SetConv: A New Approach for Learning from Imbalanced Data

Yang Gao[1], Yi-Fan Li[1], Yu Lin[1], Charu Aggarwal[2], Latifur Khan[1]
[1]The University of Texas at Dallas, USA
[2]IBM T.J. Watson, USA
{yxg122530,yli,yxl163430,lkhan}@utdallas.edu,charu@us.ibm.com

Abstract

For many real-world classification problems, e.g., sentiment classification, most existing machine learning methods are biased towards the majority class when the Imbalance Ratio (IR) is high. To address this problem, we propose a set convolution (SetConv) operation and an episodic training strategy to extract a single representative for each class, so that classifiers can later be trained on a balanced class distribution. We prove that our proposed algorithm is permutation-invariant despite the order of inputs, and experiments on multiple large-scale benchmark text datasets show the superiority of our proposed framework when compared to other SOTA methods.

1 Introduction

In many real-world NLP applications, the collected data follow a skewed distribution (Deng et al., 2009; Fernández et al., 2013; Yan et al., 2017), i.e., data from a few classes appear much more frequently than those of other classes. For example, tweets related to incidents such as shooting or fire are usually rarer compared to those about sports or entertainments. These data instances often represent objects of interest as their rareness may carry important and useful knowledge (He and Garcia, 2009; Sun et al., 2007; Chen and Shyu, 2011). However, most learning algorithms tend to inefficiently utilize them due to their disadvantage in the population (Krawczyk, 2016). Hence, learning discriminative models with imbalanced class distribution is an important and challenging task to the machine learning community.

Solutions proposed in previous literature can be generally divided into three categories (Krawczyk, 2016): (1) *Data-level methods* that employ under-sampling or over-sampling technique to balance the class distributions (Barua et al., 2014; Smith et al., 2014; Sobhani et al., 2014; Zheng et al., 2015). (2) *Algorithm-level methods* that modify existing learners to alleviate their bias towards the majority classes. The most popular branch is the *cost-sensitive* algorithms, which assign a higher cost on misclassifying the minority class instances. (Díaz-Vico et al., 2018). (3) *Ensemble-based methods* that combine advantages of data-level and algorithm-level methods by merging data-level solutions with classifier ensembles, resulting in robust and efficient learners (Galar et al., 2012; Wang et al., 2015).

Despite the success of these approaches on many applications, some of their drawbacks have been observed. Resampling-based methods need to either remove lots of samples from the majority class or introduce a large amount of synthetic samples to the minority class, which may respectively lose important information or significantly increase the adverse correlation among samples (Wu et al., 2017). It is difficult to set the actual cost value in cost-sensitive approaches and they are often not given by expert before hand (Krawczyk, 2016). Also, how to guarantee and utilize the diversity of classification ensembles is still an open problem in ensemble-based methods (Wu et al., 2017; Huo et al., 2016).

In this paper, we propose a novel *set convolution* (SetConv) operation and a new training strategy named as *episodic training* to assist learning from imbalanced class distributions. The proposed solution naturally addresses the drawbacks of existing methods. Specifically, SetConv explicitly learns the weights of convolution kernels based on the intra-class and inter-class correlations, and uses the learned kernels to extract discriminative features from data of each class. It then compresses these features into a single class representative. These representatives are later applied for classification. Thus, SetConv helps the model to *ignore sample-specific noisy information*, and *focuses on the la-*

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 1284–1294,
November 16–20, 2020.

