2018

DEEP GENERATIVE MODEL FOR MULTI-CLASS IMBALANCED LEARNING

Yazhou Zhang
University of Rhode Island, yazhou@ele.uri.edu

DEEP GENERATIVE MODEL FOR MULTI-CLASS IMBALANCED LEARNING

BY

YAZHOU ZHANG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

UNIVERSITY OF RHODE ISLAND

2018

MASTER OF SCIENCE THESIS

OF

YAZHOU ZHANG

APPROVED:

Thesis Committee:

Major Professor	Haibo He
	Lutz Hamel
	Yan Sun
	Nasser H. Zawia
	DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2018

ABSTRACT

Learning from imbalanced data has drawn growing attentions nowadays in the machine learning and data mining area. The imbalanced distribution will influence the performance of many machine learning algorithms, especially those need big amount of data. To reduce the influence of skewed data distribution on discriminative models, various synthetic oversampling methods have been proposed to generate extra samples for data balance. However, most of the classic oversampling algorithms, such as Synthetic Minority Over-sampling Technique (SMOTE) or Adaptive Synthetic Sampling Approach (ADASYN), were developed only focusing on balancing the data distribution of low dimensional data in a binary feature space, which limits their application on high dimensional multi-class data.

To deal with the deficiency of current imbalanced learning methods, this thesis proposed a deep generative model based multi-class imbalanced learning algorithm. Both Variational Autoencoder (VAE) and Generative Adversarial Network (GAN) are implemented as data generators for creating high dimensional image data. Besides, we designed an Extended Nearest Neighbor (ENN) based selection process to add the most relevant samples to the original imbalanced database to further improve the classification performance. Based on our experiments on two data sets and comparisons with traditional oversampling algorithms, we demonstrate the effectiveness and robustness of our model.

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor Dr. Haibo He for the useful comments, remarks and engagement through the learning process of this master thesis. Furthermore I would like to thank Zhiqiang Wan who worked with me on this research topics together all these months. Also, I like to thank all the committee members: Dr. Yan Sun, Dr. Lutz Hamel and Defense Chair Dr. Lisa DiPippo who shared their precious time during the reviewing process of my thesis and proposal. I would like to thank my parents, who have supported me throughout entire process, both by keeping me positive and helping me build my confidence. I will be grateful forever for your love.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

Learning from imbalanced data has drawn significant amount of attentions nowadays owing to the pervasive skewed data distribution in numerous data bases. It is a situation where the number of observations belonging to one class is significantly lower than those from the other classes. The skewed distribution leads to poor performance when applying conventional machine learning methods, owing to particularly underrepresented features learned from minority classes.

Furthermore, machine learning algorithms are usually created to improve accuracy by avoiding the erroneous predictions. Therefore, most of these algorithms ignore the data distribution among different classes. When facing imbalanced data set, standard algorithms like K Nearest Neighbors [1] and Decision Tree [2] tend to treat minority samples as noise and hence produce a strong bias towards the majority class.

It is truly worthwhile to explore effective imbalanced learning methods, because imbalanced data is prevalent in many application areas in industry, where anomaly detection is critical like electricity pilferage, fraudulent transactions in banks, identification of rare diseases, etc [3]. In the cyber security area, recognizing patterns in imbalanced data plays a crucial role for data analysis. For instance, to detect cyber attack in a large network, an unusual pattern only takes a relatively small percentage of total data information but plays an crucial role in computer intrusion detection [4]. In the financial engineering area, it is important to detect fraudulent activities, such as credit card, insurance, and insider trading frauds, among a large number of transactions [5].

In order to improve the classification performance of imbalanced data sets, two sets of methodologies were proposed by data mining researchers. Published imbalanced learning solutions can be categorized as algorithm level and data level algorithm [6]. At algorithm level, the classifier itself is modified to bias towards the minority class without changing the original data, such as cost-sensitive learning and recognition-based learning [7]. Cost sensitive learning grants corresponding weights to the cost of different kinds of misclassification. The goal of this type of learning is to minimize the total cost [8]. At data level, oversampling and undersampling methods are applied to create or delete samples to achieve a balanced data distribution.

The classic synthetic oversampling methods achieved the state-of-the-art performance when dealing with imbalanced data. However, those methods are only designed for low dimensional feature space samples in binary classification scenarios and hard to cope with high dimensional data samples, like images, audio signals and time series. For multi-class scenerios, Wang [9] studied the challenges from multi-class imbalanced problems and investigated the generalization abilities of some ensemble solutions, including their recently proposed algorithm AdaBoost.NC. Zong [10] propsed a weighted extreme learning machine method to deal with multiclass imbalanced data which can also be generalized to cost sensitive learning. Li [11] proposed a Boosting weighted Extreme learning machine to solve the weight selection problems of weighted ELM, which also can be used in multiclass imbalanced scenerio. However, all those methods designed for multiclass imbalanced data just focused on balancing data distribution in feature space and got tested on simple data set like UCI [12] and KEEL [13].

The emerging research surge of deep generative models gave us the inspirations for alternative imbalanced learning method to deal with more complicated imbalanced data. Variational Autoencoders (VAE) and Generative Adversarial

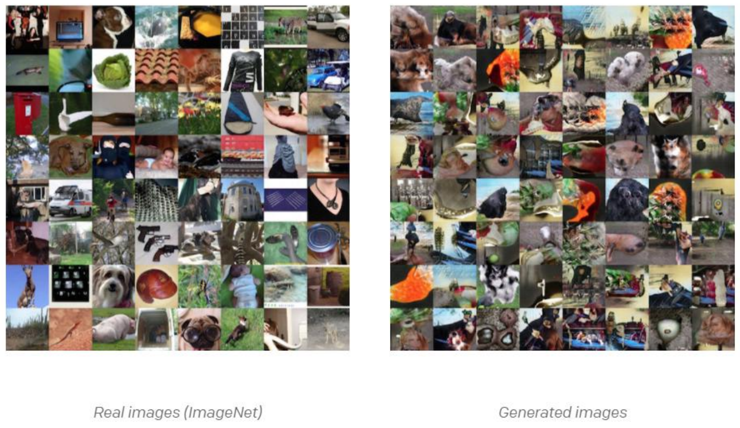


Figure 1: The Image Samples Generated by GAN.

Networks (GAN) as two of the most popular model to learn data distribution in an unsupervised way, have already achieved success in generating a variety of complex data, including handwritten digits, faces, house numbers and CIFAR images [14]. Fig.1 shows the real images and the generated images by GAN .

In this thesis, we explore the possibilities of applying these two generative models in imbalanced learning areas. We choose image data as the input of generative models, and apply Extended Nearest Neighbor (ENN) method to select synthetic candidates for the minority class, and compare the generation results with the traditional synthetic oversampling methods on several different evaluation metrics.

This thesis consists of five chapters, which is arranged as follows: Chapter 1 provides some basic knowledge of traditional imbalanced learning problem and puts forward some potential draw backs of classical approaches. Besides, we give some insights of using deep generative models like GAN and VAE for imbalanced data sets. Chapter 2 offers the details of the background of classical imbalanced learning problems and compares the advantages and disadvantages of traditional oversampling methods. Chapter 3 states the mathmatical foundations of two most famous deep generative models: GAN and VAE, and proposes a novel ENN based selection method to find more suitable samples from the model outputs for imbalanced learning. Chapter 4 provides the performance and analysis of the experiments on two different data sets to prove the effectiveness of our proposed method. The last chapter summarizes the whole thesis and shows the possible future works could be done.

List of References

[1] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE transactions on systems, man, and cybernetics*, no. 4, pp. 580–585, 1985.

[2] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[3] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special issue on learning from imbalanced data sets," *ACM Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 1–6, 2004.

[4] D. J. Marchette, *Computer intrusion detection and network monitoring: a statistical viewpoint.* Springer Science & Business Media, 2001.

[5] M. Ahmed, A. N. Mahmood, and M. R. Islam, "A survey of anomaly detection techniques in financial domain," *Future Generation Computer Systems*, vol. 55, pp. 278–288, 2016.

[6] X.-Y. Liu and Z.-H. Zhou, "The influence of class imbalance on cost-sensitive learning: An empirical study," in *Data Mining, 2006. ICDM'06. Sixth International Conference on.* IEEE, 2006, pp. 970–974.

[7] M. Kukar, I. Kononenko, *et al.*, "Cost-sensitive learning with neural networks." in *ECAI*, 1998, pp. 445–449.

[8] C. Ling and V. Sheng, "Cost-sensitive learning and the class imbalance problem. 2008."

[9] S. Wang and X. Yao, "Multiclass imbalance problems: Analysis and potential solutions," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 4, pp. 1119–1130, 2012.

[10] W. Zong, G.-B. Huang, and Y. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, pp. 229–242, 2013.

[11] K. Li, X. Kong, Z. Lu, L. Wenyin, and J. Yin, "Boosting weighted elm for imbalanced learning," *Neurocomputing*, vol. 128, pp. 15–21, 2014.

[12] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.

[13] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework." *Journal of Multiple-Valued Logic & Soft Computing*, vol. 17, 2011.

