disentangled representation learning and (2) anomaly detector. After β-VAE is learned using unlabelled training normal samples, it then can be employed by the anomaly detector to efficiently obtain latent encoding and reconstructed version of a sample. Once latent embeddings of both training samples (or a representative subset from the training data) and a test sample are obtained, t-SNE is used to map the latent representations of these samples further to the 2-dimensional space (called t-SNE space or map), such that the average distance between the 2D representation of the test sample and its k nearest neighbors from the 2D representations of training samples is calculated. Finally, this distance is combined with the reconstruction error of the test sample to define its anomaly score. The essential parts of this framework are discussed below in details. Full AnoDM approach is given in Algorithm 3.

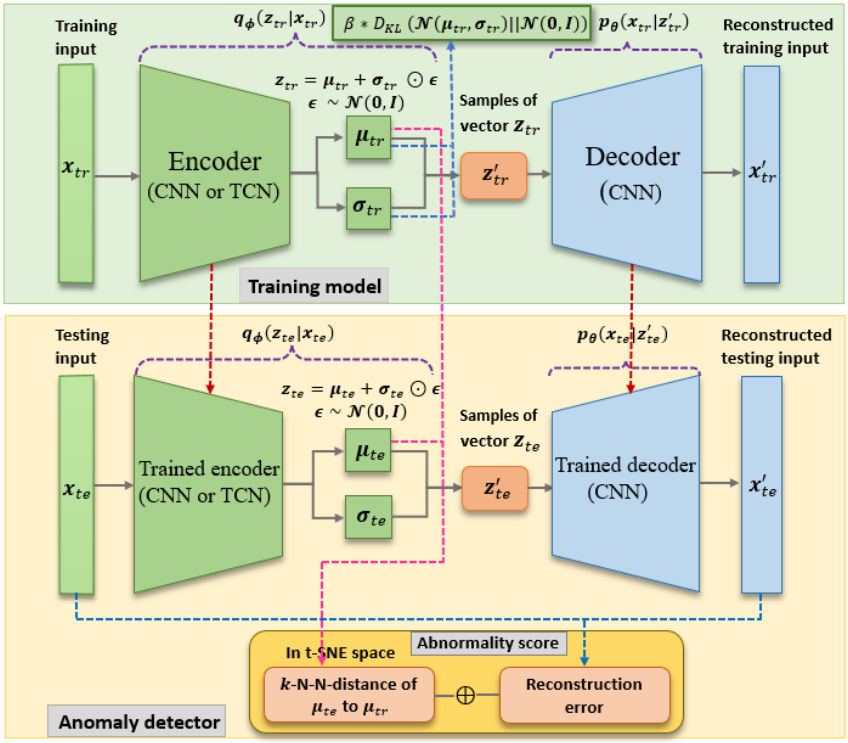


Figure 4.1: Architecture of AnoDM implemented by β-VAE and t-SNE for anomaly detection.

Algorithm 3: AnoDM Algorithm

Result: Anomaly scores of test samples

Inputs: $\boldsymbol{X}_{\text{tr}}$: training samples, $\boldsymbol{X}_{\text{te}}$: test samples, $\beta > 0$: hyperparameter for β-VAE

while *epoch no more than training iterations* **do**

  Encoder net maps $\boldsymbol{X}_{\text{tr}}$ into $\boldsymbol{\mu}_{\text{tr}}$ and $\boldsymbol{\sigma}_{\text{tr}}$;

  $\boldsymbol{Z}_{\text{tr}} = \boldsymbol{\mu}_{\text{tr}} + \boldsymbol{\sigma}_{\text{tr}} \odot \boldsymbol{\epsilon},\ \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathcal{I})$;

  Decoder net reconstructs $\boldsymbol{X}_{\text{tr}}$ to $\boldsymbol{X}'_{\text{tr}}$ using $\boldsymbol{Z}_{\text{tr}}$;

  Update β-VAE's parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$;

end

For $\boldsymbol{X}_{\text{tr}}$ and $\boldsymbol{X}_{\text{te}}$, obtain $\boldsymbol{\mu}_{\text{tr}}$ and $\boldsymbol{\mu}_{\text{te}}$ respectively using trained β-VAE;

Use t-SNE to map $\boldsymbol{\mu}_{\text{tr}}$ and $\boldsymbol{\mu}_{\text{te}}$ to 2D representations $\boldsymbol{l}_{\text{tr}}$ and $\boldsymbol{l}_{\text{te}}$;

for $\boldsymbol{x}_{te}^{(i)}$ *within* $\boldsymbol{X}_{te}$ **do**

  $\mathcal{D}_{\text{RE}}(\boldsymbol{x}_{\text{te}}^{(i)}) \triangleq \text{NSE}(\boldsymbol{x}_{\text{te}}^{(i)}, \boldsymbol{x}_{\text{te}}'^{(i)}) = \frac{\|\boldsymbol{x}_{\text{te}}^{(i)} - \boldsymbol{x}_{\text{te}}'^{(i)}\|_2^2}{\|\boldsymbol{x}_{\text{te}}^{(i)}\|_2}$; // reconstruction error

  $\mathcal{D}_{\text{tSNE}}^{k}(\boldsymbol{x}_{\text{te}}^{(i)}) \triangleq \frac{1}{k}\sum_{j \in N(i,k)} \|\boldsymbol{l}_{\text{te}}^{(i)} - \boldsymbol{l}_{\text{tr}}^{(j)}\|_2$; // $N(i,k)$ is the set of indices of $\boldsymbol{l}_{\text{te}}^{(i)}$'s k nearest neighbors from $\boldsymbol{l}_{\text{tr}}$

  $\mathcal{S}_{\beta\text{VAE+tSNE}}(\boldsymbol{x}_{\text{te}}^{(i)}) = \alpha\mathcal{D}_{\text{RE}} + (1-\alpha)\mathcal{D}_{\text{tSNE}}^{k}$; // $\alpha \in [0,1]$

end

4.1.1 Unsupervised Disentangled Representation Learning

The unsupervised disentangled representation learning component in the proposed architecture is implemented but not limited by β-VAE [28, 8, 36, 59]. A self-inclusive description of β-VAE is provided in Chapter 2.6. The objective function to be maximized for β-VAE is defined as [8]:

$$\mathcal{L} = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \beta|D_{\mathrm{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) - C|. \tag{4.1}$$

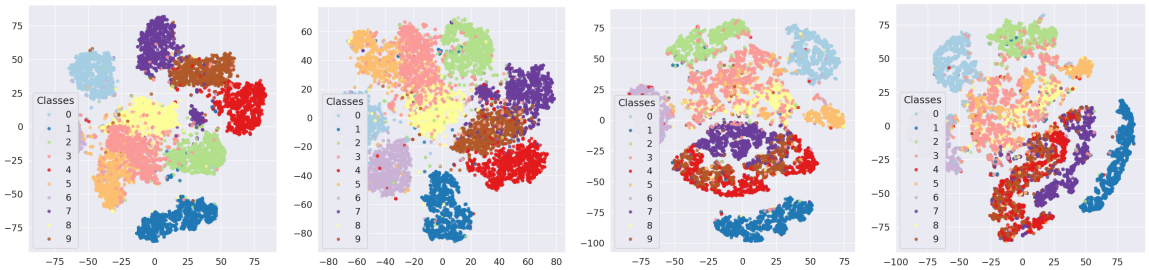
The first term of this objective function corresponds to reconstruction error in the raw feature space. The KL divergence characterizes the discrepancy between approximate posterior and isotropic prior of latent representations. A small discrepancy between them indicates high disentanglement of latent representations in independent variables. C is a hyperparameter which is used to improve the quality of reconstructed images. The loss function of the original β-VAE proposed in [28] does not have this hyperparameter. The value of β trades off reconstruction error and disentanglement. Unlike [28] and [8], I consider $\beta > 0$ rather than just $\beta > 1$, because it is unnecessary to bound the value of β by 1, $\beta > 0$ allows us search for a more appropriate disentanglement. It worth highlighting that other unsupervised disentanglement models can be used as well in AnoDM. For example, Mathieu et al. [59] interpreted disentanglement as decomposition instead of independence by adding an additional regularization term to reduce the discrepancy between the aggregate posterior and a desired structured prior. However, designing a properly structured prior could be practically challenging. The adoption of β-VAE in my proposed framework is sufficient to prove the concept that unsupervised disentanglement helps anomaly detection.

4.1.2 Effectivity of t-SNE Algorithm

In addition to β-VAE's reconstruction error, the average distance between a test sample and its k-nearest neighbors from the collection of training samples in the t-SNE map is used to score the outlierness of a test sample. As discussed in Chapter 2.7, t-SNE is significantly influenced by perplexity. The nature of complexity in data distributions makes it impossible to utilize a uniform criteria to define optimal perplexity for all data. Moreover, Wattenberg et al. [77] mentioned several weaknesses of t-SNE, for examples, (1) it naturally expands dense clusters and contracts sparse ones, evening out cluster sizes, and (2) distances between clusters might not reflect global geometry. However, it is likely that k-nearest neighbors still work for local clumps, because, with proper value of perplexity, local topological information of latent distributions can be preserved by the t-SNE plot. Thus, in t-SNE space, the measure of k-nearest neighbors is more suitable than full density estimation which is very sensitive with the sizes of clusters. Furthermore, as to be shown in Section 4.2, distancing in the 2D t-SNE map could be more robust than distancing in β-VAE's latent space where many non-determinative factors may influence the calculation of distances. Finally, it worth clarifying that a 2D latent representation from β-VAE is not directly learnt, because it will bottleneck too much the information flow for reconstruction. Instead, a lower-dimensional representation is learned through t-SNE for density estimation only.

4.1.3 Deterministic or Stochastic Latent Representations for t-SNE

Either the mean $\boldsymbol{\mu}$ or a sample $\boldsymbol{z}$ from the approximate inference distribution $q(\boldsymbol{z}|\boldsymbol{x})$ can be passed to t-SNE to calculate the k-NN distance of a test sample. There is trivial difference between performances achieved by these two methods in my framework. Generally, the $\boldsymbol{\mu}$-based method achieved slight better performance. The comparison of these two methods can be found in Table 4.1. Furthermore, from the latent representations' t-SNE maps (see Figure 4.2), one can interestingly see that when β is small (not overly large), the t-SNE maps for both methods are quite similar. As β becomes overly large, some normal classes can still form their own clusters (even though some similar classes, such as classes 3 and 8 in MNIST, tend to mingle together) in $\boldsymbol{\mu}$-based method, but in $\boldsymbol{z}$-based method all classes are entangled with each other. Same phenomena can be observed on the other datasets (see Figure 4.3 and 4.4). Therefore, the $\boldsymbol{\mu}$-based method is used in current design of AnoDM.



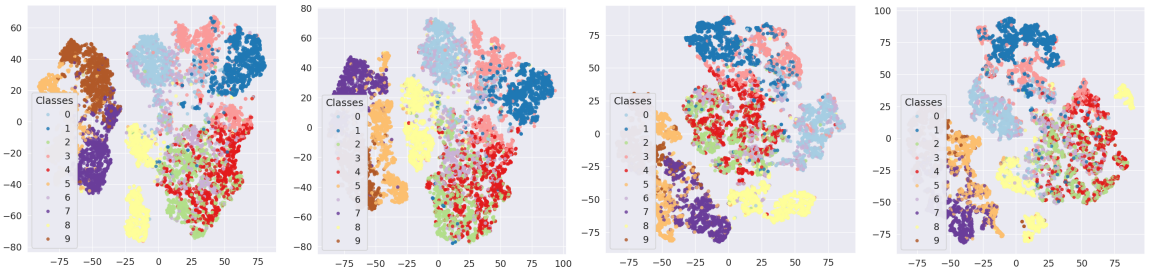
(a) $\beta = 0.01$ ($\boldsymbol{\mu}$-based) (b) $\beta = 0.8$ ($\boldsymbol{\mu}$-based) (c) $\beta = 5.0$ ($\boldsymbol{\mu}$-based) (d) $\beta = 10.0$ ($\boldsymbol{\mu}$-based)

(e) $\beta = 0.01$ ($\boldsymbol{z}$-based) (f) $\beta = 0.8$ ($\boldsymbol{z}$-based) (g) $\beta = 5.0$ ($\boldsymbol{z}$-based) (h) $\beta = 10.0$ ($\boldsymbol{z}$-based)

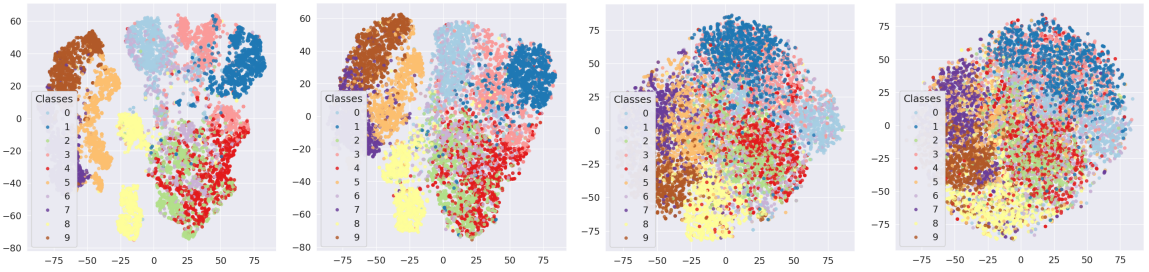
Figure 4.2: Comparison of $\boldsymbol{\mu}$-based method or $\boldsymbol{z}$-based method on MNIST data (anomalous class is 3) for inferring latent representations that are visualized in t-SNE map.