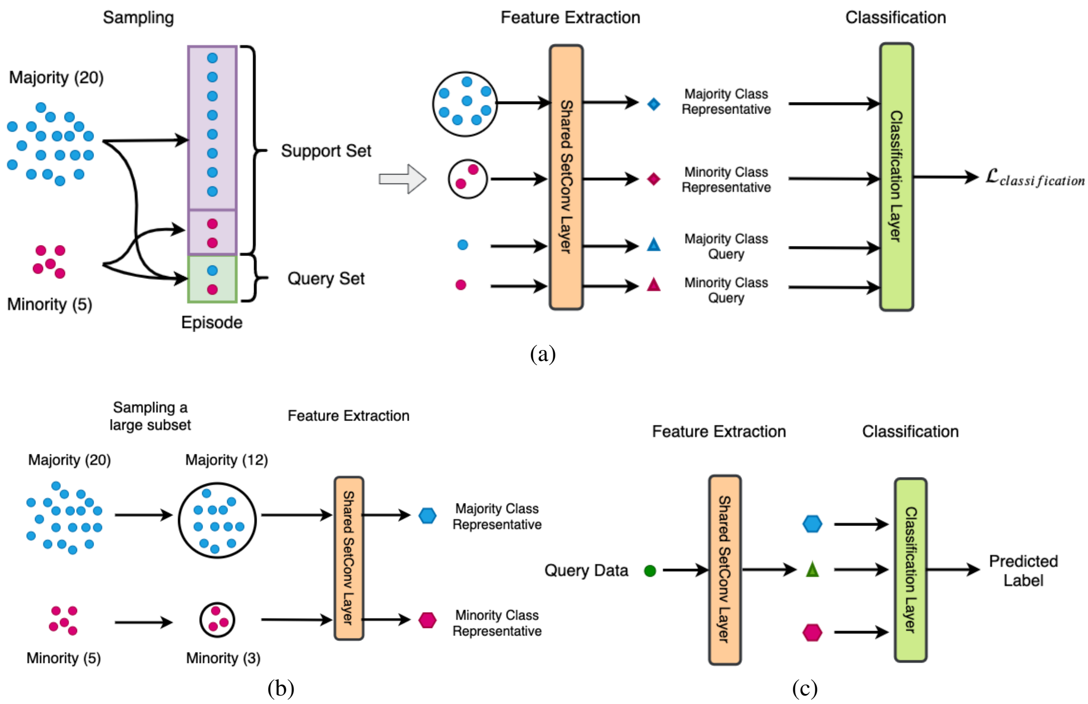

Figure 1: Overview of the proposed approach. (a) The training procedure of SetConv. At each iteration, SetConv is fed with an episode to evaluate the classification loss for model update. Each episode consists of a support set and a query set. The support set is formed by a group of samples where the imbalance ratio is preserved. The query set contains only one sample from each class. (b) The post training step of SetConv, which is performed only once after the main training procedure. In this step, we extract a representative for each class from the training data and will later use them for inference. Here we only perform inference using the trained model and do not update it. (c) The inference procedure of SetConv. Each query data is compared with every class representative to determine its label.

tent concept not only common to different samples of the same class but also discriminative to other classes. On the other hand, in episodic training, we assign equal weights to different classes and do not perform resampling on data. Moreover, at each iteration during training, the model is fed with an episode formed by a set of samples where the class imbalance ratio is preserved. It encourages the model learning to *extract discriminative features even when class distribution is highly unbalanced.*

Building models with SetConv and episodic training has several additional benefits:

(1) *Data-Sensitive Convolution.* By utilizing SetConv, each input sample is associated with a set of weights that are estimated based on its relation to the minority class. This data-sensitive convolution helps the model to customize the feature extraction process for each input sample, which potentially improves the model performance.

(2) *Automatic Class Balancing.* At each iteration, no matter how many data of a class is fed into the model, SetConv always extracts the most discriminative information from them and compress it into a single distributed representation. Thus, the subsequent classifier, which takes these class representatives as input, always perceives a balanced class distribution.

(3) *No dependence on unknown prior knowledge.* The only prior knowledge needed in episodic training is the class imbalance ratio, which can be easily obtained from data in real-world applications.

2 Related Work

2.1 Data-level Methods

The data-level methods modifies the collection of examples by resampling techniques to balance class distributions. Existing data-level methods can be roughly classified into two categories: (1) *Undersampling based methods*: this type of methods balances the distributions between the majority and minority classes by reducing majority-class samples. IHT (Smith et al., 2014) propose to performs undersampling based on instance hardness. On the other hand, EUS (Triguero et al., 2015) introduce evolutionary undersampling methods to deal with large-scale classification problems. (2) *Over-sampling based methods*: these methods balance the class distribution by adding samples to the minority class. SMOTE (Chawla et al., 2002) is the first synthetic minority oversampling technique. MWMOTE (Barua et al., 2014) first identifies the hard-to-learn informative minority class samples and then uses the weighted version of these samples to generate synthetic samples. Recently, based on k-means clustering and SMOTE, KMEANS-SMOTE (Last et al., 2017) is introduced to eliminate inter-class imbalance while at the same time avoiding the generation of noisy samples. However, for highly unbalanced data, resampling methods either discard a large amount of samples from the majority class or introduce many synthetic samples into the minority class. It leads to either the loss of important information (undersampling) or the improper increase of the adverse correlation among samples (oversampling), which will degrade the model performance (Wu et al., 2017).