[14] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

CHAPTER 2

Background

2.1 Random Sampling Method

Compared with cost sensitive imbalanced learning, the data level sampling methods for imbalanced learning provides straight forward insights to overcome the difficulties of skewed data set. Studies have shown that for several base classifiers, a balanced data set improved overall performance compared with an imbalanced one [1][2][3].

At data level, the goal is to re-balance the data distribution by resampling the database, including oversampling the instances of minority class and under-sampling the majority class.

Among all the resampling method, oversampling methods balance the data set by increasing the number of minority samples, while undersampling methods tries to reduce some majority samples to keep balance [4]. Random oversampling adds the minority samples by randomly replicating existing minority members, which in some degree improve the performance of learning process while it does not provide any additional information to the training set. Besides, random oversampling method may cause overfitting of machine learning models. Compared with random oversampling, random undersampling methods even lose some training information which may have a negative effect on the learning process [5]. Fig.2 and Fig.3 shows the details of these two methods.

2.2 Synthetic Oversampling Methods

In order to provide more information to the training data, synthetic oversampling methods create new samples to balance the data set and achieved better learning performance.

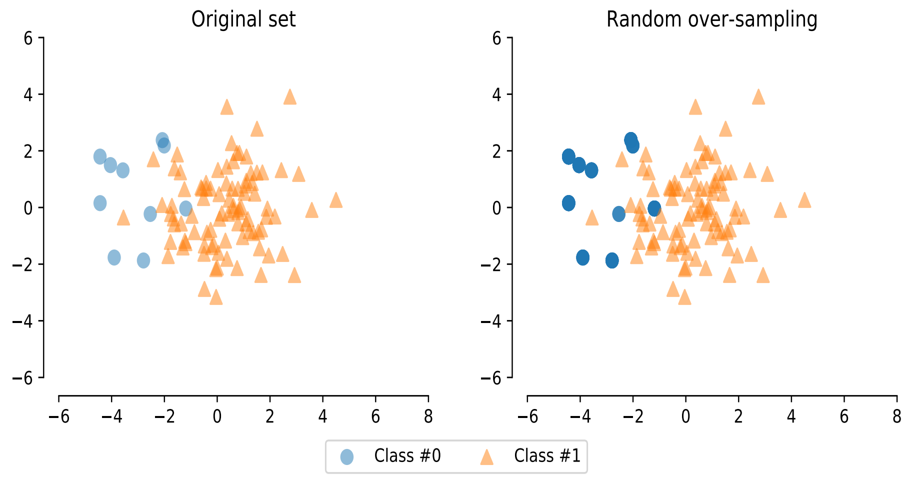


Figure 2: Random Oversampling.

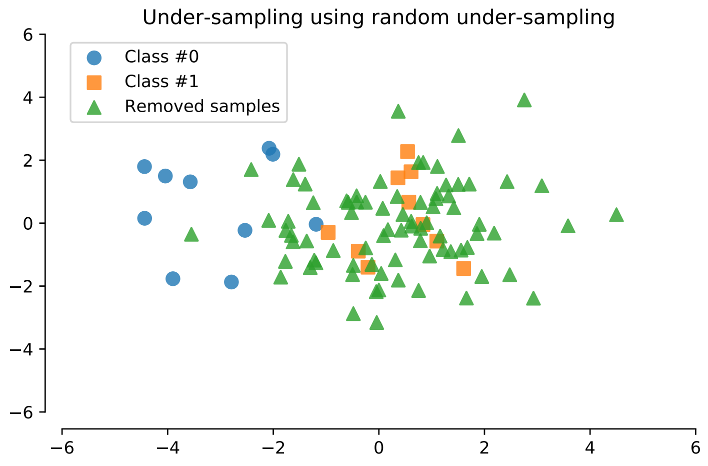


Figure 3: Random Undersampling.

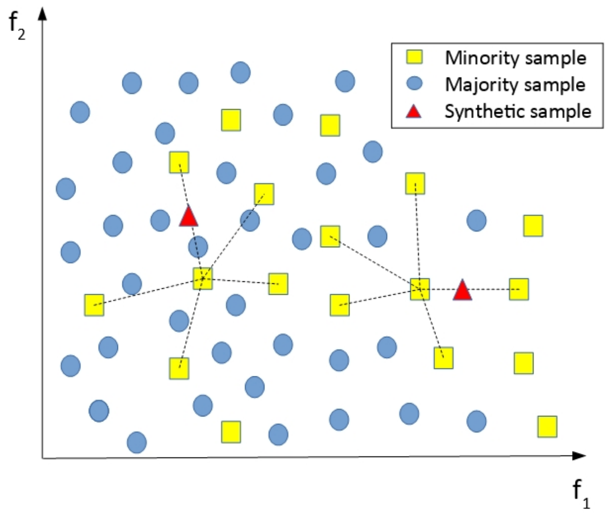


Figure 4: Synthetic Data Generation Based on SMOTE.

As one of the most popular oversampling methods to cope with imbalanced data, the Synthetic Minority Oversampling TEchnique (SMOTE) aims to create ”synthetic” examples based on original samples instead of adding replicated ones to the minority class [6]. SMOTE generates samples according to the similarities among existing minority instance. For specific feature sample x_i in a feature set S, SMOTE find the K-nearest neighbors of x_i in the feature space. To generate a synthetic sample, one of these K-nearest neighbors is randomly selected, then calculate the euclidean distance between these two samples, and at last add the multiplication result of the distance with a random number between $[0, 1]$ to the original feature instance [7]:

$$x_{new} = x_i + (\hat{x_i} - x_i) * \delta \tag{1}$$

Fig.4 shows the process of SMOTE. Compared with random oversampling, SMOTE reduces the overfitting problem to a certain degree and enlarges the minority data in a way that benefits the learning process. However, SMOTE also have its disadvantages like generalization and variance issues [8].

Inspired by the SMOTE algorithm, Han et al. proposed Borderline-SMOTE methods to generate synthetic samples on the borderline between two classes for better classification results [9]. The idea resulted from the fact that the samples close to the borderline are more significant for classification, considering that most machine learning algorithms try to learn the borderline between each class.

The Borderline-SMOTE1 algorithm first finds every example's k nearest neighbors in the entire training set [9]. For every sample p_i, let k' be the number of majority samples in p's k nearest neighbors. If $k' = k$, all the k nearest neighbors are majority samples, which means this sample can be regarded as noise. If $k/2 \leq k' < k$, more majority samples can be found in the k neighbors than the minority ones, so we put p_i in the DANGER set. If $0 \leq k' < k/2$, p_i won't be considered as an endangered sample. So DANGER set can be defined as:

$$DANGER = \{p_1, p_2, ..., p_{dn}\},\ 0 \leq d_n \leq n \tag{2}$$

Where d_n is the number of endangered samples in training set, while n is the size of the training set. At last, we calculate the differences dif_j between p_i and its s nearest neighbors. Then we can generate s new samples by:

$$synthetic_j = p_i + r_j * dif_j,\ j = 1, 2, ..., s \tag{3}$$

Where r_j is a random number between 0 and 1.

Different from Borderline-SMOTE1, Borderline-SMOTE2 not only creates synthetic examples from the nearest minority neighbors of each sample in DANGER set, but also does that from its nearest majority neighbors. Besides, to make the synthetic examples closer to the minority class, a random number between 0 and 0.5 is multiplied by the difference between the endangered sample and its neighbors [9].

In [10], a novel adaptive learning algorithm Adaptive Synthetic Sampling Approach (ADASYN) is proposed to balance the skewed data distribution by generate synthetic samples adaptively. More synthetic examples are created for minority samples which have more majority samples in their K nearest neighbors compared with those have less majority neighbors. Consequently, ADASYN can autonomously shift the decision boundary to those samples that are difficult to learn by the classifier [10].

The procedure of ADASYN is described as below:

Step.1 To begin with, we assume there is an imbalanced data set D_{imb} with m samples x_i, y_i, where $i = 1, ..., m$, x_i and y_i are data samples and labels respectively. Besides, we define the number of majority samples are m_l and the number of minority samples are m_s. Therefore, we got $m_s + m_l = m$ and $m_s < m_l$ [10].

Step.2 Calculate the imbalanced degree of this data set:

$$d = m_s/m_l \tag{4}$$

where $d \in (0, 1]$.

Step.3 Define a threshold for the maximum tolerable imbalance degree d_{th}. If $d < d_{th}$:

(a) Calculate the exact number of samples need to be generated for the minority class:

$$G = (m_l - m_s) * \beta \tag{5}$$

where $\beta \in [0, 1]$ is a number defined to specify the balance degree after sample generation. Usually, β is set to 1 since in most situation, a completely balanced data set is desired.