4.1.4 TCN Encoder for Unsupervised Sequence Modelling

Bai et al. [5] distilled superior design in convolutional network into a simple architecture and referred it as a temporal convolutional network (TCN) with two distinctive characteristics: (1) the convolutions in the architecture are causal, and (2) the architecture can take a sequence of any length and map it to an output vector of fixed length, just as with an RNN. They also explained that TCNs capture significantly longer history than recurrent networks. Inspired by [5], CNN is replaced with TCN in the encoder of β-VAE when evaluating the proposed AnoDM framework

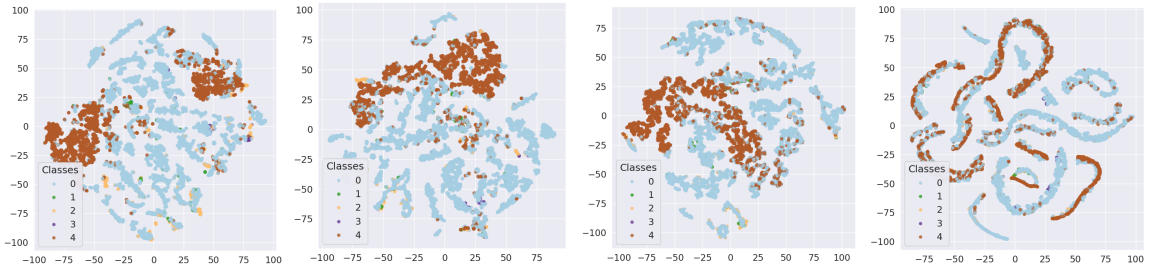


(a) $\beta = 0.01$ ($\boldsymbol{\mu}$-based) (b) $\beta = 0.8$ ($\boldsymbol{\mu}$-based) (c) $\beta = 5.0$ ($\boldsymbol{\mu}$-based) (d) $\beta = 10.0$ ($\boldsymbol{\mu}$-based)

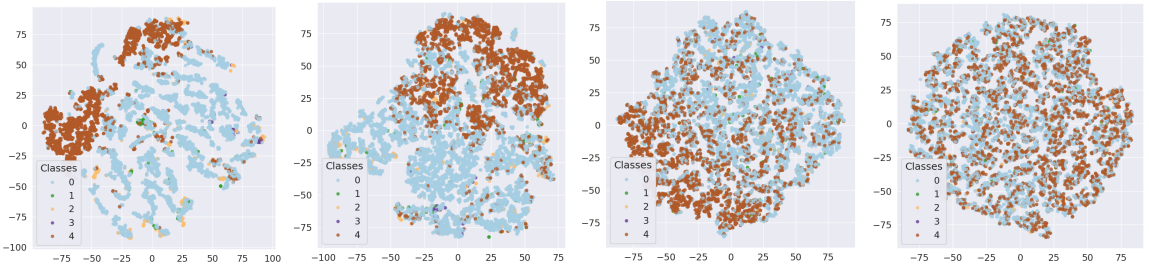


(e) $\beta = 0.01$ ($\boldsymbol{z}$-based) (f) $\beta = 0.8$ ($\boldsymbol{z}$-based) (g) $\beta = 5.0$ ($\boldsymbol{z}$-based) (h) $\beta = 10.0$ ($\boldsymbol{z}$-based)

Figure 4.3: Comparison of $\boldsymbol{\mu}$-based method or $\boldsymbol{z}$-based method on Fashion-MNIST data (anomalous class is 1 ("Trouser/pants")) for inferring latent representations that are visualized in t-SNE space.



(a) $\beta = 0.01$ ($\boldsymbol{\mu}$-based) (b) $\beta = 0.8$ ($\boldsymbol{\mu}$-based) (c) $\beta = 5.0$ ($\boldsymbol{\mu}$-based) (d) $\beta = 10.0$ ($\boldsymbol{\mu}$-based)



(e) $\beta = 0.01$ ($\boldsymbol{z}$-based) (f) $\beta = 0.8$ ($\boldsymbol{z}$-based) (g) $\beta = 5.0$ ($\boldsymbol{z}$-based) (h) $\beta = 10.0$ ($\boldsymbol{z}$-based)

Figure 4.4: Comparison of $\boldsymbol{\mu}$-based method or $\boldsymbol{z}$-based method on Arrhythmia time-series data (anomalous class is 4 ("Q")) for inferring latent representations that are visualized in t-SNE space.

on time-series data, while CNN is still used in the decoder, because the preliminary experiments demonstrated that keeping decoder as simpler CNN can help achieve comparable even better results, and take much less computing time. The architecture of TCN used in this thesis is depicted in Figure 4.5. In this TCN, kernel size is set to 4 and dilation factors to $[1, 2, 4, 8, 16, 32]$. In Section 4.2, the comparison among TCN, CNN, and LSTM encoders in the proposed framework also shows that TCN outperforms CNN and particularly LSTM to a great extent for ECG signal anomaly detection.

4.1.5 Anomaly Score Function in AnoDM

In the anomaly detector, the reconstruction error of a test sample in the original feature space and the average distance from its k-nearest-neighbors in training samples within the 2D t-SNE map are combined as a final anomaly score function:

$$\mathcal{S}_{\beta\text{VAE+tSNE}}(\boldsymbol{x}_{\text{te}}) = \alpha\mathcal{D}_{\text{RE}}(\boldsymbol{x}_{\text{te}}) + (1-\alpha)\mathcal{D}^{k}_{\text{tSNE}}(\boldsymbol{x}_{\text{te}}), \tag{4.2}$$

where the first term is defined using normalized squared error (NSE):

$$\mathcal{D}_{RE}(\boldsymbol{x}_{\text{te}}) \triangleq \text{NSE}(\boldsymbol{x}_{\text{te}}, \boldsymbol{x}'_{\text{te}}) = \frac{\|\boldsymbol{x}_{\text{te}} - \boldsymbol{x}'_{\text{te}}\|_2^2}{\|\boldsymbol{x}_{\text{te}}\|_2}, \tag{4.3}$$

where $\boldsymbol{x}_{\text{te}}$ is a test sample, and $\boldsymbol{x}'_{\text{te}}$ is its reconstructed version by sending the stochastic latent encoding through the decoder of β-VAE. The second term in Equation (4.2) is defined using β-VAE's deterministic latent encoding (mean from the encoder of learned β-VAE) as input to t-SNE:

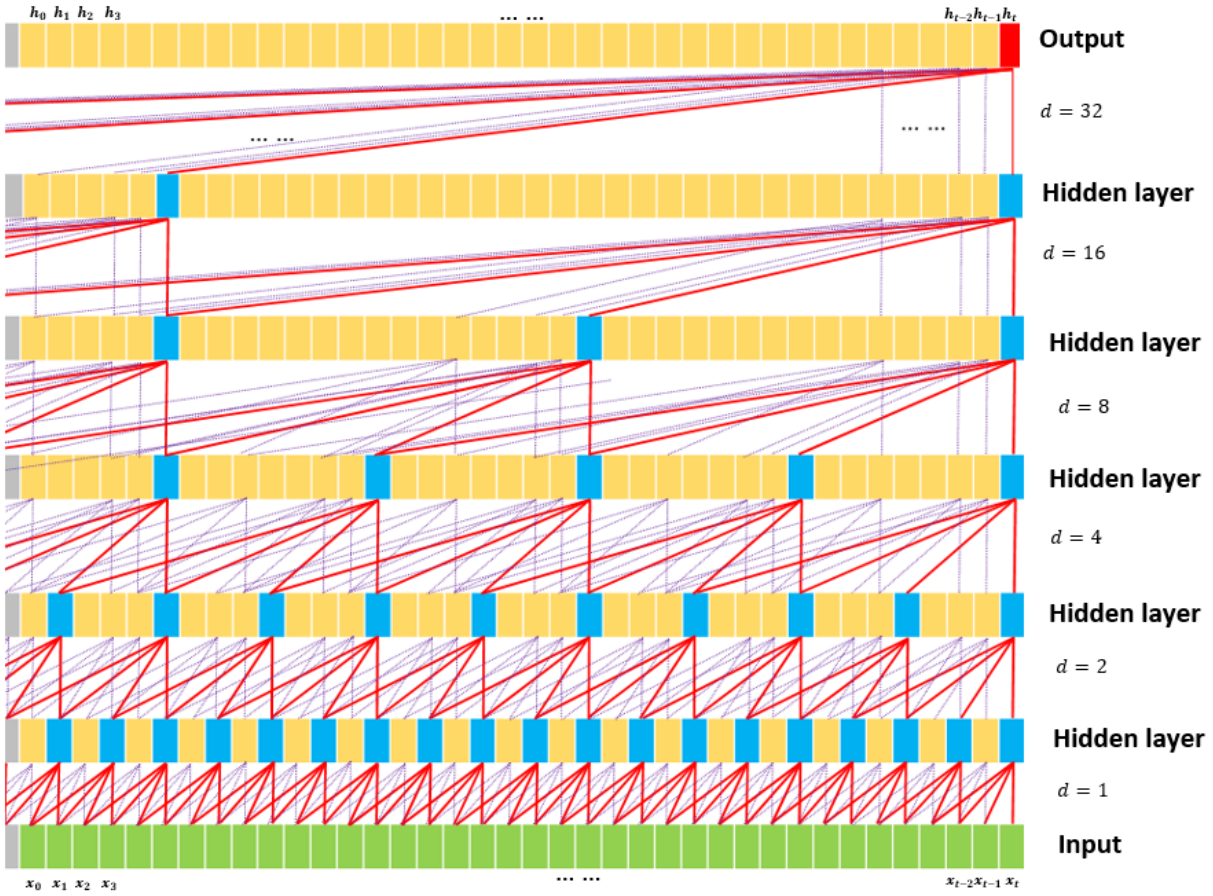
$$\mathcal{D}^{k}_{\text{tSNE}}(\boldsymbol{x}^{(i)}_{\text{te}}) \triangleq \frac{1}{k}\sum_{j\in N(i,k)} \|\boldsymbol{l}^{(i)}_{\text{te}} - \boldsymbol{l}^{(j)}_{\text{tr}}\|_2, \tag{4.4}$$

where $\boldsymbol{l}^{(i)}_{\text{te}}$ is the 2D representation of the i-th test sample in t-SNE map, $N(i,k)$ is the set of indices of $\boldsymbol{l}^{(i)}_{\text{te}}$'s k nearest neighbors from training samples' 2D representations $\boldsymbol{l}_{\text{tr}}$ in t-SNE map. In Equation (4.2), $\alpha \in [0, 1]$ is the combination hyperparameter such that the two terms can effectively complement each other. To allow the anomaly score function to achieve its full potential, α value should be sensitively searched, because the values of $\mathcal{D}_{\text{RE}}$ and $\mathcal{D}^{k}_{\text{tSNE}}$ can stay at different magnitudes, a very small change of α value may dramatically alter the contributions of these two terms.

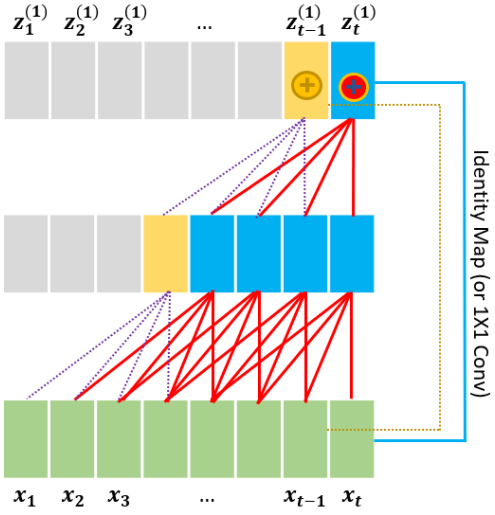
4.2 Experiments and Results

4.2.1 Datasets

- **MNIST** [50], **Fashion-MNIST** [78] and **Small-Norb** [51] (please see Section 3.2.2 for detailed explanation)



(a) A dilated causal convolution with dilation factors $d = [1, 2, 4, 8, 16, 32]$ and filter size $k = 4$.



(b) Residual block ($k = 4$, $d = 1$).

Figure 4.5: Architecture of the temporal convolutional network (TCN).

- **Arrhythmia**: It is derived from Physionet's MIT-BIH Arrhythmia Datase which consists of ECG recordings from 47 different subjects recorded at the sampling rate of $360 H_z$. This

dataset is created by five different heart beat categories: "N", "S", "V", "F" and "Q", in accordance with Association for the Advancement of Medical Instrumentation (AAMI) EC57 standard [39]. The meanings of these categories are explained below.

1. "N": normal, left or right bundle branch block, atrial escape and nodal escape.
2. "S": atrial premature, aberrant atrial premature, nodal premature and supra-ventricular premature.
3. "V": premature ventricular contraction and ventricular escape.
4. "F": fusion of ventricular and normal.
5. "Q": paced, fusion of paced and normal and unclassifiable.