2.2 Algorithm-level Methods

Algorithm-level methods focus on modifying existing learners to alleviate their bias towards the majority classes. The most popular branch is the cost-sensitive approaches that attempt to assign a higher cost on misclassifying the minority class instances. Cost-sensitive multilayer perceptron (CS-MLP) (Castro and de Pádua Braga, 2013) utilizes a single cost parameter to distinguish the importance of class errors. CLEMS (Huang and Lin, 2017) introduces a cost-sensitive label embedding technique that takes the cost function of interest into account. CS-DMLP (Díaz-Vico et al., 2018) is a deep multi-layer percetron model utilizing cost-sensitive learning to regularize the posterior probability distribution predicted for a given sample. This type of methods normally requires domain knowledge to define the actual cost value, which is often hard in real-world scenarios (Krawczyk, 2016).

2.3 Ensemble-based Methods

Ensemble-based methods usually combine advantages of data-level and algorithm-level methods by merging data-level solutions with classifier ensembles. A typical example is an ensemble model named as WEOB2 (Wang et al., 2015) which utilizes undersampling based online bagging with adaptive weight adjustment to effectively adjust the learning bias from the majority class to the minority class. Unfortunately, how to guarantee and utilize the diversity of classification ensembles is still an open problem in ensemble-based methods (Wu et al., 2017).

3 Model

3.1 Overview

Our goal is to develop a classification model that works well when the class distribution is highly unbalanced. For simplicity, we first consider a binary classification problem and later extend it to the multi-class scenario. As shown in Fig. 1a, our model is composed of a SetConv layer and a classification layer. At each iteration during training, the model is fed with an *episode* sampled from the training data, which is composed of a support set and a query set. The support set preserves the imbalance ratio of training data, and the query set contains only one sample from each class. Once the SetConv layer receives an episode, it extracts features for every sample in the episode and produces a representative for each class in the support set. Then, each sample in the query set is compared with these class representatives in classification layer to determine its label and evaluate the classification loss for model update. We refer this training procedure as *episodic training*.

We choose episodic training due to following reasons: (1) It encourages the SetConv layer learning to extract discriminative features even when the class distribution of the input data is highly unbalanced. (2) Since the episodes are randomly sampled from data with significantly different configuration of support and query sets (i.e., data forming these sets vary from iteration to iteration), it requires the SetConv layer to capture the underlying class concepts that are common among different episodes.

After training, a post training step is performed only once to extract a representative for each class from the training data, which will later be used for inference (Fig. 1b). It is conducted by randomly sampling a large subset of training data (referred as S_{post}) and feeding them to the SetConv layer. *Note that we only perform inference using the trained model and do not update it in this step.* We can conduct this operation because the SetConv layer has learned to capture the class concepts, which are insensitive to the episode configuration during training. We demonstrate it in experiments and the result is shown in Section 4.6.

The inference procedure of the proposed ap-

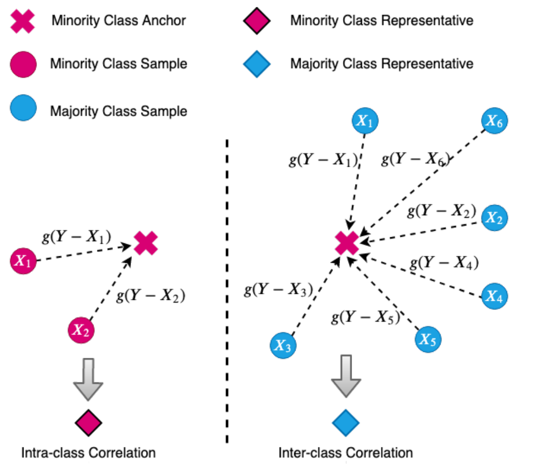

Figure 2: Relations between the input samples and a pre-selected minority class anchor are used by SetConv to estimate both intra-class correlations and inter-class correlations.

proach is straightforward (Fig. 1c). For each query sample, we extract its feature via the SetConv layer and then compare it with those class representatives obtained in post training step. The class that is most similar to the query is assigned as the predicted label.