(b) For each sample x_i in the minority class, find its K nearest neighbors based on the Euclidean distance. Define ratio γ_i as

$$\gamma_i = \Delta_i/K, i = 1, ..., m_s \tag{6}$$

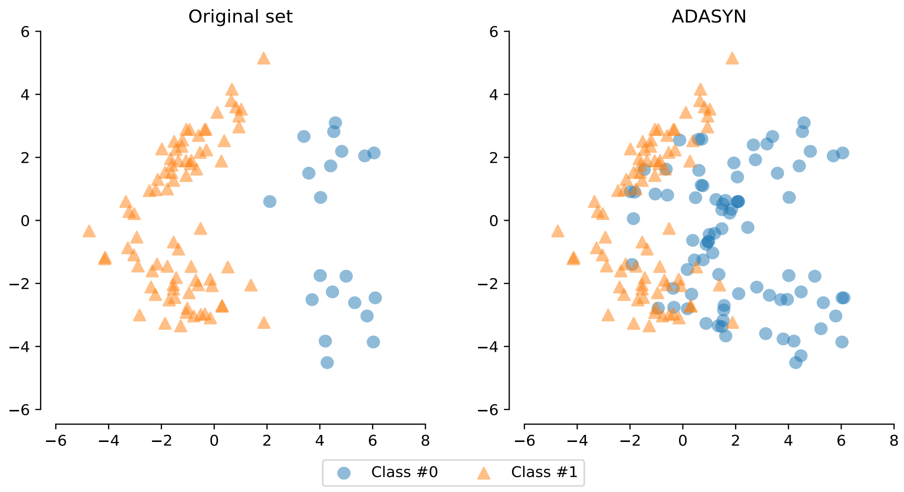


Figure 5: Synthetic Data Generation Based on ADASYN.

where Δ_i is the number of majority samples in the K nearest neighbors of x_i. Therefore $\gamma_i \in [0, 1]$.

(c) Do normalization for γ:

$$\hat{\gamma_i} = \gamma_i / \sum_{i=1}^{m_s} r_i \tag{7}$$

(d) Find the number of synthetic examples which need to be generated for each minority example x_i:

$$g_i = \hat{\gamma_i} * G \tag{8}$$

where G is from Eq.5.

(e) Generate g_i samples for each minority sample x_i iteratively, based on SMOTE algorithm:

$$x_{new} = x_i + (\hat{x_i} - x_i) * \delta \tag{9}$$

where $\delta \in [0, 1]$, is a random number.

2.3 Limitation of Conventional Oversampling Methods

However, the vast majority of existing oversampling methods are proposed to cope with two class scenario in the low dimensional feature space, which limits its application of generating high-dimensional samples in raw data space.

However, all the synthetic sampling methods and techniques mentioned before are designed for imbalanced data set in two-class scenarios. To apply those algorithms in multiclass situations, researchers utilize class decomposition to convert a multiclass problem to a set of binary class subproblems [11]. Given a data set with multi-class N ($N > 2$), a common decomposition plan is to treat one class as minority and to combine all the rest class together as the majority. This approach is also referred as one-against-all (OAA) [12].

Besides the limitation of appliance in multiclass scenarios, introduced synthetic oversampling methods seldom show their superiority when the target sample contains more complex or high-dimensional features, such as image or audio samples. According to the research results from Blagus et al, when dealing with high-dimensional class-imbalanced data, SMOTE does not reduce the classification bias towards majority class for most classifiers, and even less effective than random undersampling [13]. Furthermore, as the number of dimensions grows, the Euclidean distance becomes a meaningless metric to measure the similarity between samples [14]. In some situations, the distance between the target sample and its nearest neighbor might be larger than the distance to its furthest neighbor.

Furthermore, the existing oversampling methods are proposed to deal with the imbalanced learning problems in feature space where feature vectors are presented in a fixed and regularized pattern. However, for the imbalanced data sets which is hard to extract features in an regularized way (e.g. image data), those synthetic oversampling methods fail to generate new samples similar as but different from the original ones.

Therefore, it is necessary for researchers to explore new approaches to generate samples to balance the skewed distribution of high-dimensional data.

List of References

[1] G. M. Weiss and F. Provost, "The effect of class distribution on classifier learning: an empirical study," *Rutgers Univ*, 2001.

[2] J. Laurikkala, "Improving identification of difficult small classes by balancing class distribution," in *Conference on Artificial Intelligence in Medicine in Europe.* Springer, 2001, pp. 63–66.

[3] A. Estabrooks, T. Jo, and N. Japkowicz, "A multiple resampling method for learning from imbalanced data sets," *Computational intelligence*, vol. 20, no. 1, pp. 18–36, 2004.

[4] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent data analysis*, vol. 6, no. 5, pp. 429–449, 2002.

[5] N. V. Chawla, "Data mining for imbalanced datasets: An overview," in *Data mining and knowledge discovery handbook.* Springer, 2009, pp. 875–886.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[7] Z. Wan, Y. Zhang, and H. He, "Variational autoencoder based synthetic data generation for imbalanced learning," in *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on.* IEEE, 2017, pp. 1–7.

[8] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[9] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," *Advances in intelligent computing*, pp. 878–887, 2005.

[10] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on.* IEEE, 2008, pp. 1322–1328.

[11] G. Ou and Y. L. Murphey, "Multi-class pattern classification using neural networks," *Pattern Recognition*, vol. 40, no. 1, pp. 4–18, 2007.

[12] S. Wang and X. Yao, "Multiclass imbalance problems: Analysis and potential solutions," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 4, pp. 1119–1130, 2012.

[13] R. Blagus and L. Lusa, "Evaluation of smote for high-dimensional class-imbalanced microarray data," in *Machine learning and applications (icmla), 2012 11th international conference on*, vol. 2. IEEE, 2012, pp. 89–94.

[14] A. Holzinger, *Machine Learning for Health Informatics: State-of-the-Art and Future Challenges.* Springer, 2016, vol. 9605.

CHAPTER 3

Methodology

3.1 Deep Generative Model

Based on previous statements, it is necessary to explore some new strategies to generate new high-dimensional instances in the raw data space to compensate the skewed distribution. Our research is gong to test the possibility of applying deep generative models, like Variational Autoencoder (VAE) and Generative Adversarial Network (GAN), to generate new samples for imbalanced database.

The generative models recently proposed by deep learning researchers achieved prominent success in image data generation. Variational Autoencoders and Generative Adversarial Networks as two of the most popular models to learn complicated distributions in an unsupervised way, have already show success in generating many kinds of complex data, including handwritten digits, faces, house numbers and CIFAR image [1].

Generative modeling is a wide area of unsupervised learning methods which attempts to learn the underlying data distributions of the original data set [2]. A generative model captures the joint probability of the input data and labels $P(x, y)$ simultaneously, which can be used to generate new data sample similar to existing ones. For example, considering images as input data, each sample (image) has thousands of dimensions (pixels) and the generative model's job is to capture the dependencies between pixels, e.g., that pixels close to each other may be formed into an recognizable object [1]. However, this is not enough for us to generate more samples similar to those already in a database, but not exactly the same, which is the purpose of imbalanced learning. Mathematically, we want to achieve a distribution P which is as close as possible to the original data distribution P_{ori} and where we can get new sample from.

Training this kind of generative models has been a big challenge for several decades, resulting from three serious drawbacks: First, strict assumptions on the original data may be required to achieve reliable results. Second, it is easy to lead to a local optimum if applying sever approximation about the data structure [3]. Third, when applying algorithms like Markov Chain Monte Carlo [4], the training process is very computationally expensive .

3.2 Variational Autoencoder Based Data Generation

More recently, powerful function approximators like neural networks provide more reliable way to training generative model through backpropagation. Variational Autoencoder is one of the most widely implemented deep generative models. Unlike the traditional generative models which either require strong assumptions about the structure of the data or rely on computationally expensive inference procedures, VAE only makes weak assumption on the data, and the training procedure is fast via back propagation [1]. According to the variational auto-encoder literature [1], we consider the following latent variable model for the data X.

VAE tries to learn the joint probability of the input data and labels $P(X, y)$ simultaneously, which can be used to generate new data sample similar to existing ones. In Eq.(10), $P(X \mid z)$ is almost zero for most value of z, therefore, z contributes little to the estimation of $P(X)$. The main goal of the VAE is to sample those values of z which are very likely to generate the data X. Then, $P(X)$ is computed from those z based on Eq.(10).

$$P(X, y) = \int P(X \mid z; \theta, y) P(z) dz. \tag{10}$$

For the purpose of attaining those z values, we need a new function $Q(z \mid X)$ which outputs a distribution over z that are likely to produce data X. The KullbackLeibler (KL) divergence between $Q\left(z \mid X\right)$ and $P\left(z \mid X\right)$ is shown as

$$\begin{aligned}
&D_{KL}\left(Q(z \mid X) \parallel P(z \mid X)\right) \\
=&\mathbb{E}_{z\sim Q}\left[\log Q\left(z \mid X\right) - \log P\left(z \mid X\right)\right] \\
=&\mathbb{E}_{z\sim Q}\left[\log Q\left(z \mid X\right) - \log P\left(X \mid z\right) - \log P\left(z\right)\right] + \log P\left(X\right)
\end{aligned} \tag{11}$$

Then, the marginal likelihood of each sample X can be written as Eq.(12)

$$\begin{aligned}
\log P\left(X\right) = &D_{KL}\left(Q(z \mid X) \parallel P(z \mid X)\right) \\
&+ \mathbb{E}_{z\sim Q}\left[\log P\left(X, z\right) - \log Q\left(z \mid X\right)\right]
\end{aligned} \tag{12}$$

where the first term on the right hand side is the KL divergence between the approximate and the true posterior distribution, and the second term is the variational lower bound on the marginal likelihood of sample X. Eq. (12) can be rewritten as

$$\begin{aligned}
&\log P\left(X\right) - D_{KL}\left(Q(z \mid X) \parallel P(z \mid X)\right) \\
=& - D_{KL}\left[Q\left(z \mid X\right) \parallel P(z)\right] + \mathbb{E}_{z\sim Q}\left[\log P\left(X \mid z\right)\right].
\end{aligned} \tag{13}$$

Eq. (13) is the core of VAE. The left hand side is the one we want to maximize while the right hand side is the one we can optimize via gradient descent. Our goal is to maximize the marginal likelihood $\log P\left(X\right)$ and minimize the KL divergence $D_{KL}\left(Q(z \mid X) \parallel P(z \mid X)\right)$. By minimizing the KL divergence, we are pushing the approximate posterior $Q\left(z \mid X\right)$ to match the true posterior $P\left(z \mid X\right)$. The architecture of VAE is shown in Fig. 6, where P and Q are implemented by neural networks. The architecture of VAE is similar to that of autoencoder, Q encodes data X into latent variable z, and P decodes z and reconstructs X.