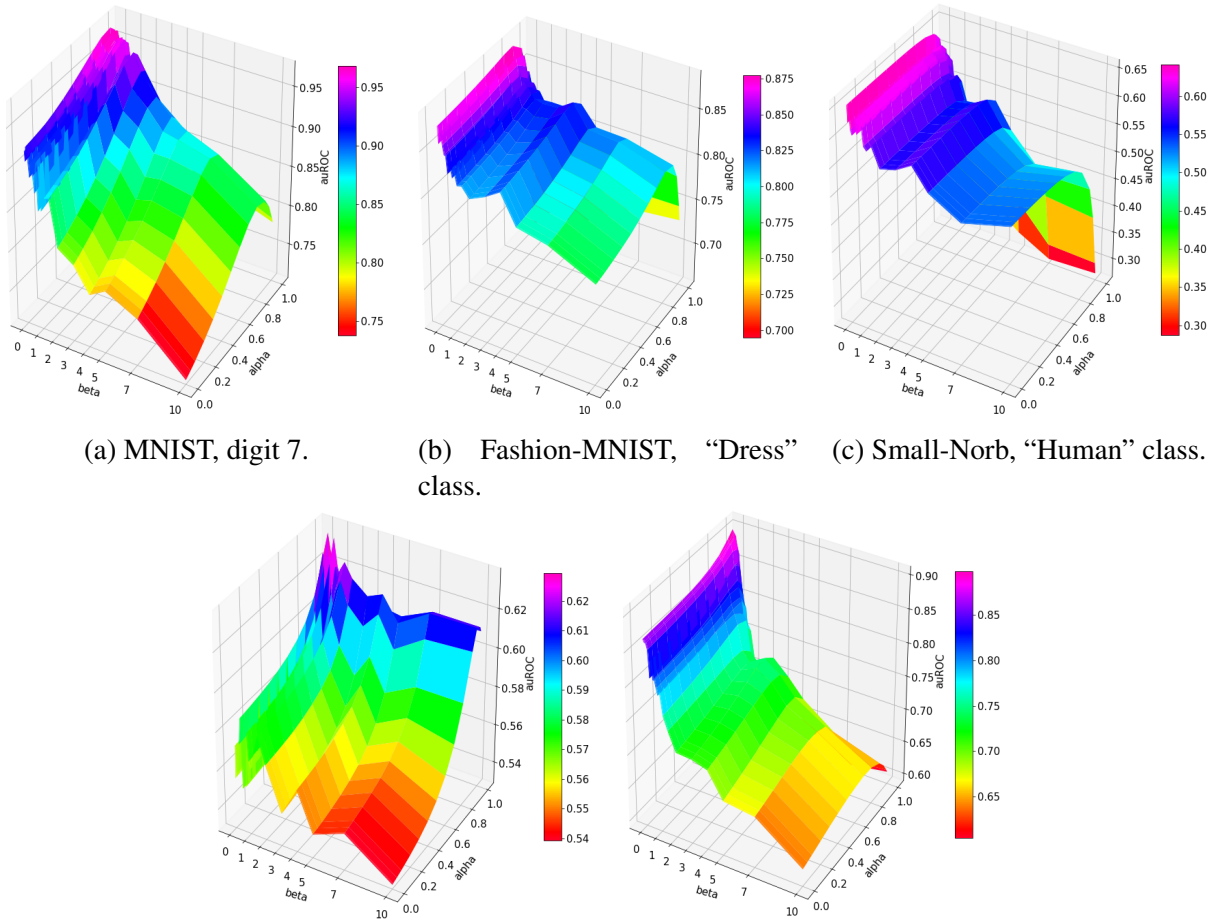
4.2.2 Experimental Setup

The detail of β-VAE architecture is given in Table 4.3. The same architecture was employed for experiments on MNIST and Fashion-MNIST datasets. In this architecture, decoder is the reverse of encoder with 3×32 2D convolutional kernels (stride 2). Both encoder and decoder in β-VAE architecture of CIFAR-10 dataset consisted of three convolution layers (with kernels: $3 \times 3 \times 64$ and $5 \times 5 \times 128$, and stride 2) and two fully connected layers followed by batch normalization. Dropout was set as 0.9 on output of fully connected layers. The architecture of β-VAE on Small-Norb dataset was quite similar to that on CIFAR-10 dataset, but with different convolutional kernels ($3 \times 3 \times 64$ and $3 \times 3 \times 128$) and dropout rate (0.7). For ECG (Arrhythmia) dataset, TCN architecture was applied in the encoder. The decoder part included 2 fully connected and 5 1D convolution layers. ReLU activation was used for deterministic units in both encoder and decoder networks on all β-VAE architectures. The number of epochs was set to 20 for experiments on MNIST and Fashion-MNIST datasets, and 50 for CIFAR-10, Small-Norb and Arrhythmia datasets; batch size was set to 100 and learning rate was chose to 0.001 for experiments on all these datasets. When using t-SNE, the dimension of t-SNE map was set as 2 for all datasets, perplexity 30, the learning rate 200, and maximum number of iterations 1000. The value of α in the anomaly score function is searched from set $\{0.0, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99, 1.0\}$. $k = 1$ is used for calculating k-nearest-neighbors distance in t-SNE map (k can also be set to 3 or 5, the results were similar). Algorithm 3 depicts the detailed process of calculating anomaly scores of test samples. β-VAE is firstly learnt through back propagating the reconstruction loss of training samples. The trained β-VAE is then employed to obtain latent embeddings of both training samples and a test sample. Once latent embeddings of both training samples and this test sample are obtained, these embeddings are further passed into 2-dimentional t-SNE space to calculate average distance between the 2D representation of the test sample and its k nearest neighbors from the 2D representations of training samples. Finally, this distance is combined with the reconstruction error of the test sample to define its anomaly score.

4.2.3 Computational Complexity

Both encoder and decoder nets consist of multiple convolutional layers in β-VAE, therefore the complexity of training a β-VAE will be same as the complexity of training a CNN, which is $O(m \times n \times w)$, where m is the number of training samples, n is the number of iterations and w is the number of total weights (in both encoder and decoder nets). T-SNE's computational and memory complexity is $O(m^2)$, therefore the total computational complexity of training β-VAE is $O(m^2 + m \times n \times w)$. Same as AnoCapsNet method, computation of anomaly scores of test samples is ignored in calculating complexity of AnoDM.

4.2.4 Impact of Beta to Performance



(a) MNIST, digit 7.

(b) Fashion-MNIST, "Dress" class.

(c) Small-Norb, "Human" class.

(d) CIFAR-10, "Dog" class.

(e) Arrhythmia, "F" class.

Figure 4.6: Performances (measured in terms of auROC) of AnoDM evaluated on five datasets: MNIST, Fashion-MNIST, Small-Norb, CIFAR-10 and Arrhythmia. On each dataset, the anomalous class is indicated in the corresponding subcaption while treating the rest classes as normal classes.

As discussed in [28], β (> 1) functions as a controller to encourage most efficient latent representation learning via limiting the capacity of latent information channel. Mathieu et al.

[59] however interpreted the objective of β as tuning a proper level of overlap of encodings by working with another term that regularizes the divergence of the aggregate posterior $q_\phi(\boldsymbol{z})$ and the desired prior $p(\boldsymbol{z})$. The main intuition is that purely increasing β induces too much overlap which actually discourages the disentanglement of data (information which is necessary for expressing desired structure is lost). Higgins et al. [28] demonstrated that β-VAE with $\beta > 1$ leads to interesting results when learning interpretable factorized latent representations on a variety of datasets. Surprisingly, my investigation demonstrates that by setting $0 < \beta < 1$, it actually achieved the state-of-the-art results for anomaly detection problems on a range of datasets, such as MNIST, Fashion-MNIST and Arrhythmia. Figure 4.6 illustrates the impact of values of β and α in the anomaly score function. Interestingly, best performances were achieved when $\beta < 1$ and the performances generally degrade as β increases, resonating with [59] that overly large values of β actually causes a mismatch between $q_\phi(\boldsymbol{z})$ and $p(\boldsymbol{z})$ (resulting in inappropriate level of overlap in the latent space). This phenomenon can be further seen in the t-SNE maps of latent embeddings in Figure 4.7. When β becomes extremely larger than the appropriate value, the anomalous class becomes entangled with its normal neighboring classes and the boundaries between normal classes become unclear. Theoretically, adding the divergence of the aggregate posterior $q_\phi(\boldsymbol{z})$ and the desired structured prior $p(\boldsymbol{z})$ is an effective way to limit the level of overlap when β is too large. However, it is practically challenging to design an appropriate structured prior. Therefore, in this thesis, the full range of β's value is explored in β-VAE for the impact of disentanglement to anomaly detection. Since this framework is quite general, it can be easily extended to other unsupervised disentangled representation learning models for anomaly detection.

4.2.5 Impact of Beta to t-SNE Representations

The t-SNE plots in Figure 4.7 reflect the impact of β's value on latent representations in the case of identifying anomalous digit 7 from MNIST. In this example, the best performance (auROC = 0.975) was achieved when $\beta = 0.01$. Clearly, as β increases, all latent clusters become less dense, and more anomalous latent data points move to neighboring clusters. Furthermore, Figure 4.7 also corroborates that, even though in t-SNE maps distances between clusters might not reflect global geometry and cluster sizes might not mirror the true sizes [77], using averaged distance from a test sample to its k nearest normal data points represented in t-SNE space to qualify outlierness still is a very effective way for distinguishing anomalous samples when β is tuned properly.

4.2.6 Evaluation of Anomaly Score Function

In order to better evaluate the anomaly score function, as formulated in Equation (4.2), a comprehensive comparison with methods only based on either distance in t-SNE map ($\mathcal{D}^k_{\text{tSNE}}$) or reconstruction-error in raw feature space ($\mathcal{D}_{\text{RE}}$) is conducted. To see the contribution of t-SNE, it is also compared with the method that calculates nearest neighbor distance directly in latent space of β-VAE. Figure 4.8 displays the ROC curves of these four approaches when assuming anomalous classes are respectively 1 ("Trouser/pants"), 3 ("Dress"), 5 ("Sandal"), and 7

(a) $\beta = 0.01$. (b) $\beta = 0.3$. (c) $\beta = 1.0$.

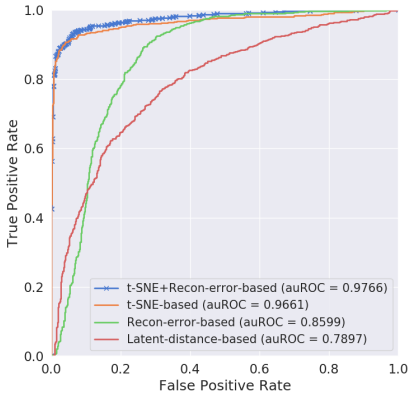
(d) $\beta = 2.0$. (e) $\beta = 4.0$. (f) $\beta = 10.0$.

Figure 4.7: The impact of β's value to t-SNE map of latent representations of MNIST samples. Class 7 is treated as anomalous class data. Each map displays 10000 data points identical in all maps including 5000 training data points and 5000 test data points.

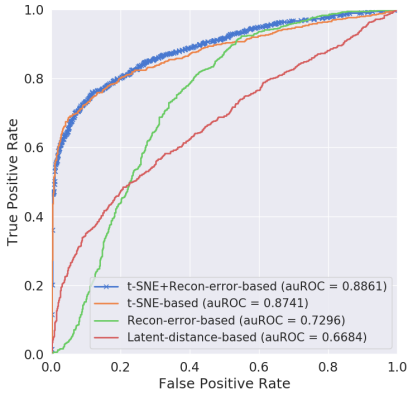
("Sneaker") on Fashion-MNIST. It is obvious that AnoDM achieves best results among them by taking advantages of both β-VAE reconstruction and t-SNE embedding. The β-VAE reconstruction reflects whether useful information is captured by the model through recovering the input $\boldsymbol{x}$; the t-SNE embedding indicates the disentanglement of latent representations $\boldsymbol{z}$. Both measures effectively complement each other. Besides, comparing the auROCs between t-SNE-based and latent-distance-based score functions, one can clearly see that the former dramatically outperforms the latter. Same conclusion can be drawn for MNIST, CIFAR-10, and Arrhythmia, as displayed in Figures 4.9, 4.11, 4.13, 4.14, and 4.15.

4.2.7 Comparison with CapsNet, GANs, and VAE

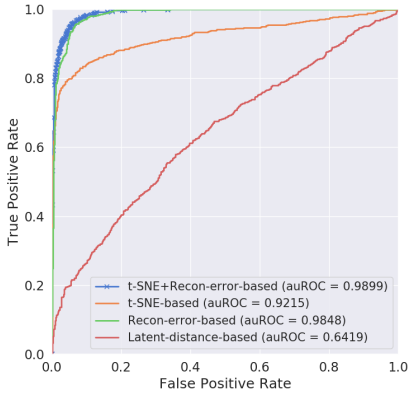
AnoDM is compared with state-of-the-art algorithms including a supervised method – AnoCapsNet, and two types of generative models – GANs (including AnoGAN and ADGAN) [13] and β-VAE (implemented by setting $\alpha = 1$ thus using only reconstruction error as anomaly score). As shown in Table 4.2, on average, AnoDM achieved excellent performance on MNIST and Fashion-MNIST in terms of auROC. On Fashion-MNIST, AnoCapsNet (prediction-probability-based), as the best benchmark method, obtained an average auROC of 0.833, while AnoDM successfully achieved 0.883 auROC. On MNIST, AnoDM accomplished auROC of 0.975, following AnoCapNet which performed the best (with averaged 0.981 auROC). As mentioned in Chapter 3, PP+RE-based AnoCapsNet had an impressive performance on CIFAR-10. The aver-



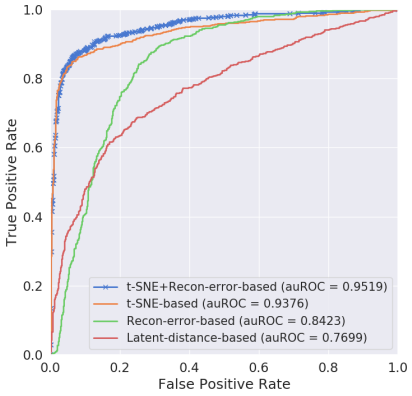
(a) Anomalous class: 1.