3.2 SetConv Layer

In many real-world applications, the minority class instances often carry important and useful knowledge that need intensive attention by the machine learning models (He and Garcia, 2009; Sun et al., 2007; Chen and Shyu, 2011).

Based on this prior knowledge, we choose to design the SetConv layer in a way such that the *feature extraction process focuses on the minority class*. We achieve it by estimating the weights of the SetConv layer based on the relation between the input samples and a pre-selected minority class anchor. This anchor can be freely determined by the user. In this paper, we adopt a simple option, i.e., *average-pooling* of the minority class samples. Specifically, for each input variable, we compute its mean value across all the minority class samples in the training data. It is executable because the minority-class samples are usually limited in real-world applications[1]. As shown in Figure 2, this weight estimation method assists the SetConv layer in capturing not only the intra-class correlation of the minority class, but also the inter-class correlation between the majority and minority classes.

[1]Otherwise, we may sample a subset from the minority class samples to compute the anchor.

Suppose $\mathcal{E}_t = \{\mathcal{S}_t, \mathcal{Q}_t\}$ is the episode sent to the SetConv layer at iteration t, where $\mathcal{S}_t = \big(X_{maj} \in \mathcal{R}^{N_1\times d}, X_{min} \in \mathcal{R}^{N_2\times d}\big)$ is the support set and $\mathcal{Q}_t = \big(q_{maj} \in \mathcal{R}^{1\times d}, q_{min} \in \mathcal{R}^{1\times d}\big)$ is the query set. In general, X_{maj}, X_{min}, q_{maj} and q_{min} can be considered as a sample set of size N_1, N_2, 1 and 1 respectively. For simplicity, we abstract this sample set into $X \in \mathcal{R}^{N\times d}, N \in \{N_1, N_2, 1\}$.

Remind that the standard discrete convolution is:

$$h[n] = (f \star g)[n] = \sum_{m=-M}^{m=M} f[m]g[n-m] \quad (1)$$

Here, f and g denote the feature map and kernel weights respectively.

Similarly, in our case, we define the set convolution (SetConv) operation as:

$$\begin{aligned} h[Y] &= \frac{1}{N}\sum_{i=1}^{N} X_i \cdot g(Y - X_i) \\ &= \frac{1}{N}\Big(X \circ g(Y - X)\Big) \end{aligned} \quad (2)$$

where $Y \in \mathcal{R}^{1\times d}$, $g(Y-X) \in \mathcal{R}^{N\times d\times d_o}$ and $h[Y] \in \mathcal{R}^{1\times d_o}$ denote the minority class anchor, kernel weights and the output embedding respectively. Here, $\circ$ is the tensor dot product operator, i.e., for every $i \in \{1, 2, \dots, d_o\}$, we compute the dot product of X and $g(Y-X)[:, :, i]$.

Unfortunately, directly learning $g(Y-X)$ is memory intensive and computationally expensive, especially for large-scale high-dimensional data. To overcome this issue, we introduce an efficient method to approximate these kernel weights. Instead of taking X as a set of d-dimensional samples, we stack these samples and consider it as a giant dummy sample $X' = Concat(X) \in \mathcal{R}^{1\times Nd}$. Then, Eq. 2 is rewritten as

$$h[Y] = \frac{1}{N}\Big(X' \cdot g'(Y - X)\Big) \quad (3)$$

where $g'(Y-X) \in \mathcal{R}^{Nd\times d_o}$ is the transformed kernel weights. To efficiently compute $g'(Y-X)$, we propose to approximate it as the *Khatri-Rao* product[2] (Rabanser et al., 2017) of two individual components, i.e.,

$$\begin{aligned} g'(Y-X) &= g_1(Y-X) \circledast g_2(W) \\ &= \text{MLP}(Y-X;\theta) \circledast \text{SoftMax}(W, 0) \end{aligned} \quad (4)$$