During the training process, the right hand side of Eq.(13) is maximized by gradient descent. Its first term is the KL divergence between the approximate posterior and the prior distribution. The approximate posterior is often chosen as

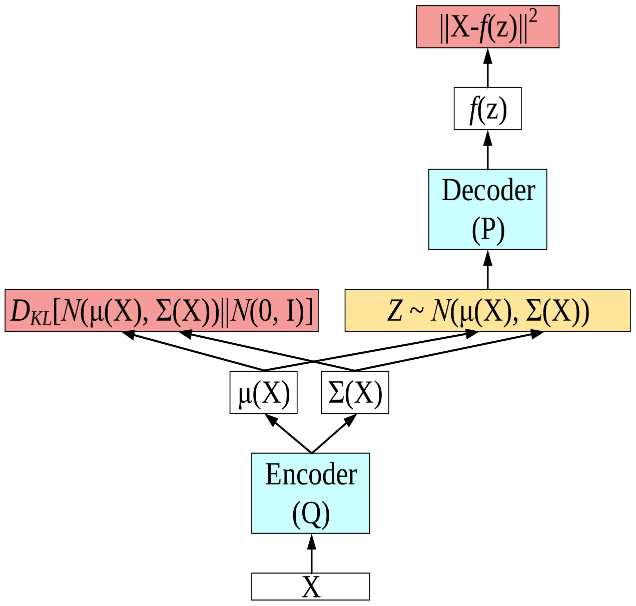


Figure 6: The Architecture of VAE.

a multivariate Gaussian $Q(z \mid X) = \mathcal{N}(z \mid \mu(X;\vartheta), \Sigma(X;\vartheta))$, in which μ and Σ are arbitrary deterministic functions whose distribution parameters ϑ are learned from data [3]. In addition, Σ is constrained to be a diagonal matrix. For the prior distribution, a common used prior over the latent variable is the centered isotropic multivariate Gaussian $P(z) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. By choosing these multivariate Gaussian distributions, it becomes very easy to compute the KL divergence between $Q(z \mid X)$ and $P(z)$ as

$$\begin{aligned} & D_{KL}\left[\mathcal{N}(\mu(X), \Sigma(X)) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I})\right] \\ =& \frac{1}{2}\left[tr(\Sigma(X)) + \mu(X)^T\mu(X) - k - \log\det(\Sigma(X))\right], \end{aligned} \tag{14}$$

Where k is the dimension of the distribution. The second term on the right side of Eq. (13) is an expected negative reconstruction error. Here, we take one sample of z and use $P(X \mid z)$ to approximate $\mathbb{E}_{z\sim Q}[\log P(X \mid z)]$. In general, $P(X \mid z)$

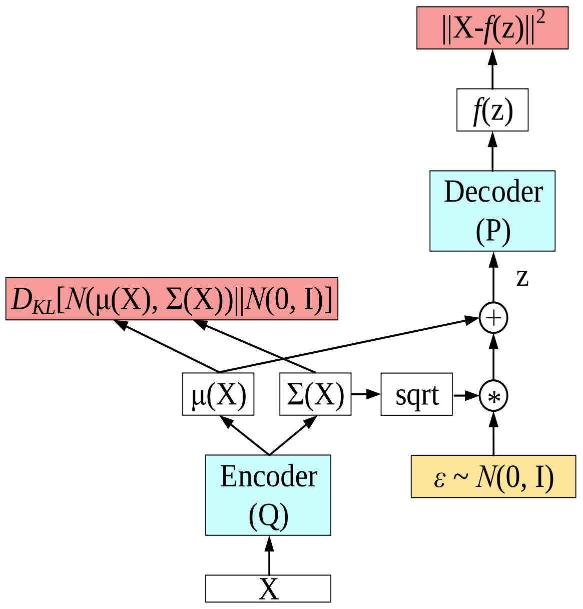


Figure 7: The Architecture of VAE With "Reparameterization trick"

is a multivariate Bernoulli distribution so that

$$\begin{aligned}\log P\left(X \mid z\right) &= \parallel X - f(z) \parallel^2 \\ &= X \log f(z) + (1 - X) \log(1 - f(z)).\end{aligned} \tag{15}$$

The forward pass of Fig.6 works fine, but we can not backpropagate the error through the layer which samples from $Q\left(z \mid X\right)$. This layer has no gradient because the sampling operation is non-continuous. The "reparameterization trick" is proposed to solve this problem by moving the sampling operation to the input layer. The architecture of the modified VAE is shown in Fig. 7. When given the mean and covariance of $Q\left(z \mid X\right)$, i.e., $\mu(X)$ and $\Sigma(X)$, we can first sample ε from $\mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$, then z will equal to $\mu(X) + \Sigma^{1/2}(X) * \varepsilon$. With this modification, we can backpropagate the error from the decoder to encoder [3].

The overall algorithm of VAE based synthetic data generation [3] is shown in Algorithm 1. During the training process, the weights of VAE are updated by minimizing the loss function. After VAE is well trained, we can randomly sample

Algorithm 1 VAE Based Synthetic Data Generation.

```
Require: Initialize weights $\theta$ of the encoder and decoder.

1: for epoch=1:N do
2:     for batch_num=1:M do
3:         Randomly select minibatch of data.
4:         Randomly sample $\varepsilon$ from $\mathcal{N}(\mathbf{0}, \mathbf{I})$.
5:         Feed data and $\varepsilon$ into VAE.
6:         Update weights $\theta$ by minimizing the loss function
           $D_{KL}[\mathcal{N}(\mu(X), \Sigma(X)) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I})] - \parallel X - f(z) \parallel^2$.
7:     end for
8: end for

9: Randomly sample $z$ from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and input these values into the decoder to
   generate new samples.
```

latent variable z from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and apply the decoder to generate new samples.

3.3 Generative Adversarial Network Based Data Generation

Among all the recently proposed generative models, Generative Adversarial Network, or GAN in short, is another outstanding and successful framework used to generate image samples [5]. A GAN consists of two networks: a generative net G that can be trained to acquire the knowledge of data distribution, and a discriminative net D which aims to distinguish the generated samples from real ones [6]. The simple structure of a generative adversarial network is shown in Fig.8. The generator's distribution P_G over data x is modeled as a differentiable function $G(z; \theta_g)$, which can be implemented by a neural network with parameters θ_g and input noise variables z. The discriminator $D(x; \theta_d)$, also implemented by a neural network with parameters θ_g, takes sample x as input and outputs a single scalar representing the probability that x came from the data rather than P_G. The training goal of GAN is to learn a generator distribution $P_G(x)$ that matches the real data distribution $P_{data}(x)$ [5]. To achieve this goal, a minimax gameplay value

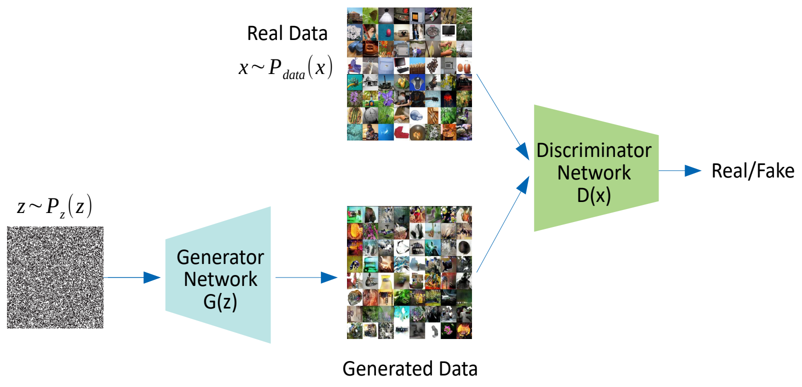


Figure 8: The Architecture of GAN.

function is proposed as follow:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_{data}}[\log D(x)] \\ +\mathbb{E}_{z\sim noise}[\log(1-D(G(z)))] \tag{16}$$

In the training process, the discriminator aims to make $D(G(z))$ approach 0 and $D(x)$ approach 1, while the generator attempts to make $D(G(z))$ approach 1 to maximize the probability of discriminator making a mistake. This network structure corresponds to a minimax two-play game [6].