(b) Anomalous class: 3.



(c) Anomalous class: 5.



(d) Anomalous class: 7.

Figure 4.8: ROC curves of four β-VAE based methods on Fashion-MNIST. These four examples illustrate the results of using four different anomaly score functions: t-SNE+Recon-error-based ($S_{\beta\text{VAE+tSNE}} = \alpha\mathcal{D}_{\text{RE}} + (1-\alpha)\mathcal{D}^k_{\text{tSNE}}$), t-SNE-based ($\mathcal{D}^k_{\text{tSNE}}$), Reconstruction-error-based ($\mathcal{D}_{\text{RE}}$) and latent-distance-based (calculating distances in latent space of β-VAE). The α values for these four plots are 0.8, 0.8, 0.95, and 0.8, respectively.

aged performance of AnoCapsNet (0.677 auROC) scored around 4% higher than AnoDM method (the best framework among four generative methods: AnoGAN, ADGAN, β-VAE and AnoDM). However, AnoCapsNet is a supervised method that take advantages of class information, while AnoDM is completely unsupervised that is more suitable in many practices where class information is either incomplete or unavailable. Moreover, all generative models did not work well on Small-Norb, mainly because these models used convolution to extract features from image, but convolution is only able to capture translation but not other affine transformations. AnoCapsNet learns these transformations as a supervised method and achieved good averaged auROC of 0.703.

Furthermore, by comparing AnoDM with β-VAE that only considers reconstruction error as anomaly score, AnoDM dramatically improved the performance in all cases. In other words, t-SNE makes a prominent contribution to improve β-VAE for anomaly detection problems.

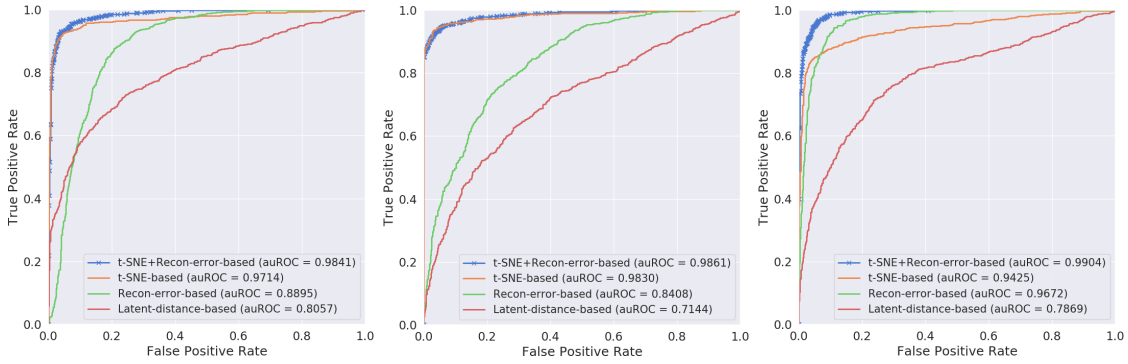
4.2.8 AnoDM for Time-Series

As mentioned in Section 4.1, the proposed AnoDM framework uses a TCN encoder in β-VAE for time-series anomaly detection. Figure 4.16 displays the comparison among TCN, CNN and LSTM encoders in the AnoDM framework on Arrhythmia. For the five classes in Arrhythmia, iteratively one class was treated as anomalous class, while the other classes were used as normal classes. The TCN-encoder-based method outperforms the other two methods significantly in all five cases. Even though the CNN encoder achieved impressive results when detecting anomalous class "S", "V", "F" and "Q" respectively, it did not work quite well when class "N" was treated as anomaly. One possible reason might be that comparing with TCN and LSTM, the performance of CNN is more sensitive on the training sample size. Taking the above case as an example, as class 0 ("N") accounts for over 80% of training data, when considering it as anomaly, normal training data hence become less sufficient for learning β-VAE. Nevertheless, in TCN-based β-VAE, each hidden unit of the last deterministic hidden layer before latent encoding at the bottleneck is calculated based on much longer sequence dependency, such that it is less sensitive to the limitation of small sample size. Conclusively, as mentioned in [5], TCN should be regarded as a natural starting point for sequence modeling tasks.

As a case study, Figure 4.17a shows the original EGG signals and reconstructed signals by TCN-based β-VAE when considering class "Q" as anomaly. Normal samples can be reconstructed very well, whereas abnormal samples suffer from larger reconstruction errors. Meanwhile, the corresponding t-SNE plot in Figure 4.17b displays two distinctive clusters of abnormal samples. The combination of these two measures thus leads to the best performance as seen in Figure 4.17c.

Table 4.1: AuROCs of AnoDM for both μ-based and z-based techniques.

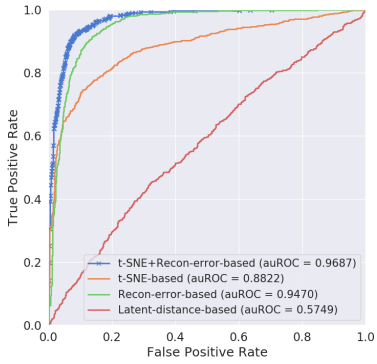
Dataset	Class	μ-based			z-based		
		auROC	β	α	**auROC**	β	α
MNIST	0	0.984	0.01	0.8	**0.985**	0.8	0.1
	1	0.986	0.01	0.6	**0.987**	0.01	0.6
	2	0.990	0.4	0.9	**0.991**	0.2	0.9
	3	**0.969**	0.05	0.9	0.968	0.05	0.9
	4	0.974	0.01	0.95	**0.975**	0.1	0.95
	5	0.975	0.01	0.95	**0.976**	0.01	0.95
	6	**0.983**	0.01	0.8	0.980	0.01	0.8
	7	0.975	0.01	0.9	**0.977**	0.01	0.9
	8	0.980	0.01	0.9	**0.982**	0.01	0.9
	9	**0.928**	0.4	0.8	0.925	0.05	0.8
	avg.	0.974	–	–	**0.975**	-	–
Fashion-MNIST	0	**0.844**	0.05	0.2	0.840	0.05	0.0
	1	0.977	0.01	0.8	**0.978**	0.01	0.8
	2	**0.783**	0.01	0.0	**0.783**	0.05	0.0
	3	**0.886**	0.01	0.8	0.884	0.01	0.8
	4	0.760	0.1	0.0	**0.763**	0.3	0.0
	5	**0.990**	0.2	0.95	**0.990**	0.2	0.95
	6	**0.713**	0.05	0.0	0.709	0.05	0.0
	7	**0.952**	0.01	0.8	0.950	0.01	0.8
	8	**0.980**	0.05	0.8	**0.980**	0.05	0.8
	9	0.940	0.3	0.8	**0.944**	0.3	0.7
	avg.	**0.883**	–	–	0.882	-	–
CIFAR-10	0	**0.635**	0.4	0.0	0.634	0.2	0.0
	1	0.752	0.6	0.99	**0.754**	0.05	0.99
	2	**0.589**	1.0	0.0	0.558	0.7	0.0
	3	**0.608**	10.0	0.95	0.606	10.0	0.99
	4	**0.564**	0.01	0.0	0.563	0.4	0.0
	5	**0.638**	0.4	0.95	0.627	0.7	0.95
	6	**0.600**	0.3	0.0	0.590	0.7	0.0
	7	**0.648**	0.1	0.95	0.644	0.01	0.95
	8	**0.642**	0.3	0.0	0.624	1.2	0.0
	9	0.717	1.0	0.95	**0.718**	0.01	0.95
	avg.	**0.640**	–	–	0.632	-	–
Small-Norb	0	0.512	0.1	0.0	**0.520**	7.0	0.0
	1	**0.656**	0.01	0.0	0.647	0.01	0.0
	2	**0.771**	0.05	1.0	**0.771**	0.05	1.0
	3	**0.581**	10.0	1.0	**0.581**	10.0	1.0
	4	**0.564**	0.5	1.0	**0.564**	0.5	1.0
	avg.	**0.617**	–	–	**0.617**	-	–
ECG (Arrhythmia)	0	**0.911**	0.05	0.95	0.906	0.01	0.95
	1	0.924	0.01	0.95	**0.925**	0.01	0.95
	2	**0.970**	0.01	0.99	**0.970**	0.01	0.99
	3	0.905	0.01	0.95	**0.910**	0.01	0.95
	4	**0.991**	0.01	0.99	**0.991**	0.01	0.99
	avg.	**0.940**	–	–	**0.940**	-	–



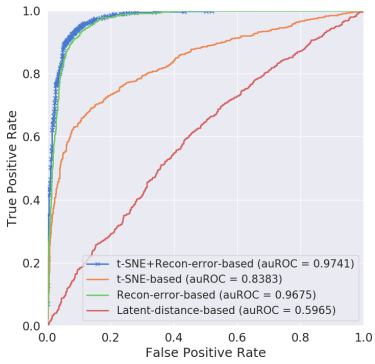
(a) digit 0 (as anoamly)

(b) digit 1 (as anoamly)

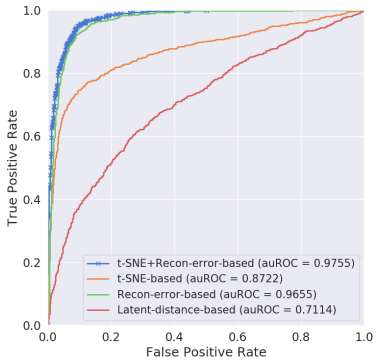
(c) digit 2 (as anoamly)



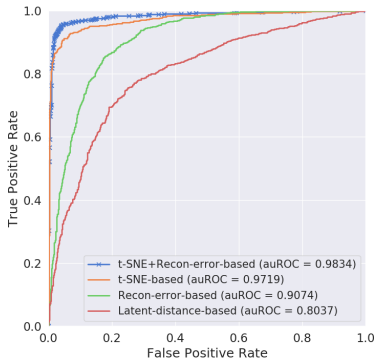
(d) digit 3 (as anoamly)



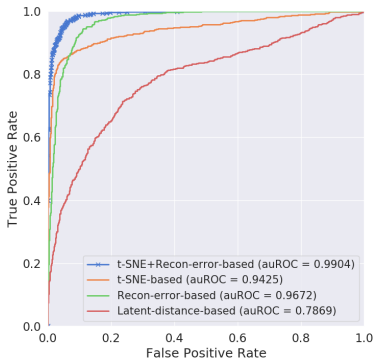
(e) digit 4 (as anoamly)



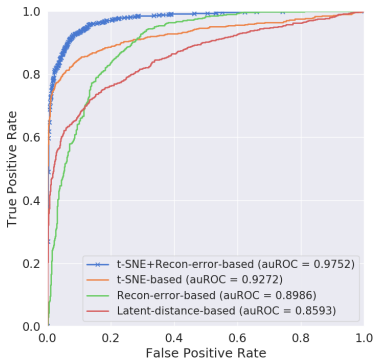
(f) digit 5 (as anoamly)



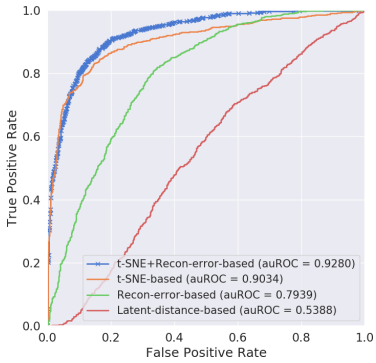
(g) digit 6 (as anoamly)



(h) digit 7 (as anoamly)

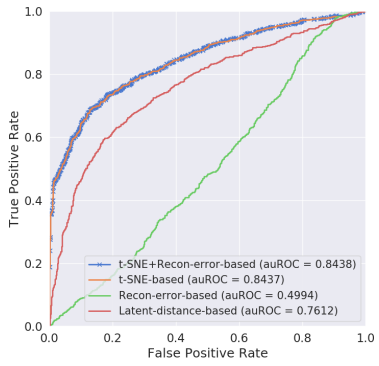


(i) digit 8 (as anoamly)

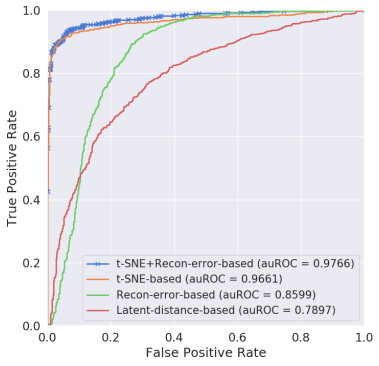


(j) digit 9 (as anoamly)

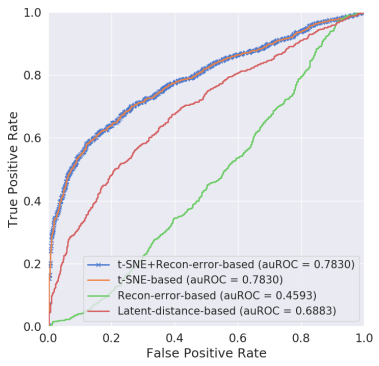
Figure 4.9: ROC plots of AnoDM on MNIST (μ-based).