[2]https://en.wikipedia.org/wiki/Kronecker_product

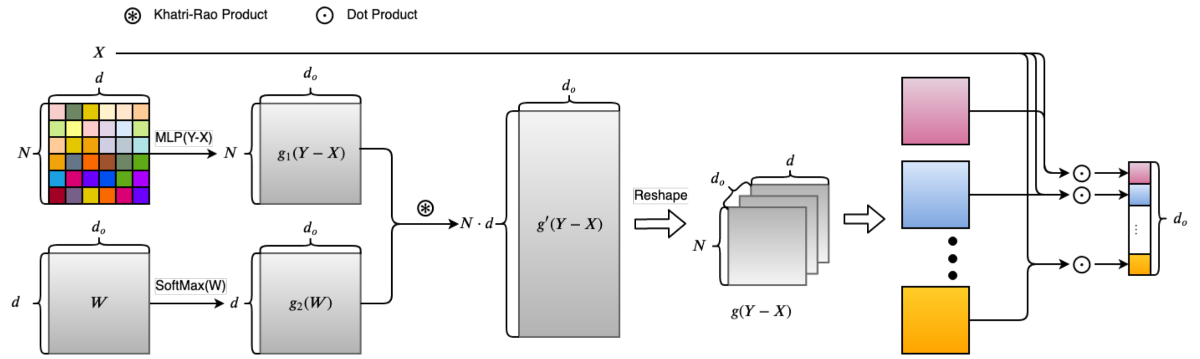

Figure 3: The computation graph of the SetConv layer. Here Y is a minority class anchor. $W \in \mathcal{R}^{d \times d_o}$ is a weight matrix to learn that records the correlation between the input and output variables. Specifically, the i_{th} column of $g_2(W)$ gives the weight distribution over input features for the i_{th} output feature. It is indeed a feature-level attention matrix. In addition, we estimate another data-sensitive weight matrix $g_1(Y - X)$ from the input data. The final convolution weight tensor is simply the Khatri-Rao product of $g_1(Y - X)$ and $g_2(W)$.

where $W \in \mathcal{R}^{d \times d_o}$ is a weight matrix that represents the correlation between input and output variables. $g_2(W)$ takes softmax over the first dimension of W, and is indeed a *feature-level attention* matrix. The i_{th} column of $g_2(W)$ provides the weight distribution over input features for the i_{th} output feature. On the other hand, $g_1(Y - X)$ is a *data-sensitive* weight matrix estimated from input data via a MLP by considering their relation to the minority class anchor. Similar to data-level attention, $g_1(Y - X)$ helps the model customize the feature extraction process for input samples, which potentially improves the model performance. Figure 3 shows the detailed computation graph of the SetConv layer.

Discussion: An important property of the SetConv layer is ***permutation-invariant***, i.e., it is insensitive to the order of input samples. As long as the input samples are same, no matter in which order they are sent to the model, the SetConv layer always produces the same feature representation. Mathematically, let π denote an arbitrary permutation matrix, we have $SetConv(\pi X) = SetConv(X)$. The detailed proof of this property is provided in the supplementary material.

3.3 Classification

Suppose the feature representation obtained from the SetConv layer for X_{maj}, X_{min}, q_{maj} and q_{min} in the episode are denoted by v^s_{maj}, v^s_{min}, v^q_{maj} and v^q_{min} respectively. The probability of predicting v^q_{maj} or v^q_{min} as the majority class is given by

$$P(c = 0|x) = \frac{\exp(x \odot v^s_{maj})}{\exp(x \odot v^s_{maj}) + \exp(x \odot v^s_{min})} \tag{5}$$

where $\odot$ represents the dot product operation and $x \in \{v^q_{maj}, v^q_{min}\}$.

Similarly, the probability of predicting v^q_{maj} or v^q_{min} as the minority class is

$$P(c = 1|x) = \frac{\exp(x \odot v^s_{min})}{\exp(x \odot v^s_{maj}) + \exp(x \odot v^s_{min})} \tag{6}$$

where $x \in \{v^q_{maj}, v^q_{min}\}$.

We adopt the well-known cross-entropy loss for error estimation and use the Adam optimizer to update model.