Concretely, if the batch size of noise variables and training samples is m, we need to update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[logD\left(x^{(i)}\right) + log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right] \tag{17}$$

On the opposite, update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \tag{18}$$

After proper training, the ideal solution to this minimax game is nash equilibrium [7]. The overall algorithm of GAN is shown below:

To adapt to our situation, we can implement a Deep Convolutional Generative Adversarial Network(DCGAN) [8], which applied convolutional layers in the

Algorithm 2 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter.

```
1: for number of training iterations do
2:     for $k$ steps do
3:         Sample minibatch of $m$ noise samples$z^{(1)}, ..., z^{(m)}$ from noise prior $p_g(z)$
4:         Sample minibatch of $m$ samples$x^{(1)}, ..., x^{(m)}$ from noise prior $p_{data}(z)$
5:         Update the discriminator by ascending its stochastic gradient:
               $$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ logD\Big(x^{(i)}\Big) + log\Big(1 - D\Big(G\Big(z^{(i)}\Big)\Big)\Big)\right] \quad (19)$$
6:     end for
7:     Sample minibatch of $m$ noise samples$z^{(1)}, ..., z^{(m)}$ from noise prior $p_g(z)$
8:     Update the generator by descending its stochastic gradient:
               $$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} log\Big(1 - D\Big(G\Big(z^{(i)}\Big)\Big)\Big) \quad (20)$$
9: end for
   The gradient-based updates can use any standard gradient-based learning rule.
```

construction of discriminator and generator. DCGAN proves a powerful image generative model which can be tested for the sample generation for imbalanced image data.

3.4 Extended Nearest Neighbor Based Selection for Borderline Samples

Based on our previous research result [3], which is published in Computational Intelligence (SSCI), 2017 IEEE Symposium Series on IEEE, the outputs of generative models (VAE) can be used together with original data to improve the classification performance. However, not all of the generated instances are of the same significance in classification process.

From the analysis in [9] and [10], most of the existing classifiers attempt to learn the borderline between each class as accurately as possible in the training process, which makes the examples close to the borderline are more likely to be misclassified than ones far from the borderline, thus more significant for classification.

Enlightened by the idea of synthesizing borderline samples, we proposed a novel method to select borderline samples from generative model outputs based on Extended Nearest Neighbor Method (ENN) [11]. Through our selection, only the generated minority samples that close to the boundaries of classes are appended to the original imbalanced data, in order to improve the performance of imbalanced learning.

Compared with further improved versions of K-nearest neighbor(KNN) method [12][13][14][15], the Extended Nearest Neighbor(ENN) predicts input patterns on the basis of the maximum intra-class coherence increment [11]. ENN takes into consideration not only the nearest neighbors of the test sample, but ones who regard the test sample as their nearest neighbors. This two-way communication style provides access to the integral distribution of training data, thus outperforms other related pattern recognition algorithms [11].

The essential definition of ENN is the generalized class-wise statistic T_i. In two-class classification scenario, the generalized class-wise statistic T_i for class i is defined as the following:

$$\begin{aligned} T_i &= \frac{1}{n_i k} \sum_{x \in S_i} \sum_{r=1}^{k} I_r(x, S = S_1 \cup S_2) \\ i &= 1, 2 \end{aligned} \tag{21}$$

where S_1 and S_2 represent samples in class 1 and class 2, respectively, x denotes one single sample in $S = S_1 \cup S_2$, n_i is the number of samples in S_i, and k is the number of nearest neighbors to search in the prediction process. The indicator function $I_r(x, S)$ specifies if both the sample x and its $r-$th nearest neighbor belong to the same class, defined as follows:

$$I_r(x,S)=\begin{cases}1, if\ x\in S_i\ and\ NN_r(x,S)\in S_i\\ 0,\ otherwise\end{cases} \tag{22}$$

where $NN_r(x,S)$ represents the $r-$th nearest neighbor of x in S. This equation indicates that if both x and its $r-$th nearest neighbor in the pool of S belong to the same class, then $I_r(x,S)$ equals 1; otherwise, it equals 0. The large T_i suggests that samples in S_i are much closer together and their nearest neighbors are dominated by the same class samples, whereas a small T_i indicates that samples in S_i have excessive nearest neighbors from other classes [11]. Accordingly, T_i can be used to specify the data distribution across multiple classes. Therefore, the concept of intra-class coherence is defined as follows:

$$\Theta=\sum_{i=1}^{2}T_i \tag{23}$$

To classify an unknown sample Z in multiclass situation, Z is assigned respectively to class 1, class 2,...and class m, to obtain m^2 generalized class-wise statistics T_i^j:

$$T_i^j=\frac{1}{n_i^{'}k}\sum_{x\in S_{i,j}^{'}}\sum_{r=1}^{k}I_r(x,S^{'}=S_1\cup S_2\cup Z)$$
$$i,j=1,2,...,N \tag{24}$$

where $n_i^{'}$ is the size of $S_{i,j}^{'}$ and $S_{i,j}^{'}$ is defined as

$$S_{i,j}^{'}=\begin{cases}S_i\cup Z, & if\ j=i\\ S_i, & if\ j\neq i\end{cases} \tag{25}$$

ENN classifier predicts Z's membership according the following target function:

$$f_{ENN}=\underset{j\in 1,2,...,N}{\operatorname{argmax}}\sum_{i=1}^{n_j}T_i^j=\underset{j\in 1,2,...,N}{\operatorname{argmax}}\Theta^j \tag{26}$$

where

$$\Theta^j = \sum_{i=1}^{N} T_i^j \tag{27}$$

For computational convenience in practical applications, [11] recommended an equivalent target function $f_{ENN.V1}$ to replace f_{ENN}:

$$f_{ENN.V1} = \underset{j \in 1,2,\ldots,N}{\operatorname{argmax}} \left\{ \left(\frac{\Delta n_i^j + k_i - kT_i}{(n_i+1)k} \right)_{i=j} - \sum_{i \neq j}^{N} \frac{\Delta n_i^j}{n_i k} \right\} \tag{28}$$

where k is the defined number of nearest neighbors for prediction, n_i is the number of training samples for class i, k_i is the number of the nearest neighbors of the test sample Z from class i, Δn_i^j represents the change of k nearest neighbors for class i when assigning the test sample Z to class j, and T_i represents the generalized class-wise statistic of original class i without the introduction of Z.

Since $f_{ENN.V1}$ and f_{ENN} are equivalent, we can instead use $f_{ENN.V1}$ to predict the class membership of every generated samples from out model. To select those samples near the boundaries of classes, we proposed a criterion for the selection, the chosen sample must meet the following two requirements:

1) The generated sample must be classified as its original class based on ENN.

2) The generated sample must be neighbored by at least one sample from other class based on the number of nearest neighbor k defined in Eq.24.

Every qualified samples selected from the output of the generative models are added to the original data set until the skewed data distribution is balanced. In case of the situation of not enough samples having neighbors from other classes, we only apply requirement 1) to select samples such that the data set can also be balanced. The complete algorithm structure is shown as Algorithm.3.

Compared with directly using the generated images from our deep generative models, the ENN based selected samples provide more useful information on the boundaries between different categories, therefore leads to better classification performance.

Algorithm 3 Deep Generative Model Based Minority Class Data Generation.

Require: Training ENN with the target imbalanced data set

```
 1: Train VAE/GAN with the samples from all the $N_{minority}$ classes.
 2: Sample a noise vector $z$ as the input of generative network.
 3: for n=1:$N_{minority}$ do
 4:     for $t_{thre}$=1:$N_{thre}$ do
 5:         Generate a class $n$ sample $x_{gen}$ and classify $x_{gen}$ based on ENN.
 6:         Name classification result as $y_{gen}$
 7:         Attain the vector $k_n$
 8:         if $y_{gen}$== n and $k_n$ contains more than 1 none-zero elements :
 9:             Add $x_{gen}$ to the original data set, $N_{gen}++$.
10:         if $N_{gen} + N_{minority} == N_{majority}$:
11:             break
12:     end for
13:     if $N_{gen} + N_{minority} = N_{majority}$ :
14:         continue
15:     else :
16:     repeat:
17:         Step.5 to Step.7.
18:         if $y_{gen}$== n :
19:             Add $x_{gen}$ to the original data set, $N_{gen}++$.
20:     until $N_{gen} + N_{minority} == N_{majority}$
21: end for
```

List of References

[1] C. Doersch, “Tutorial on variational autoencoders,” *arXiv preprint arXiv:1606.05908*, 2016.

[2] E. L. Denton, S. Chintala, R. Fergus, *et al.*, “Deep generative image models using a laplacian pyramid of adversarial networks,” in *Advances in neural information processing systems*, 2015, pp. 1486–1494.

[3] Z. Wan, Y. Zhang, and H. He, “Variational autoencoder based synthetic data generation for imbalanced learning,” in *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on.* IEEE, 2017, pp. 1–7.

[4] W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov chain Monte Carlo in practice.* CRC press, 1995.

[5] I. Goodfellow, “Nips 2016 tutorial: Generative adversarial networks,” *arXiv preprint arXiv:1701.00160*, 2016.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[7] R. Gibbons, *A primer in game theory.* Harvester Wheatsheaf, 1992.

[8] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[9] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," *Advances in intelligent computing*, pp. 878–887, 2005.

[10] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on.* IEEE, 2008, pp. 1322–1328.