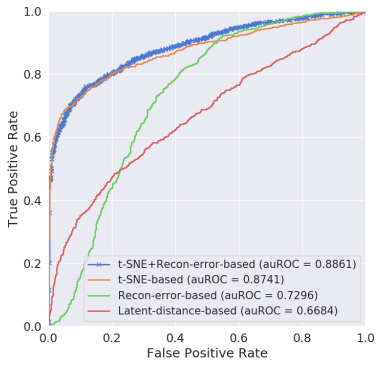
(a) “T-shirt/top” (as anoamly)



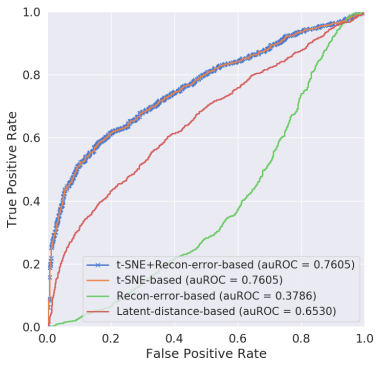
(b) “Trouser” (as anoamly)



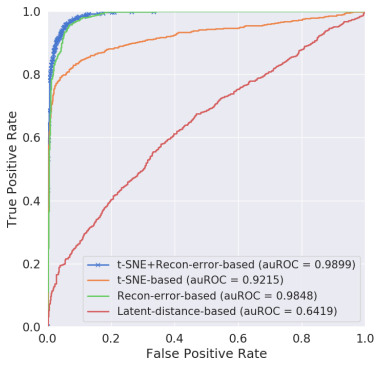
(c) “Pullover” (as anoamly)



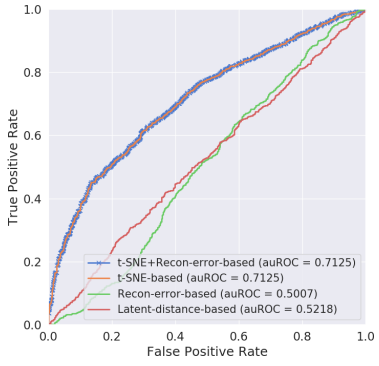
(d) “Dress” (as anoamly)



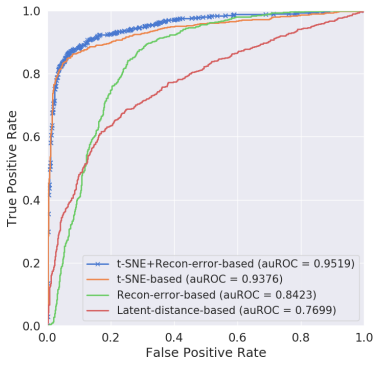
(e) “Coat” (as anoamly)



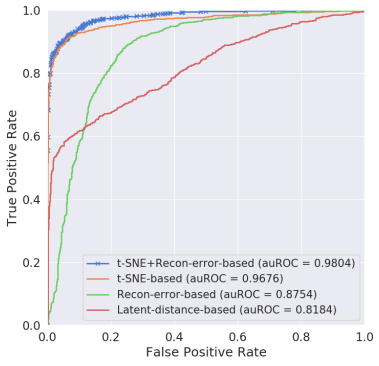
(f) “Sandal” (as anoamly)



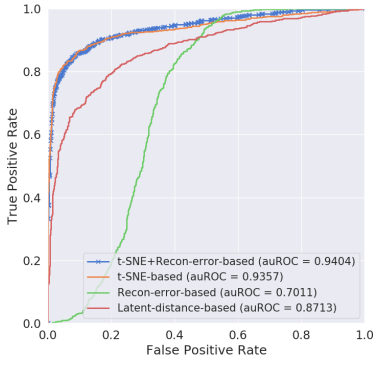
(g) “Shirt” (as anoamly)



(h) “Sneaker” (as anoamly)

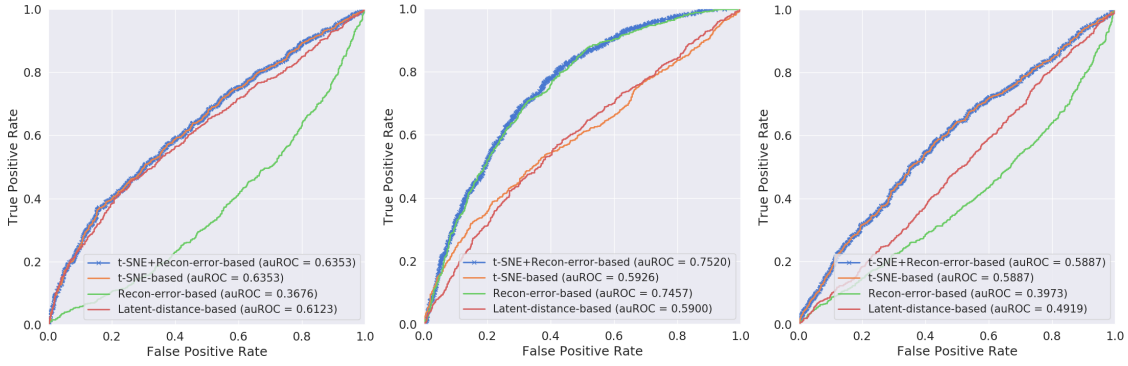


(i) “Bag” (as anoamly)



(j) “Ankle boot” (as anoamly)

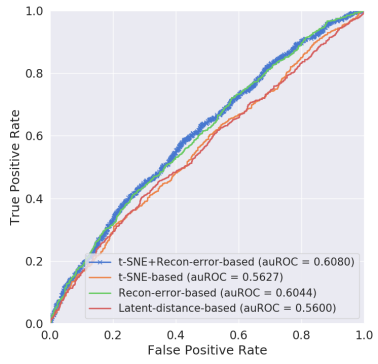
Figure 4.10: ROC plots of AnoDM on Fashion-MNIST (μ-based).



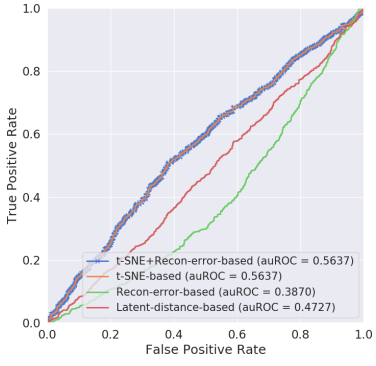
(a) “Airplane” (as anoamly)

(b) “Automobile” (as anoamly)

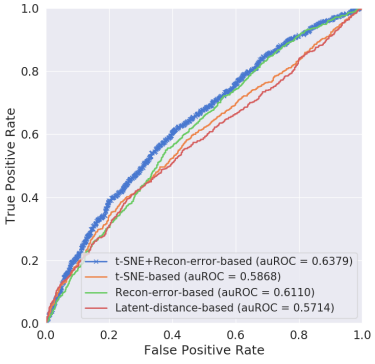
(c) “Bird” (as anoamly)



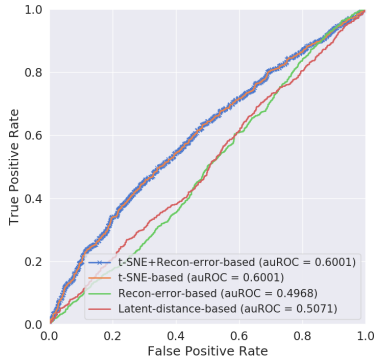
(d) “Cat” (as anoamly)



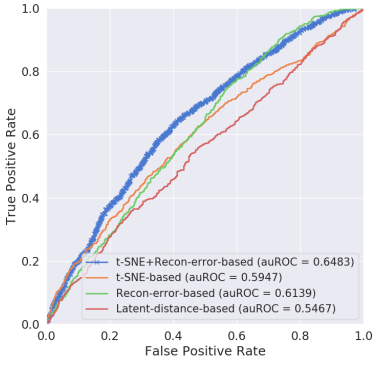
(e) “Deer” (as anoamly)



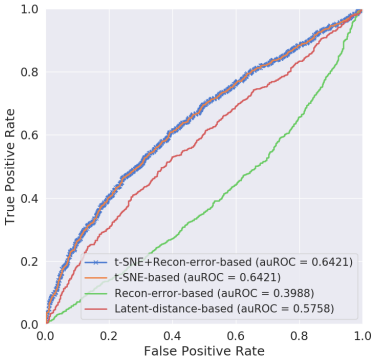
(f) “Dog” (as anoamly)



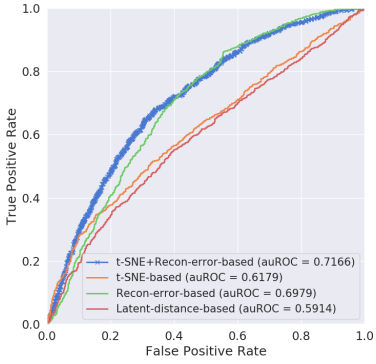
(g) “Frog” (as anoamly)



(h) “Horse” (as anoamly)

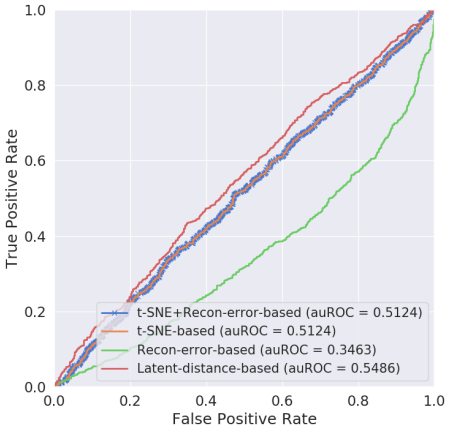


(i) “Ship” (as anoamly)

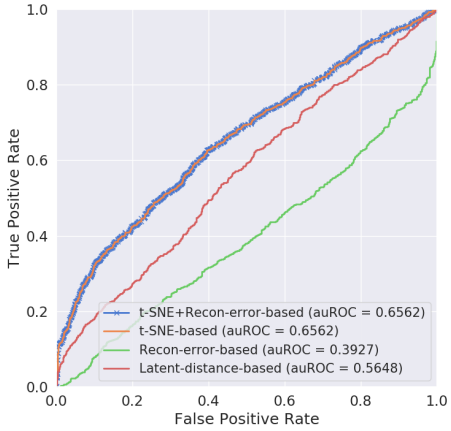


(j) “Truck” (as anoamly)

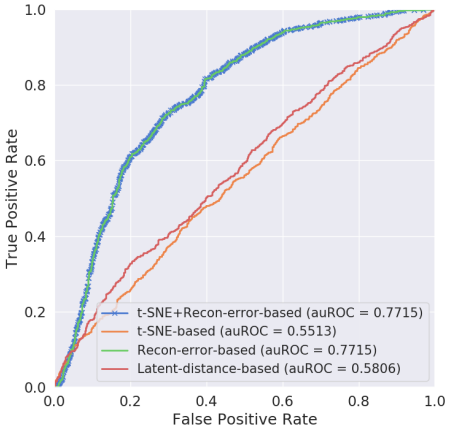
Figure 4.11: AuROC plots of AnoDM on CIFAR-10 (μ-based).



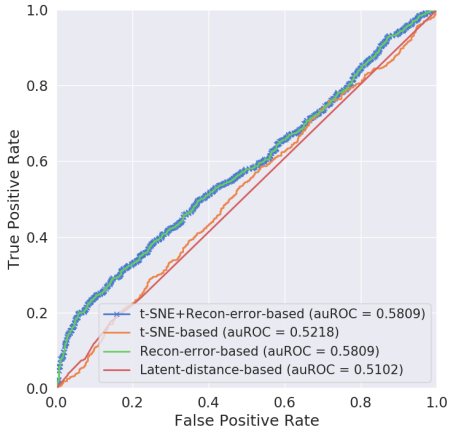
(a) “Four-legged animals” (as anoamly)



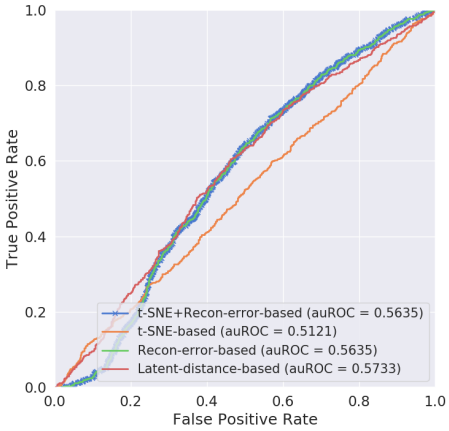
(b) “Human figures” (as anoamly)



(c) “Airplanes” (as anoamly)

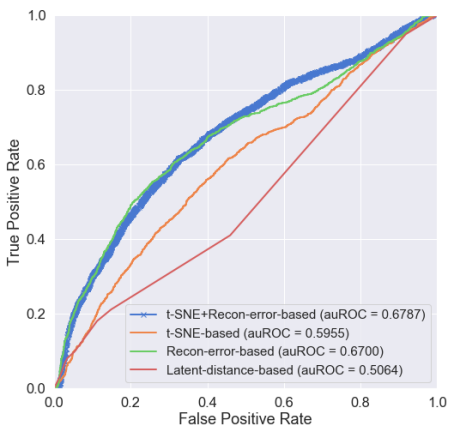


(d) “Trucks” (as anoamly)

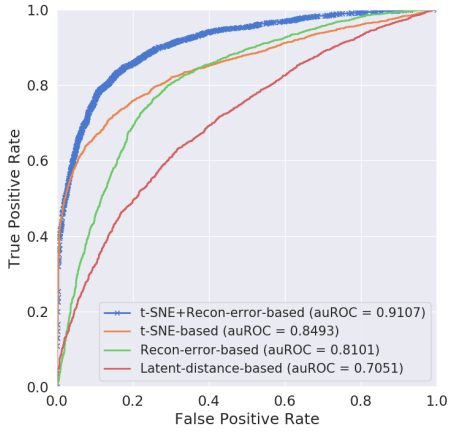


(e) “Cars” (as anoamly)

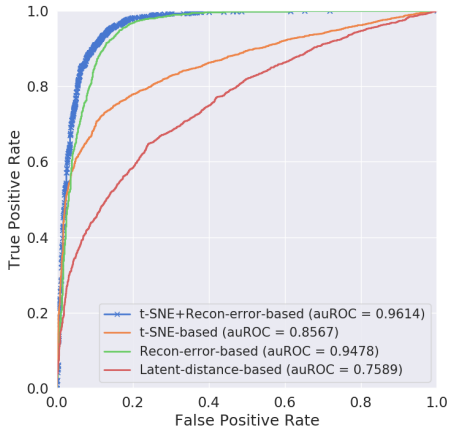
Figure 4.12: ROC plots of AnoDM on Small-Norb (μ-based).



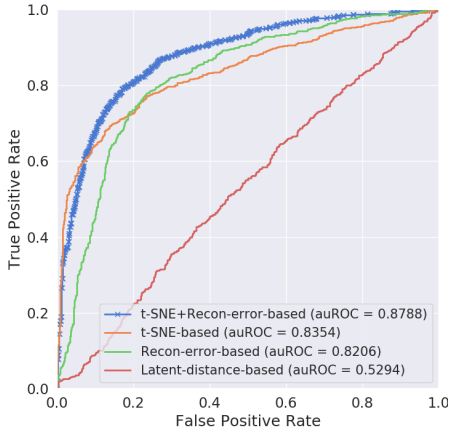
(a) “N” (as anoamly)



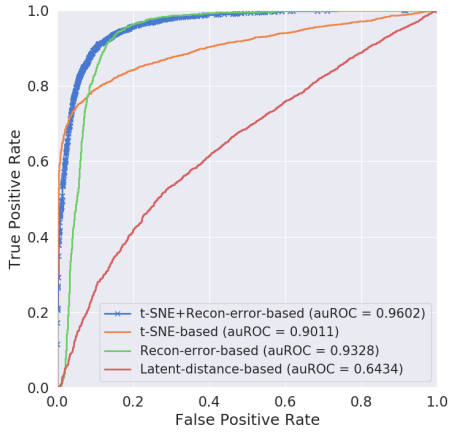
(b) “S” (as anoamly)



(c) “V” (as anoamly)

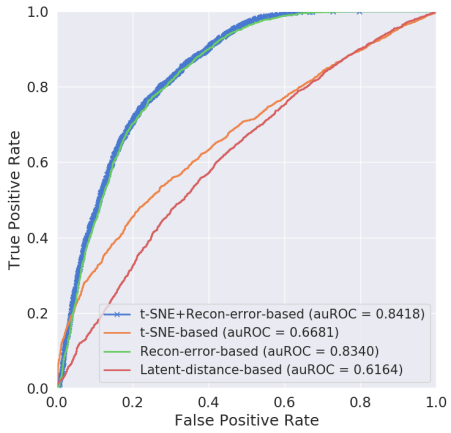


(d) “F” (as anoamly)

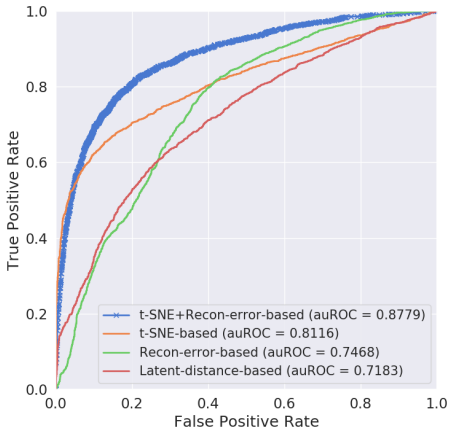


(e) “Q” (as anoamly)

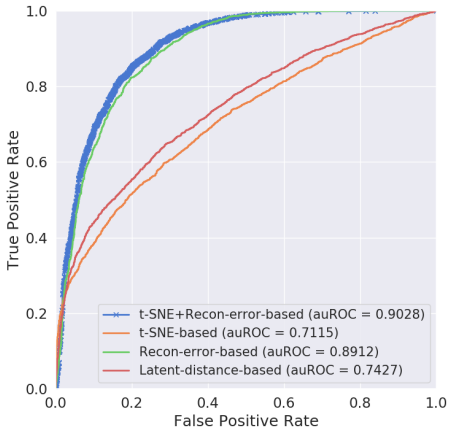
Figure 4.13: ROC plots of AnoDM on Arrhythmia using CNN-encoder (μ-based).



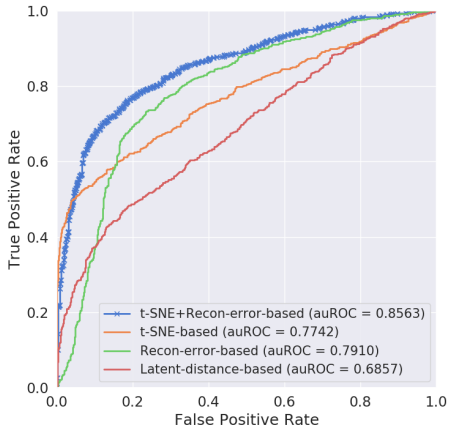
(a) “N” (as anoamly)



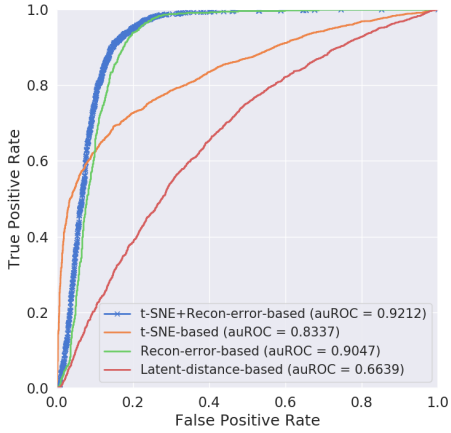
(b) “S” (as anoamly)



(c) “V” (as anoamly)

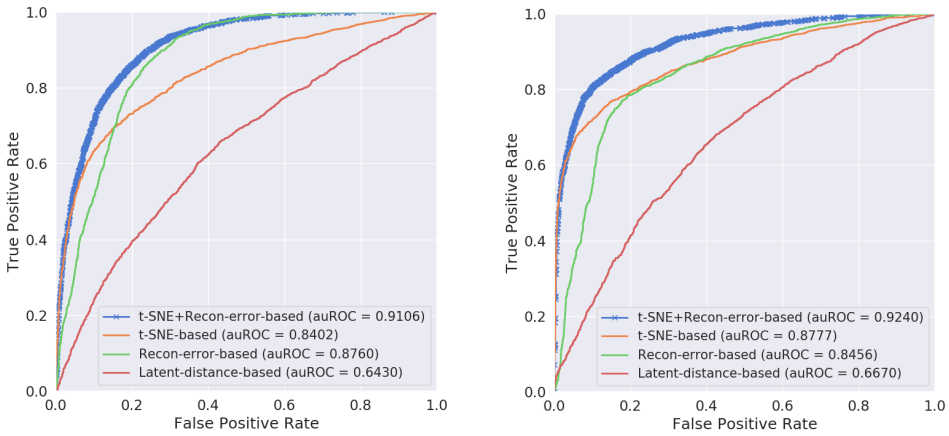


(d) “F” (as anoamly)



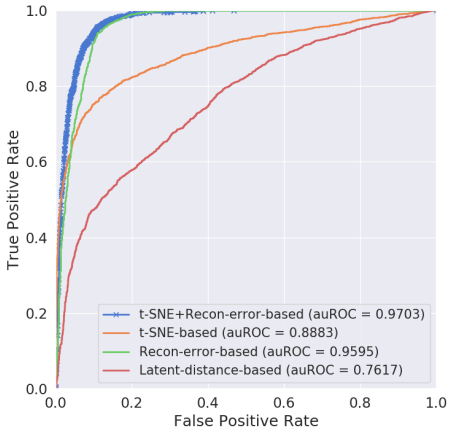
(e) “Q” (as anoamly)

Figure 4.14: ROC plots of AnoDM on Arrhythmia using LSTM-encoder (μ-based).

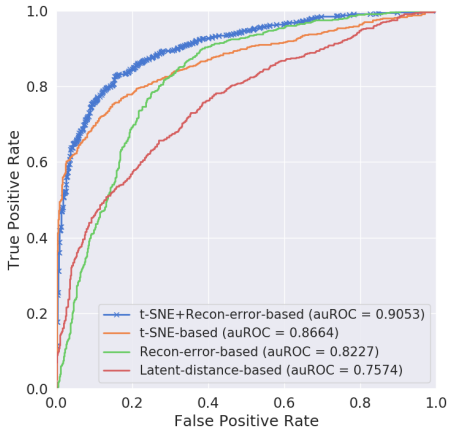


(a) “N” (as anoamly)

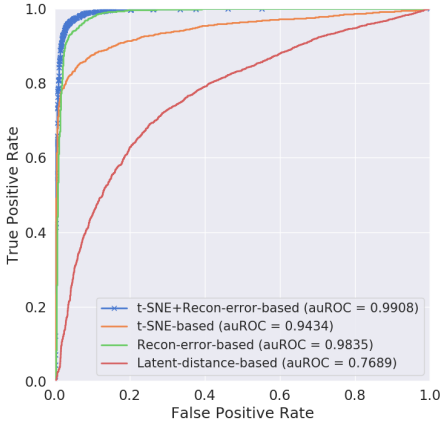
(b) “S” (as anoamly)



(c) “V” (as anoamly)



(d) “F” (as anoamly)



(e) “Q” (as anoamly)

Figure 4.15: ROC plots of AnoDM on Arrhythmia using TCN-encoder (μ-based).

Table 4.2: Performance of AnoDM in comparison with other methods in terms of auROC on image data. The results for AnoDM were obtained by either the $\boldsymbol{\mu}$-based or $\boldsymbol{z}$-based algorithm. In the column for β-VAE, only reconstruction error is used as anomaly score. The results for AnoGAN and ADGAN were obtained from [13].