[11] B. Tang and H. He, "Enn: Extended nearest neighbor method for pattern recognition [research frontier]," *IEEE Computational intelligence magazine*, vol. 10, no. 3, pp. 52–60, 2015.

[12] H. Zhang, A. C. Berg, M. Maire, and J. Malik, "Svm-knn: Discriminative nearest neighbor classification for visual category recognition," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2. IEEE, 2006, pp. 2126–2136.

[13] M.-L. Zhang and Z.-H. Zhou, "Ml-knn: A lazy learning approach to multi-label learning," *Pattern recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[14] S. A. Dudani, "The distance-weighted k-nearest-neighbor rule," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 4, pp. 325–327, 1976.

[15] T. Seidl and H.-P. Kriegel, "Optimal multi-step k-nearest neighbor search," in *Acm Sigmod Record*, vol. 27, no. 2. Acm, 1998, pp. 154–165.

CHAPTER 4

Simulations and Experiments

To prove the effectiveness of proposed model, we decided to use image data as high-dimensional input of two deep generative models. The parameters of two different models:VAE and GAN, are illustrated in this chapter. Furthermore, we applied ENN based selection to choose most relevant samples for classification. Generated pictures of classical synthetic oversampling methods and our proposed models are also displayed. Finally, we use five different evaluation metrics to compare the classification performance of different approaches.

4.1 Variational Autoencoder based generative model

First, we implement VAE for the image generation. The structure of VAE here is a little different from the vanilla one: we applied convolutional layers [1] here to extract and restore sufficient features of original data samples.

The encoder of VAE consists of six layers: three convolutional layers, one dropout layer, one flatten layer and one dense layer. The kernel size of the convolutional layers is $5 * 5$ with a stride size of $2 * 2$. Both same padding and valid padding are applied here. The encoder structure is shown in Table.1.

The decoder consists of 5 layers: one reshape layer and four deconvolutional layers. The deconvolutional layer, also called transposed convolutional layer, is implemented to map the input vector back to the image pixel space. The network structure of decoder is shown in Table.2.

4.2 Generative Adversarial Nets based generative model

Besides VAE, we applied generative adversarial network as our generative model to produce image samples for minority classes.

Table 1: The Structure of VAE Encoder.

Layer (type)	Padding	Output Shape
Input	-	(28, 28, 1)
Conv1	SAME	(14, 14, 32)
Conv2	SAME	(7, 7, 64)
Conv3	VALID	(2, 2, 128)
Dropout	-	(2, 2, 128)
Flatten	-	(512)
Dense	-	(128)

Table 2: The Structure of VAE Decoder.

Layer (type)	Kernel Size	Stride Size	Padding	Output Shape
Input	-	-	-	(128)
Expand_dims	-	-	-	(1, 1, 128)
De_Conv1	(3, 3)	(1, 1)	VALID	(3, 3, 128)
De_Conv2	(5, 5)	(1, 1)	VALID	(7, 7, 64)
De_Conv3	(5, 5)	(2, 2)	SAME	(14, 14, 32)
De_Conv4	(5, 5)	(2, 2)	SAME	(28, 28, 1)

The generator of our GAN model consists of 5 layers: two dense layers, one dimension-expanding layer and two deconvolutional layers. The first dense layer has the size of 1024, and the second dense layer has the size of $7*7*128$. The first deconvolutional layer has the kernel size of $4*4$ with a stride size of $2*2$, 64 feature maps and Relu is used here as activation function. The second deconvolutional

Table 3: The Structure of GAN Generator.

Layer (type)	Kernel Size	Stride Size	Padding	Output Shape
Input	-	-	-	(128)
Dense	-	-	-	(1024)
Dense	-	-	-	(7*7*128)
Expand_dims	-	-	-	(7, 7, 128)
De_Conv1	(4, 4)	(2, 2)	VALID	(16, 16, 64)
De_Conv2	(4, 4)	(2, 2)	VALID	(28, 28, 1)

layer has the kernel size of $4*4$ with a stride size of $2*2$, 1 feature map and sigmoid is used here as activation function. The generator structure is shown in Table.3.

The discriminator consists of 6 layers: two convolutional layers, one flatten layer and 3 dense layers. The first convolutional layer has the kernel size of $4*4$ with a stride size of $2*2$ and 64 feature maps as output. The second convolutional layer has the kernel size of $4*4$ with a stride size of $2*2$ and 128 feature maps as output. Both these two layers applys leaky Relu as the activation function. The last dense layer has the ouptput size of 1, which represent the possibility of whether generated image sample is real. The discriminator structure is shown in Table 4.

In our implementation, we trained the GAN for 1000 epochs, with a 0.001 learning rate discriminator and a 0.004 learning rate for the generator, in order to keep discriminator in an optimum state and make generator learn the distribution steadily.

Table 4: The Structure of GAN Discriminator.

Layer (type)	Kernal Size	Stride Size	Output Dimension
Conv1	(4,4)	(2,2)	64
Conv2	(4,4)	(2,2)	128
Flatten	-	-	7*7*128
Dense	-	-	1024
Dense	-	-	128
Dense	-	-	1

4.3 Experiment on MNIST Data set

To test the effectiveness of our proposed method, we trained our model on MNIST data set and compared the performance result with other oversampling methods. Since the original MNIST data set is a balanced data set with 6000 samples per class for label 0 to 9, we make some modifications on it to make it unbalanced: we choose digits from 0 to 4 as the minority classes and pick 300 samples from these classes. Then, we pick 3000 samples from label 5 to 9 as the majority classes. We divide this imbalanced data set averagely into three folds to prepare for 3 folds cross validation.

To make this distribution balanced again, we apply VAE to train the minority class samples. New samples will be created by feeding values of $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ into the decoder. To attain suitable samples for the learning process, we apply ENN to select borderline samples. The defined number of nearest neighbors k_{enn} is 10. For each class, 1800 more samples will be generated. To apply traditional sampling methods in this situation, we apply one-verse-all technique, choosing one minority class and combine other classes as one majority class to convert the multi-class problem to a series of two-class subproblems.

For SMOTE, Borderline-SMOTE1, Borderline-SMOTE2, and ADASYN, we set the parameters as below: the number of nearest neighbors used to construct synthetic samples is 5. For Borderline-SMOTE1 and Borderline-SMOTE2, the number of nearest neighbors used to determine whether a minority sample is in danger is 15. For ADASYN, we set the number of nearest neighbors $K = 10$, the balance level coefficient $\beta = 0.1937$ and the threshold for maximum tolerated imbalance ratio $d_{th} = 0.5$.

Fig.10 displays several snapshots of the generated images by SMOTE, ADASYN, and proposed VAE based method. We can observe that in the images generated by SMOTE and ADASYN, numbers are overlapped and somehow blurry. In contrast, VAE-generated numbers are much more clear and sharp. These visual information gives us some insight about the drawback of the classic synthetic sampling methods for high-dimensional data. According to [2], since the Euclidean distance is unsuitable to measure the similarity between high-dimensional samples, the synthetic sampling methods based on Euclidean distance will generate unreasonable images.

To evaluate the performance of our proposed generative structure, we choose convolutional neural networks (CNN)[3] as our basic classifier to do classification.

We applied two convolutional layers in our classifier. The first layer has a kernel size of 3*3, the stride size is 1*1, and 32 feature maps, and the second layer has the same kernel and stride size with Layer 1 with a feature map size of 64. Then we applied 2*2 sized max pooling with a 0.25 dropout rate followed behind. After that, we use a 128-dimension dense layer with 0.5 dropout rate before the final softmax layer. Fig. 9 shows the structure of our classifier. The structure of our CNN remains the same for all other databases used for all the method.

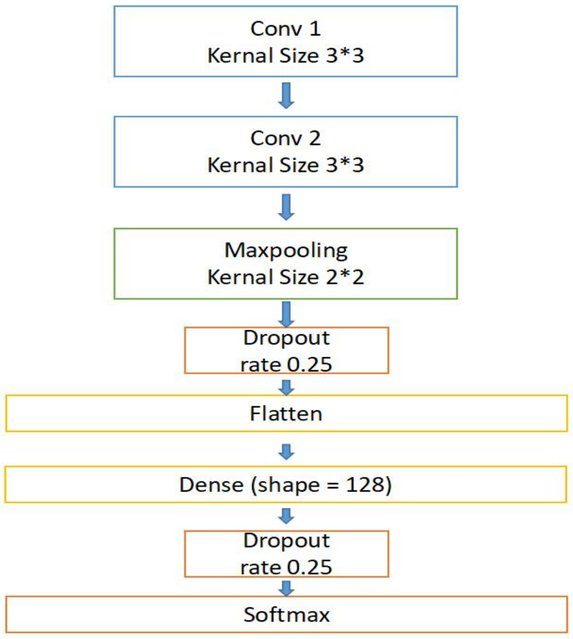
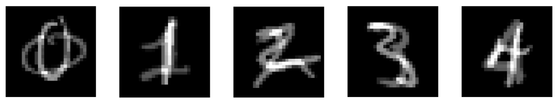
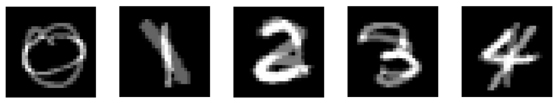


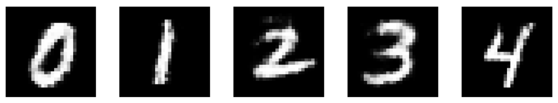
Figure 9: The Architecture of CNN Based Classifier



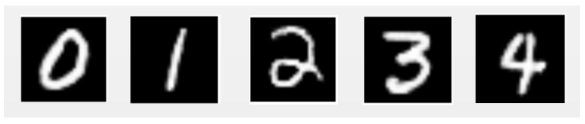
(a) SMOTE



(b) ADASYN



(c) VAE based Method



(d) GAN based Method

Figure 10: Snapshots of The Generated Images on MNIST Dataset

4.4 Evaluation Metrics

We further quantitatively compare the performance of our proposed method with the synthetic sampling methods on the MNIST test set. Considering that accuracy itself may not be sufficient for evaluating the performance of imbalanced learning algorithm [4], we instead apply a set of assessment metrics, such as precision, recall, specificity, F1 score, and G_mean.

1) Precision:

$$Precision = \frac{TP}{TP + FP} \tag{29}$$

where TP represents True Positive and FP represents False Positive.

2) Recall:

$$Recall = \frac{TP}{TP + FN} \tag{30}$$

where FN represents False Negative.

3) Specificity:

$$Specificity = \frac{TN}{TN + FP} \tag{31}$$

where TN represents True Negative.

4) F1 score:

$$F1\ score = \frac{(1 + \beta^2) \cdot Precision \cdot Recall}{\beta^2 \cdot Recall + Precision} \tag{32}$$

where β is a weight coefficient to adjust the significance of recall (usually β =1).

5) G_mean:

$$\begin{aligned} G\_mean &= \sqrt{Recall \cdot Specifity} \\ &= \sqrt{\frac{TP}{TP + FN} \cdot \frac{TN}{TN + FP}} \end{aligned} \tag{33}$$

Since the above metrics are designed for two-class, to apply these metrics in the multi-class scenarios, one-versus-all technique [5] is used to calculate average values of these metrics over all the classes.

Table 5: Evaluation Metrics and Performance Comparison on MNIST.

	Precision	Recall	F1	G_mean	Specificity
SMOTE	0.9356	0.9378	0.9323	0.9580	0.9926
SMOTE(bd1)	0.9357	0.9284	0.9263	0.9627	0.9922
SMOTE(bd2)	0.9325	0.9239	0.9257	0.9612	0.9918
ADASYN	0.9412	0.9349	0.9345	0.9660	0.9929
GAN Based	0.9442	0.9388	0.9386	0.9669	0.9934
VAE Based	**0.9476**	**0.9411**	**0.9420**	**0.9682**	**0.9957**

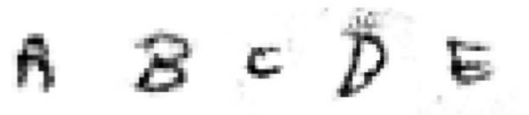
4.5 Performance Comparison

Table. 5 shows the performance of the proposed VAE based method compared to the traditional methods. These results are based on 3 fold cross validation and the high-lighted results are the best in each metrics. It illustrates that the proposed method outperforms the synthetic oversampling methods evaluated by different metrics. These results also show that our proposed method decreases the influence of the skewed data distribution by improving the accuracy for both minority and majority classes.

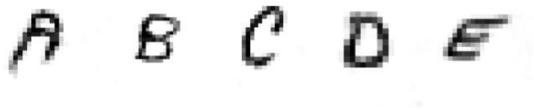
To demonstrate the credibility of our experimental results, we run 20 times of 3 fold cross validation, and conduct the Student's t-test to show significant performance difference exists between the proposed method and ADASYN. Table. 6 shows the p-values of five metrics based on previous 10 run results. Obviously, all these p-values < 0.05, which suggests that the means are significantly different with a 95% confidence.

Table 6: Significance Test for ADASYN and VAE based Method on MNIST.

	Precision	Recall	F1	G_mean	Specificity
P-value($\times 10^{-11}$)	2.77	2.64	4.75	1.67	3.47



(a) VAE based Method



(b) GAN based Method

Figure 11: Snapshots of The Generated Images on NIST19 Dataset

4.6 Experiment on NIST19 Data set

Similarly, we choose other data set NIST19 as the complements of the first experiment. This data set consists of $128 * 128$ gray-scale handwritten English letters, where we reshape the samples into $28 * 28$ and choose A to E as minority classes with 600 randomly chosen samples and F to J as majority classes with 3000 randomly chosen samples. Then the imbalanced data set is divided into three parts of same size, which will be used for cross validation. The parameters of VAE and GAN for this database are the same as the one applied in MNIST but with a 1000 training epochs instead. Besides, we choose the exactly the same classifier with Fig.9.

Fig.11 shows the selected generated images from both models. To attain suitable samples for the learning process, we apply ENN to select borderline samples. The defined number of nearest neighbors k_{enn} is 10. For each class, 1600 more samples will be generated. Again, To apply traditional sampling methods in this

Table 7: Evaluation Metrics and Performance Comparison on NIST19.

	Precision	Recall	F1	G_mean	Specificity
SMOTE	0.9394	0.9295	0.9291	0.9611	0.9922
SMOTE(bd1)	0.9390	0.9296	0.9278	0.9609	0.9919
SMOTE(bd2)	0.9302	0.9234	0.9254	0.9893	0.9917
ADASYN	0.9336	0.9319	0.9316	0.9624	**0.9924**
GAN Based	0.9382	0.9373	0.9370	0.9651	0.9920
VAE Based	**0.9402**	**0.9395**	**0.9388**	**0.9672**	0.9919

Table 8: Significance Test for ADASYN and VAE based Method on NIST19.

	Precision	Recall	F1	G_mean	Specificity
P-value($\times 10^{-11}$)	2.58	4.28	3.74	2.81	3.39

situation, we apply one-verse-all technique, choosing one minority class and combine other classes as one majority class to convert the multi-class problem to a series of two-class subproblems. Table.7 shows the performance over 5 metrics on NIST19 and Table.8 dispaly the significant test results between ADASYN and VAE based method.

To demonstrate the credibility of our experimental results, we run 20 times of 3 fold cross validation, and conduct the Student's t-test to show significant performance difference exists between the proposed VAE based method and ADASYN. Table. 8 shows the p-values of five metrics based on previous 10 results. Obviously, all these p-values < 0.05, which suggests that the means are significantly different with a 95% confidence.

4.7 Performance Analysis

Based on our experiment results, we can see that the generation results of VAE and GAN are different in two aspects:

(1) Compared with GAN, VAE based model tends to produce blurry images, but achieved better performance. (2) Compared with VAE, GAN based model tends to produce clearer and sharper images but it is difficult to train and prone to collapse.

According to [6], VAEs are easier to train and robust to hyperparameter choices and give interpretable latent variables which is learned to map the input to a lower dimensional space. The limitation of VAE is the approximation of posterior $P(X \mid z)$ is usually oversimplified, because we can't parametrize much more complex distribution than normal Gaussian.

On the other side, GAN have the advantages of generating clearer pictures because it applied the adversarial mechanism, which force the generative net to produce something hard for discriminator to distinguish. Hence the lack of adversarial training might be the reason why VAE may generate blurry images.

However, besides the fact that GANs are usually trickier to train compared with VAEs, the training process of GANs may ignore some patterns of original distribution since its ultimate goal is to satisfy the requirements of generator and discriminator.

Fig.12 shows the difference between VAE and GAN in data generation, where the gray lines shows the original data distribution, and colored ones shows the generated data distribution. We can see that from the probability point of view, GAN tends to generate samples follows a specific pattern, while VAE may generate more diverse data samples, some of which may locates out the range of the data distribution.

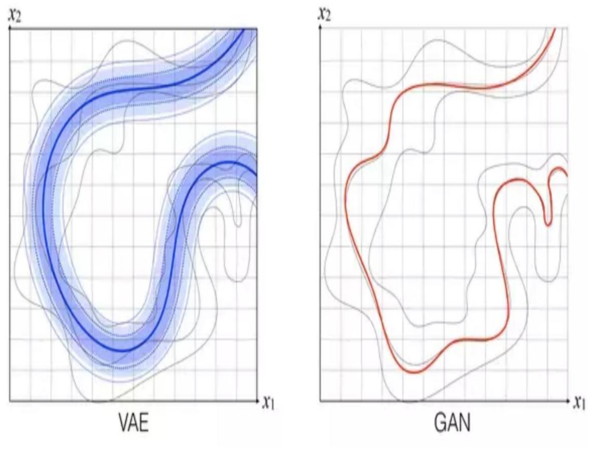


Figure 12: The Data Generation of VAE and GAN

In summary, from perspective of the classification performance, VAE seems to be better than GAN since it can generate samples possessing more diverse features which benefit the training process of classifier. However, from the perspective of generation quality, GAN does better than VAE, since the adversarial process guarantees the output samples looks approximately like the original ones.

List of References

[1] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin, "Variational autoencoder for deep learning of images, labels and captions," in *Advances in neural information processing systems*, 2016, pp. 2352–2360.