Dataset	Class	AnoCapsNet (PP+RE)	AnoGAN	ADGAN	β-VAE	AnoDM
MNIST	0	0.997	0.990	**0.999**	0.890	0.985
	1	0.990	**0.998**	0.992	0.841	0.987
	2	**0.991**	0.888	0.968	0.967	**0.991**
	3	**0.984**	0.913	0.953	0.947	0.969
	4	0.939	0.944	0.960	0.968	**0.975**
	5	**0.983**	0.912	0.955	0.966	0.976
	6	0.959	0.925	0.980	0.907	**0.983**
	7	**0.987**	0.964	0.950	0.899	0.977
	8	**0.993**	0.883	0.959	0.946	0.982
	9	**0.990**	0.958	0.965	0.794	0.928
	avg.	**0.981**	0.937	0.968	0.913	0.975
Fashion-MNIST	0	0.620	–	–	0.500	**0.844**
	1	0.915	–	–	0.860	**0.978**
	2	**0.818**	–	–	0.459	0.783
	3	**0.898**	–	–	0.730	0.886
	4	**0.790**	–	–	0.379	0.763
	5	0.982	–	–	0.985	**0.990**
	6	**0.801**	–	–	0.501	0.713
	7	0.787	–	–	0.842	**0.952**
	8	0.960	–	–	0.876	**0.980**
	9	0.754	–	–	0.701	**0.944**
	avg.	0.833	–	–	0.683	**0.883**
Small-Norb	0	**0.785**	–	–	0.346	0.520
	1	0.598	–	–	0.393	**0.656**
	2	**0.812**	–	–	0.772	0.772
	3	**0.605**	–	–	0.581	0.581
	4	**0.714**	–	–	0.564	0.564
	avg.	**0.703**	–	–	0.531	0.619
CIFAR-10	0	0.645	0.610	**0.661**	0.368	0.635
	1	0.736	0.565	0.435	0.746	**0.754**
	2	0.646	**0.648**	0.636	0.397	0.589
	3	**0.686**	0.528	0.488	0.604	0.608
	4	0.670	0.670	**0.794**	0.387	0.564
	5	**0.669**	0.592	0.640	0.611	0.638
	6	**0.723**	0.625	0.685	0.500	0.600
	7	**0.733**	0.576	0.559	0.614	0.648
	8	0.477	0.723	**0.798**	0.399	0.642
	9	**0.788**	0.582	0.643	0.698	0.718
	avg.	**0.677**	0.612	0.634	0.532	0.640

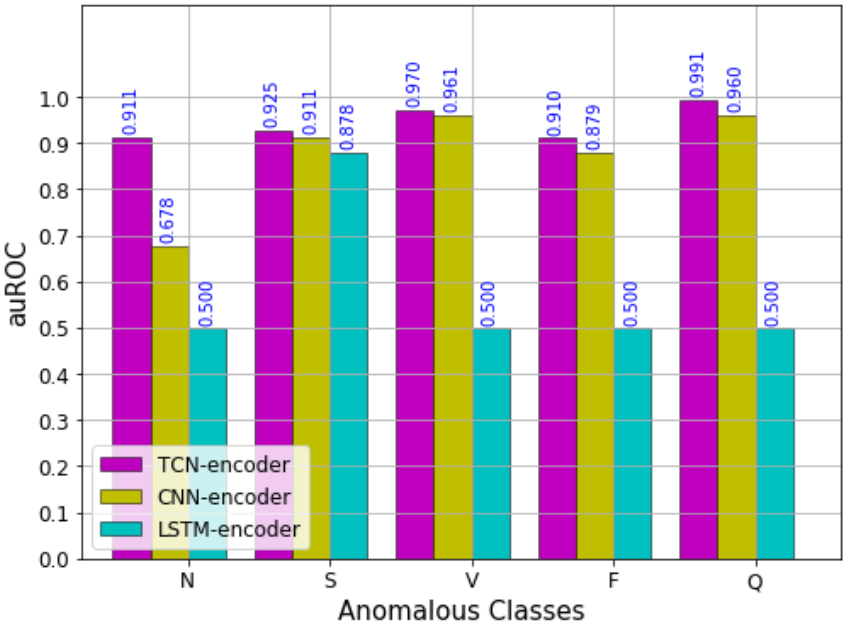
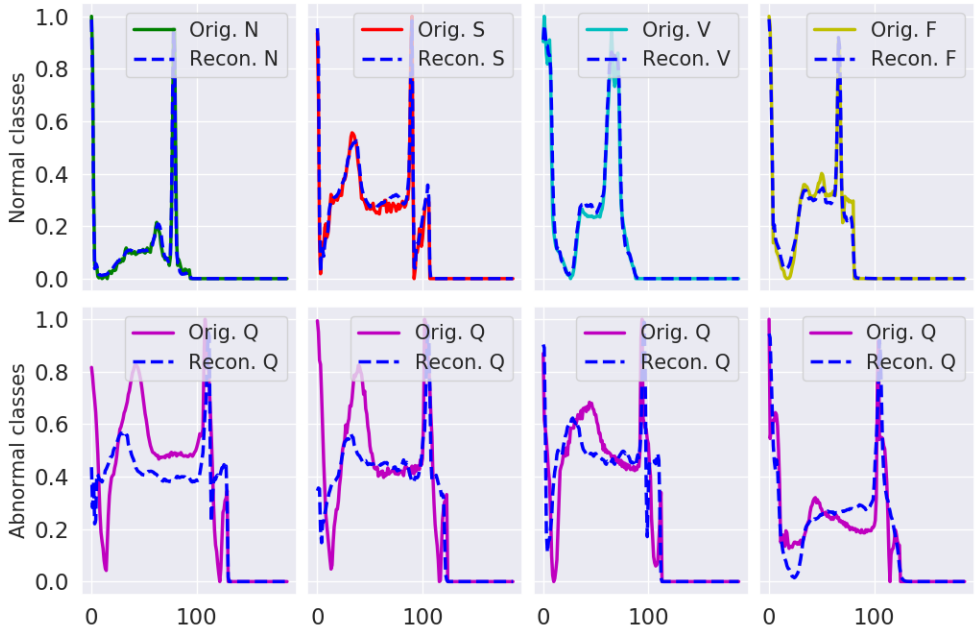
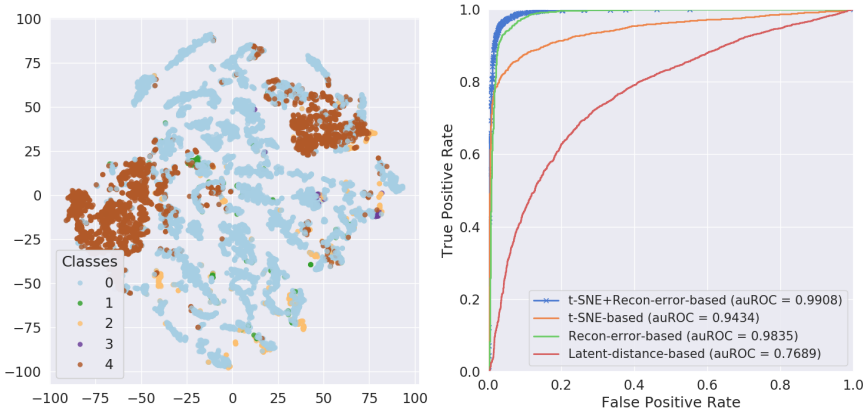


Figure 4.16: AuROCs of AnoDM on Arrhythmia.



(a) Reconstructed signals by β-VAE.



(b) t-SNE plot. (c) ROC plot.

Figure 4.17: Reconstructed signals, t-SNE map, and ROC curves on Arrhythmia with class "Q" as anomaly.

Table 4.3: Architecture of the β-VAE used in AnoDM.

Dataset	Optimizer		Architecture
MNIST	Adam ($1e^{-3}$)	Input	784 (flattened $28 \times 28 \times 1$).
		Encoder	Conv2d (filters 32, kernel_size 3, strides 2), Conv2d (filters 32, kernel_size 3, strides 2), FC (128), FC (128). ReLU activation.
		Decoder	FC (128), FC (1568), Deconv2d (filters 32, kernel_size 3, strides 2), Deconv2d (filters 1, kernel_size 3, strides 2). ReLU activation.
Fashion-MNIST	Adam ($1e^{-3}$)	Input	784 (flattened $28 \times 28 \times 1$).
		Encoder	Conv2d (filters 32, kernel_size 3, strides 2), Conv2d (filters 32, kernel_size 3, strides 2), FC (128), FC (128). ReLU activation.
		Decoder	FC (128), FC (1568), Deconv2d (filters 32, kernel_size 3, strides 2), Deconv2d (filters 1, kernel_size 3, strides 2). ReLU activation.
CIFAR-10	Adam ($1e^{-3}$)	Input	3072 (flattened $32 \times 32 \times 3$).
		Encoder	Conv2d (filters 64, kernel_size 3, strides 2), BN, Conv2d (filters 128, kernel_size 5, strides 2), BN, Conv2d (filters 128, kernel_size 5, strides 2), BN, FC (256), Dropout (0.9), BN, FC (256), Dropout (0.9), BN. ReLU activation.
		Decoder	FC (256), BN, FC (256), BN, FC (2048), BN, Deconv2d (filters 128, kernel_size 5, strides 2), BN, Deconv2d (filters 64, kernel_size 5, strides 2), BN, Deconv2d (filters 3, kernel_size 3, strides 2). ReLU activation.
Small-Norb	Adam ($1e^{-3}$)	Input	1024 (flattened $32 \times 32 \times 1$).
		Encoder	Conv2d (filters 32, kernel_size 3, strides 2), BN, Conv2d (filters 64, kernel_size 3, strides 2), BN, Conv2d (filters 128, kernel_size 3, strides 2), BN, FC (256), Dropout (0.7), BN, FC (256), Dropout (0.7), BN. ReLU activation.
		Decoder	FC (256), BN, FC (256), BN, FC (2048), BN, Deconv2d (filters 128, kernel_size 5, strides 2), Deconv2d (filters 64, kernel_size 5, strides 2), Deconv2d (filters 3, kernel_size 3, strides 2). ReLU activation.
Arrhythmia	Adam ($1e^{-3}$)	Input	187.
		Encoder	TCN (filters 64, kernel_size 4, stacks 1, dilations $[1, 2, 4, 8, 16, 32]$, padding "causal", use_skip_connections "True", dropout_rate 0.05)
		Decoder	FC (256), FC (384), FC (2048), BN, Deconv1d (filters 64, kernel_size 3, strides 2), Deconv1d (filters 64, kernel_size 3, strides 2), Deconv1d (filters 64, kernel_size 3, strides 2), Deconv1d (filters 32, kernel_size 3, strides 2), Deconv1d (filters 32, kernel_size 3, strides 2), Deconv1d (filters 1, kernel_size 3, strides 2). ReLU activation.

Chapter 5

Conclusion and Future Works

5.1 Conclusion

Many modern data-driven intelligent systems require more accurate anomaly detection techniques. However, considering modern data's (1) unstructured and complex nature, (2) ever-increasing scale, (3) multiple modalities, and (4) processing timeliness and accuracy, detecting anomalies in data flow of intelligent systems is an important but challenging problem. Classical anomaly detection methods such as kernel density estimation and support vector domain description, are incompetent to tackle these problems. Fortunately, embracing the wealth of data, deep learning models rise quickly and have achieved significant successes in various discriminative and generative modellings of modern data. Especially, representation learning mechanisms (such as convolution, word embedding, and recurrence for time-series) have been developed rapidly in recent years.

In this thesis, two deep anomaly detection frameworks: AnoCapsNet and AnoDM are presented based on disentangled representation learning techniques. In AnoCapsNet framework, novel solutions are explored in consideration of CapsNet's distinct characteristics (automatically and dynamically model affine transformations and part-whole relationships using an iterative routing-by-agreement mechanism). Using CapsNet's activation probability and reconstruction error, three normality score functions: prediction-probability-based (PP-based) normality score, reconstruction-error-based (RE-based) normality score, and combined (PP+RE-based) normality score are devised. Experiments on a range of image datasets show that PP-based and RE-based methods have complementary strengths and PP+RE-based method which combines these two methods together outperforms existing advanced solutions in many setups.

In the AnoDM framework, a new methodology is proposed, which successfully integrates t-SNE (low-dimensional manifold learning method) with unsupervised disentangled representation learning (implemented using β-VAE as an example) for anomaly detection via effectively taking the advantages of reconstruction at raw feature space and disentangled latent distribution in t-SNE map. This approach achieves state-of-the-art performances on both image data (MNIST and Fashion-MNIST) and Arrhythmia time-series data. Specifically, best performance is accomplished when $0 < \beta < 1$ for almost all cases involving β-VAE. an anomaly score function is also

defined by successfully combining low-dimensional t-SNE embedding and β-VAE reconstruction. This algorithm demonstrated that t-SNE plays an essential role for measuring abnormality. This research initiates the research on anomaly detection using unsupervised disentangled representation learning and lower-dimensional manifold learning. Besides, the AnoDM framework uses TCN network as encoding architecture for detecting anomalous time-series data and the experimental results convince us that TCN consistently outperforms CNN and LSTM.

5.2 Scopes Limitations

This thesis did not include anomaly detection methods based on two-class classification, as they essentially treat an anomaly detection task as a two-category classification problem by using both normal and abnormal samples in the training process. All out-of-distribution anomaly detection methods as discussed in this thesis only take normal samples for training, because an anomalous data point could unpredictably come from anywhere outside the normal data distribution, which is true in many application domains. It also did not cover deep reinforcement learning methods for anomaly detection as well. But this is a new area worthy of future investigation.

5.2.1 About AnoCapsNet

The performance of CapsNet-based normality score functions depends on CapsNet's learning capacity on certain data. Prediction-probability-based normality score and reconstruction-error-based normality score display complementary strengths, and their combination presents the state-of-art performance on a range of image data sets by making optimum use of their advantages. Even though Sabour et al. [69] mentioned that same as deep generative models, current CapsNets do not perform very well when the backgrounds of images vary too much (such as CIFAR-10) to be modelled, the AnoCapsNet framework using combined (PP+RE) normality score technique still accomplishes quite impressive results.

As AnoCapsNet is based on a supervised model CapsNet on normal samples, same as other supervised learning methods, it also faces some challenges. When identifying anomalous images in pure single class setting, the training of AnoCapsNet is exactly same as two classes (normal and abnormal) classification problem. Therefore, the labeled anomalous data must also be provided. However, in some situations where anomalous samples can not be fully understood, it is not feasible to collect enough anomalous data. Moreover, in many practices, even the domain of normal data can not be thoroughly understood and modeled. Since AnoCapsNet (when used for anomaly detection in multi-class setting) is learnt by feeding labeled normal samples, it is not suitable in these situations where class information is either incomplete or unavailable.