[2] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *International conference on database theory.* Springer, 2001, pp. 420–434.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[4] N. Japkowicz, "Assessment metrics for imbalanced learning," *Imbalanced learning: Foundations, algorithms, and applications*, pp. 187–206, 2013.

[5] T. R. Hoens, Q. Qian, N. V. Chawla, and Z.-H. Zhou, "Building decision trees for the multi-class imbalance problem," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Springer, 2012, pp. 122–134.

[6] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville, “Adversarially learned inference,” *arXiv preprint arXiv:1606.00704*, 2016.

CHAPTER 5

Conclusion

Based on our previous analysis and model simulation, we can conclude that deep generative models, like VAE and GAN, could be implemented as image generator to compensate the skewed data distribution, which produce more clear and meaningful samples and performs better compared with traditional feature space oversampling methods. The selection process based on ENN further optimizes the generation process to make it easier to find classification boundaries.

From the generated samples and simulation results, we can see the difference between VAE and GAN : VAE tends to generate a little blurrier imaged compared with the sharp and clear images generated by GAN. However, VAE better captured the distribution of original data base so it tends to lead to better classification results.

Besides, we also test our generative model on more complicated data set like CIFAR 50. However, it is hard to get clear and meaningful generation results, which limit its application on complicated image (like scenery, animals) imbalanced learning. As the best knowledge we have, our proposed model works well on simple structured image data sets, such as MNIST, NIST19 and Fashion MNIST.

Multi-class imbalanced learning on complex data has always been a challenge for data mining and machine learning research. Several published methods of multi-class imbalanced learning just focus on the feature level data, which limits their applications on real world raw data samples. This thesis proposed a deep generative model based imbalanced learning method for image data and provided some insights on how to deal with skewed data distributions by using the state-of-the-art deep learning techniques. However, limited by current generation abilities of GAN and VAE, our contribution is just a small step towards the goal of data space imbalanced learning. We believe that the rising of deep learning, especially the rising of new deep generative models will bring new breakthroughs on this topic.

APPENDIX

Appendix A

A.1 Data sets and Computational Resources

To verify the effectiveness of the potential solutions, we consider to apply three different image data set:

(1) MNIST database

The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST[1]. The digits have been size-normalized and centered in a fixed-size image.

(2) NIST19 database

NIST19 database is a handwritten English letter database. It publishes hand-printed sample forms from 3600 writers, 810,000 character images isolated from their forms, ground truth classifications for those images, reference forms for further data collection, and software utilities for image management and handling.

Since the training and generating process of deep generative models costs lots of computation time. We plan to run the program on the GPU supported computer in our CISA lab.

The implement of our algorithm will be coded in Python using Tensorflow, which is an open source deep learning framework developed by Google Brain[2].

List of References

[1] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

BIBLIOGRAPHY

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

Aggarwal, C. C., Hinneburg, A., and Keim, D. A., "On the surprising behavior of distance metrics in high dimensional space," in *International conference on database theory*. Springer, 2001, pp. 420–434.

Ahmed, M., Mahmood, A. N., and Islam, M. R., "A survey of anomaly detection techniques in financial domain," *Future Generation Computer Systems*, vol. 55, pp. 278–288, 2016.

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., and Herrera, F., "Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework." *Journal of Multiple-Valued Logic & Soft Computing*, vol. 17, 2011.

Asuncion, A. and Newman, D., "Uci machine learning repository," 2007.

Blagus, R. and Lusa, L., "Evaluation of smote for high-dimensional class-imbalanced microarray data," in *Machine learning and applications (icmla), 2012 11th international conference on*, vol. 2. IEEE, 2012, pp. 89–94.

Chawla, N. V., "Data mining for imbalanced datasets: An overview," in *Data mining and knowledge discovery handbook*. Springer, 2009, pp. 875–886.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P., "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

Chawla, N. V., Japkowicz, N., and Kotcz, A., "Special issue on learning from imbalanced data sets," *ACM Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 1–6, 2004.

Denton, E. L., Chintala, S., Fergus, R., *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," in *Advances in neural information processing systems*, 2015, pp. 1486–1494.

Doersch, C., "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

Dudani, S. A., "The distance-weighted k-nearest-neighbor rule," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 4, pp. 325–327, 1976.

Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., and Courville, A., "Adversarially learned inference," *arXiv preprint arXiv:1606.00704*, 2016.

Estabrooks, A., Jo, T., and Japkowicz, N., "A multiple resampling method for learning from imbalanced data sets," *Computational intelligence*, vol. 20, no. 1, pp. 18–36, 2004.

Gibbons, R., *A primer in game theory.* Harvester Wheatsheaf, 1992.

Gilks, W. R., Richardson, S., and Spiegelhalter, D., *Markov chain Monte Carlo in practice.* CRC press, 1995.

Goodfellow, I., "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

Han, H., Wang, W.-Y., and Mao, B.-H., "Borderline-smote: a new over-sampling method in imbalanced data sets learning," *Advances in intelligent computing*, pp. 878–887, 2005.

He, H., Bai, Y., Garcia, E. A., and Li, S., "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on.* IEEE, 2008, pp. 1322–1328.

He, H. and Garcia, E. A., "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

Hoens, T. R., Qian, Q., Chawla, N. V., and Zhou, Z.-H., "Building decision trees for the multi-class imbalance problem," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Springer, 2012, pp. 122–134.

Holzinger, A., *Machine Learning for Health Informatics: State-of-the-Art and Future Challenges.* Springer, 2016, vol. 9605.

Japkowicz, N., "Assessment metrics for imbalanced learning," *Imbalanced learning: Foundations, algorithms, and applications*, pp. 187–206, 2013.

Japkowicz, N. and Stephen, S., "The class imbalance problem: A systematic study," *Intelligent data analysis*, vol. 6, no. 5, pp. 429–449, 2002.

Keller, J. M., Gray, M. R., and Givens, J. A., "A fuzzy k-nearest neighbor algorithm," *IEEE transactions on systems, man, and cybernetics*, no. 4, pp. 580–585, 1985.

Krizhevsky, A., Sutskever, I., and Hinton, G. E., “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

Kukar, M., Kononenko, I., *et al.*, “Cost-sensitive learning with neural networks.” in *ECAI*, 1998, pp. 445–449.

Laurikkala, J., “Improving identification of difficult small classes by balancing class distribution,” in *Conference on Artificial Intelligence in Medicine in Europe.* Springer, 2001, pp. 63–66.

LeCun, Y., “The mnist database of handwritten digits,” *http://yann. lecun. com/exdb/mnist/*, 1998.

Li, K., Kong, X., Lu, Z., Wenyin, L., and Yin, J., “Boosting weighted elm for imbalanced learning,” *Neurocomputing*, vol. 128, pp. 15–21, 2014.

Ling, C. and Sheng, V., “Cost-sensitive learning and the class imbalance problem. 2008.”

Liu, X.-Y. and Zhou, Z.-H., “The influence of class imbalance on cost-sensitive learning: An empirical study,” in *Data Mining, 2006. ICDM’06. Sixth International Conference on.* IEEE, 2006, pp. 970–974.

Marchette, D. J., *Computer intrusion detection and network monitoring: a statistical viewpoint.* Springer Science & Business Media, 2001.

Ou, G. and Murphey, Y. L., “Multi-class pattern classification using neural networks,” *Pattern Recognition*, vol. 40, no. 1, pp. 4–18, 2007.

Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., and Carin, L., “Variational autoencoder for deep learning of images, labels and captions,” in *Advances in neural information processing systems*, 2016, pp. 2352–2360.

Radford, A., Metz, L., and Chintala, S., “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1511.06434*, 2015.

Safavian, S. R. and Landgrebe, D., “A survey of decision tree classifier methodology,” *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

Seidl, T. and Kriegel, H.-P., “Optimal multi-step k-nearest neighbor search,” in *Acm Sigmod Record*, vol. 27, no. 2. Acm, 1998, pp. 154–165.

Tang, B. and He, H., “Enn: Extended nearest neighbor method for pattern recognition [research frontier],” *IEEE Computational intelligence magazine*, vol. 10, no. 3, pp. 52–60, 2015.

Wan, Z., Zhang, Y., and He, H., “Variational autoencoder based synthetic data generation for imbalanced learning,” in *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on.* IEEE, 2017, pp. 1–7.

Wang, S. and Yao, X., “Multiclass imbalance problems: Analysis and potential solutions,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 4, pp. 1119–1130, 2012.

Wang, S. and Yao, X., “Multiclass imbalance problems: Analysis and potential solutions,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 4, pp. 1119–1130, 2012.

Weiss, G. M. and Provost, F., “The effect of class distribution on classifier learning: an empirical study,” *Rutgers Univ*, 2001.

Zhang, H., Berg, A. C., Maire, M., and Malik, J., “Svm-knn: Discriminative nearest neighbor classification for visual category recognition,” in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2. IEEE, 2006, pp. 2126–2136.

Zhang, M.-L. and Zhou, Z.-H., “Ml-knn: A lazy learning approach to multi-label learning,” *Pattern recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

Zong, W., Huang, G.-B., and Chen, Y., “Weighted extreme learning machine for imbalance learning,” *Neurocomputing*, vol. 101, pp. 229–242, 2013.