5.2.2 About Beta-VAE

In the experiments of this thesis, despite AnoDM obtained a slightly lower performance than AnoCapNet on some image data, it achieved best performance among several advanced genera-

tive methods such as two types of GAN models: AnoGAN and ADGAN [13]. Since this framework is based on purely generative model (β-VAE), it is applicable even in some applications where data samples are not well defined and modeled. Besides, with flexible architecture, it can also be effectively extended to tackle with anomaly detection on time-series data. Unfortunately, same as other deep generative models, AnoDM faces a challenge on dealing with some complex image data (e.g. Small-Norb and CIFAR-10). Aiming to improve current disentangled representation learning methods (e.g. β-VAE [28, 8] used in the proposed AnoDM), some variants of VAE, such as $\alpha\beta$-VAE [59] mentioned in Section 2.6, FactorVAE [41], β-TCVAE (Total Correlation Variational Autoencoder) [10], SR-VAE (Sequential Residual Variational Autoencoder) [4], and MIM (mutual information machine) [58] are introduced recently. FactorVAE introduced by Kim and Mnih achieves better disentanglement scores than β-VAE on the 2D Shapes and 3D Shapes data sets without degrading reconstruction quality. Kim and Mnih also proposed a simpler disentanglement metric that avoids the failure mode of β-VAE [41]. Chen et al. presented β-TCVAE, a refinement and plug-in replacement of the β-VAE for learning disentangled representations, requiring no additional hyperparameters during training. They further proposed a classifier-free disentanglement metric called mutual information gap (MIG) which can better capture subtle differences in models compared to existing metrics [10]. SR-VAE [4] introduced a "residual learning" mechanism in the training regime to improve trade-off between disentanglement and reconstruction through sequential residual learning. The authors of [4] stated that SR-VAE can achieve state-of-the-art results compared to previous approaches including β-VAE and FactorVAE. Livne et al. [58] introduced MIM approach, which is a new representation learning framework proved to produce higher mutual information and better clustering (and classification) in the latent representation than VAEs, given the same architecture and parametrization. Thus, using the above improved representation learning methods to replace β-VAE architecture in the proposed AnoDM framework may achieve even better results.

5.2.3 About t-SNE

Low-dimensional manifold learning algorithm t-SNE displays significant advantages over PCA for data visualization (as mentioned in Section 2.7.3). Experiments in Section 4.2 also proved that with proper value of perplexity, t-SNE can effectively help AnoDM approach to measure abnormality of unseen data by maintaining some local topological information of latent distributions of input data.

As mentioned in [77], current t-SNE has some weaknesses, for examples, (1) it is a computationally expensive algorithm, (2) it naturally expands dense clusters and contracts sparse ones, evening out cluster sizes, and (3) distances between clusters might not reflect global geometry. McInnes et al. [60] introduced a novel manifold learning technique called Uniform Manifold Approximation and Projection (UMAP) , which was demonstrated being competitive with t-SNE for visualization quality, and arguably preserving more of the global structures with superior run time performance and better scaling. UMAP has a solid theoretical backing and is constructed based on Riemannian geometry and algebraic topology. Since UMAP is a very new technique and the libraries and best practices are not yet firmly established or robust, t-SNE is employed as manifold learning method in current AnoDM framework.

5.3 Future Works

Considering the impressive performance of AnoCapsNet, it might be fruitful to continue deeper research for CapsNet-based anomaly detection problem. As aforementioned above, AnoCapsNet achieved good results on CIFAR-10, even though current CapsNets do not perform very well for classifying this dataset. Thus, as CapsNets acquire the capability of handling complicated datasets, CapsNet-based anomaly detection methods could become more accurate and robust. Since CapsNet is famous with its state-of-the-art performance on some public image datasets, currently there are quite few works for sequence (text) modeling based on CapsNets [81, 42]. This research could be a promising and productive area considering CapsNet is the theoretical improvement of convolution neural network.

Expectation-Maximization routing algorithm in Matrix CapsNet is essentially an inference technique. Therefore, CapsNet is actually not purely discriminative (or supervised) model, having a potential to improve and develop into a fully generative model in the near future. Though CapsNet learns some affine transformations as a supervised method, it is worth exploring unsupervised learning of affine transformations as a future topic.

As a proof of concept, the current AnoDM framework automatically inheres advantages of deep learning to address anomaly detection's issues in representability and scalability as discussed in Chapter 1. The extension of AnoDM framework to multimodal data is straightforward. It is also possible that a neural t-SNE component could be designed and integrated into the learning of β-VAE to achieve real-time efficiency. Other new well-performing manifold learning methods, such as UMAP [60] which is faster and keeps global topologies, could be employed as replacement of t-SNE.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] J. An and S. Cho. Variational autoencoder based anomaly detection using reconstruction probability. Technical report, Data Mining Center, Seoul National University, Seoul, South Korea, 2015.

[3] J. T. A. Andrews, E. J. Morton, and L. D. Griffin. Detecting anomalous data using auto-encoders. *International Journal of Machine Learning and Computing*, 6(1):21–26, 2016.

[4] Anonymous authors. Disentangled representation learning with sequential residual variational autoencoder. 2019.

[5] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *ArXiv*, page arXiv:1803.01271v2, 2018.

[6] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 2:1137–1155, 2003.

[7] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, New York, NY, USA, 1992. ACM.

[8] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in β-VAE. *ArXiv*, page arXiv:1804.03599v1, April 2018.

[9] C. Campbell and Y. Ying. *Learning with Support Vector Machines*. Santa Fe, CA: Morgan and Claypool, 2011.

[10] T. Chen, X. Li, R. B. Grosse, and D. Duvenaud. Isolating sources of disentanglement in variational autoencoders. *ArXiv*, arXiv:1802.04942v5, 2019.

[11] K. Cho, B. V. Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *ArXiv*, page arXiv:1409.1259, 2014.

[12] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. *ArXiv*, page arXiv:1612.08083, 2016.

[13] L. Deecke, R. Vandermeulen, L. Ruff, S. Mandt, and M. Kloft. Image anomaly detection with generative adversarial networks. In *Machine Learning and Knowledge Discovery in Databases*, pages 3–17, Cham, 2019. Springer International Publishing.

[14] A. V. den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu. WaveNet: a generative model for raw audio. In *Speech Synthesis Workshop*, 2016.

[15] L. V. der Maaten and G. E. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579 – 2605, 2008.

[16] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognition*, 58:121–134, 2016.

[17] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin. A convolutional encoder model for neural machine translation. *ArXiv*, page arXiv:1611.02344, 2016.

[18] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1243–1252, 2017.

[19] I. Golan and R. EI-Yaniv. Deep anomaly detection using geometric transformations. *ArXiv*, arXiv:1805.10917, 2018.

[20] I. J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *ArXiv*, page arXiv:1701.00160, 2017.

[21] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[22] Google. Embedding projector. Accessed: 2019-10-8.

[23] N. Gornitz, M. Kloft, K. Rieck, and U. Brefeld. Toward supervised anomaly detection. *Journal of Artificial Intelligence Research*, 43:235–262, 2013.

[24] A. Graves, A. Mohamed, and G. E. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649, 2013.

[25] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. LSTM: A search space odyssey. *ArXiv*, page arXiv:1503.04069, 2015.

[26] R. Hamaguchi, K. Sakurada, and R. Nakamura. Rare event detection using disentangled representation learning. *ArXiv*, page arXiv:1812.01285, 2018.

[27] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *ArXiv*, page arXiv:1512.03385, 2015.

[28] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations*, 2(5):6, April 2017.

[29] G. E. Hinton, A. Krizhevsky, and S. D. Wang. Transforming auto-encoder. In *International Conference on Artificial Neural Networks*, pages 44–51, 2011.

[30] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[31] G. E. Hinton and S. Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems 15*, pages 857–864, Cambridge, MA, USA, 2002. MIT Press.

[32] G. E. Hinton, S. Sabour, and N. Frosst. Matrix capsules with EM routing. In *International Conferences on Learning Representations*, 2018.

[33] G. E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006.

[34] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, December 1997.

[35] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

[36] M. D. Hoffman, C. Riquelme, and M. J. Johnson. The β-VAE's implicit priors. In *Neural Information Processing Systems 2017 Workshop Bayesian Deep Learning*, volume 2, 2017.

[37] A. Javaid, Q. Niyaz, W. Sun, and M. Alam. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pages 21–26. ICST (Institute for Computer Sciences, SocialInformatics and Telecommunications Engineering), 2016.

[38] R. Jozefowicz, W. Zaremba, and I. Sutskever. An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning*, pages 2342–2350, 2015.

[39] M. Kachuee, S. Fazeli, and M. Sarrafzadeh. ECG heartbeat classification: A deep transferable representation. *ArXiv*, page arXiv:1805.00794, 2018.

[40] W. Karush. *Minima of Functions of Several Variables with Inequalities as Side Constraints*. Master's thesis, Univ. of Chicago, Illinois, 1939.

[41] H. Kim and A. Mnih. Disentangling by factorising. *ArXiv*, arXiv:1802.05983v3, 2019.

[42] J. Kim, S. Jang, S. Choi, and E. Park. Text classification using capsules. *ArXiv*, arXiv:1808.03976v2, 2018.

[43] J. Kim and C. Scott. Robust kernel density estimation. *Journal of Machine Learning Research*, 13:3381–3384, May 2008.

[44] D. P Kingma and M. Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

[45] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Department of Computer Science, University of Toronto, Toronto, Canada, 2009.

[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[47] H. W. Kuhn and A. W. Tucke. Nonlinear programming. In *Berkeley Symposium*, pages 481–492, 1951.

[48] Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[49] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, December 1989.

[50] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[51] Y. LeCun, F. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004.

[52] D. Li, D. Chen, J. Goh, and S. Ng. Anomaly detection with generative adversarial networks for multivariate time series. *ArXiv*, page arXiv:1809.04758, 2018.

[53] Y. Li. *Sparse machine learning models in bioinformatics*. PhD thesis, School of Computer Science, University of Windsor, Windsor, Ontario, Canada, 2013.

[54] Y. Li, B. J. Oommen, A. Ngom, and L. Rueda. Pattern classification using a new border identification paradigm: The nearest border technique. *Neurocomptuing*, 157:105–117, 2015.

[55] Y. Li, F. Wu, and A. Ngom. A review on machine learning principles for multi-view biological data integration. *Briefings in Bioinformatics*, 19(2):325–340, 2018.

[56] Y. Li and X. Zhu. Exploring Helmholtz machine and deep belief net in the exponential family perspective. In *International Conference on Machine Learning 2018 Workshop on Theoretical Foundations and Applications of Deep Generative Models*, July 2018.

[57] Y. Li and X. Zhu. Exponential family restricted Boltzmann machines and annealed importance sampling. In *International Joint Conference on Neural Networks (IJCNN)*, pages 39–48, July 2018.

[58] M. Livne, K. Swersky, and D. J. Fleet. Mim: Mutual information machine. *ArXiv*, arXiv:1910.03175v2, 2019.

[59] E. Mathieu, T. Rainforth, N. Siddharth, and Y. W. Teh. Disentangling disentanglement in variational autoencoders. In *ICML*, volume 97, pages 4402–4412, Long Beach, California, USA, Jun 2019. PMLR.

[60] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *ArXiv*, page arXiv:1802.03426, 2018.

[61] G. Melis, C. Dyer, and P. Blunsom. On the state of the art of evaluation in neural language models. *ArXiv*, page arXiv:1707.05589, 2017.

[62] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani. Deep learning for iot big data and streaming analytics: A survey. *ArXiv*, page arXiv:1712.04301, 2017.

[63] D. Park, Y. Hoshi, and C. C. Kemp. A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder. *IEEE Robotics and Automation Letters*, 3(3), 2017.

[64] E. Parzen. On estimation of a probability density function and mode. *Ann. Math. Statist*, 33(3):1065–1076, 1962.

[65] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *Neural Information Processing Systems Autodiff Workshop*, 2017.

[66] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. Nov. 2015.

[67] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages II–1278–II–1286, 2014.

[68] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft. Deep one-class classification. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 4393–4402. PMLR, 2018.

[69] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In *Neural Information Processing Systems*, pages 3856–3866, 2017.

[70] T. Schlegl, P. Seebock, S. M. Waldstein, U. S. Erfurth, and G. Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International Conference on Information Processing in Medical Imaging*, pages 146–157. Springer, 2017.

[71] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comput*, 13(7):1443–1471, Jul 2001.

[72] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT, 2001.

[73] D. M. J. Tax and R. P. W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20:1191–1199, 1999.

[74] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[75] V. N. Vapnik. *Statistical Learning Theory*. Wiley-IEEE Press, New York, 1998.

[76] A. Waibel, T. Hanazawa, G. E. Hinton, K. Shikano, and K. J. Lang. *Readings in Speech Recognition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[77] M. Wattenberg, F. Viégas, and I. Johnson. How to use t-SNE effectively. *Distill*, 2016.

[78] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *ArXiv*, page arXiv:1708.07747, 2017.

[79] X. Xie, C. Wang, S. Chen, G. Shi, and Z. Zhao. Real-time illegal parking detection system based on deep learning. In *Proceedings of the 2017 International Conference on Deep Learning Technologies*, pages 23–27. ACM, 2017.

[80] C. Zhang, J. Butepage, H. Kjellstrom, and S. Mandt. Advances in variational inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):2008 – 2026, 2018.

[81] W. Zhao, J. Ye, M. Yang, Z. Lei, S. Zhang, and Z. Zhao. Investigating capsule networks with dynamic routing for text classification. *ArXiv*, arXiv:1804.00538v4, 2018.