3.4 Extension to Multi-Class Scenario

Extending SetConv for multi-class imbalance learning is straightforward. We translate the multi-class classification problem into multiple binary classification problems, i.e., we create a one-vs-all classifier for each of the N classes. Specifically, for a class c, we treat those instances with label $y = c$ as positive and those with $y \neq c$ as negative. The anchor is hence computed based on the smaller one of the positive and negative classes. The prediction probability $P(y = c|x)$ for a given instance x is computed in a similar way as Eq. 5,

$$P(y = c|x) = \frac{\exp(x \odot v^s_{y=c})}{\exp(x \odot v^s_{y \neq c}) + \exp(x \odot v^s_{y=c})} \tag{7}$$

Therefore, the predicted label of the instance x is $\text{argmax}_c P(y = c|x)$.

4 Experiment

We evaluate SetConv on two typical tasks, including *incident detection on social media* and *sentiment classification*.

Table 1: Class distribution in the IRT dataset.

	Two Classes		Four Classes			
	Yes	No	Crash	Fire	Shooting	No
Boston (USA)	604	2216	347	188	28	2257
Sydney (AUS)	852	1991	587	189	39	2208
Brisbane (AUS)	689	1898	497	164	12	1915
Chicago (USA)	214	1270	129	81	4	1270
Dublin (IRE)	199	2616	131	33	21	2630
London (UK)	552	2444	283	95	29	2475
Memphis (USA)	361	721	23	30	27	721
NYC (USA)	413	1446	129	239	45	1446
SF (USA)	304	1176	161	82	61	1176
Seattle (USA)	800	1404	204	153	139	390

Table 2: Class distribution in Amazon Review and SemiEval Datasets.

Dataset	Negative	Positive	IR
Amazon-Books	72039	7389	9.75
Amazon-Electronics	13560	1908	7.11
Amazon-Movies	12896	2066	6.24
SemiEval	39123	7273	5.38

4.1 Benchmark Dataset

4.1.1 Incident Detection on Social Media

We conduct experiments on a real-world benchmark *Incident-Related Tweet*[3] (Schulz et al., 2017) (**IRT**) dataset. It contains $22,170$ tweets collected from 10 cities, and allows us to evaluate our approach against geographical variations. The IRT dataset supports two different problem settings: binary classification and multi-class classification. In binary classification, each tweet is either "incident-related" or "not incident-related". In multi-class classification, each tweet belongs to one of the four categories including "crash", "fire", "shooting" and a neutral class "not incident related". The details of this dataset are shown in Table 1.

4.1.2 Sentiment Classification

We conduct experiments on two large-scale benchmark datasets, including *Amazon Review*[4] (He and McAuley, 2016) and *SemiEval*[5] (Rosenthal et al., 2017), which have been widely used for sentiment classification. Similar to MSDA (Li et al., 2019) and SCL-MI (Blitzer et al., 2007), we treat the amazon reviews with rating > 3 as positive examples, those with rating < 3 as negative examples, and discard the rest because their polarities are ambiguous. In addition, due to the tremendous size of Amazon Review dataset, we choose its 3 largest categories, i.e., "Books", "Electronics", and "Movies and TV", and uniformly sample from these categories to form a subset that contains $109,858$ reviews. This subset is sufficiently large to evaluate the effectiveness of our method. More importantly, the imbalance ratio of each category in this subset is exactly same as that in the original dataset. Details of Amazon Review and SemiEval datasets are listed in Table 2.

[3]http://www.doc.gold.ac.uk/%7Ecguck001/IncidentTweets/

[4]http://jmcauley.ucsd.edu/data/amazon/

[5]http://alt.qcri.org/semeval2017/task4/index.php?id=data-and-tools

4.2 Baseline

We compare our algorithm with several state-of-the-art imbalance learning methods.

- **IHT** (Smith et al., 2014) (*under-sampling*) is a model that performs undersampling based on instance hardness.
- **WEOB2** (Wang et al., 2015) (*ensemble*) is an undersampling based ensemble model that effectively adjusts the learning bias from the majority class to the minority class via adaptive weight adjustment. It only supports binary classification.
- **KMeans-SMOTE** (Last et al., 2017) (*over-sampling*) is an oversampling technique that avoids the generation of noisy samples and effectively overcomes the imbalance between classes.
- **IML** (Wang et al., 2018) (*metric learning*) is a method that utilizes metric learning to explore the correlations among imbalance data and constructs an effective data space for classification.
- **CS-DMLP** (Díaz-Vico et al., 2018) (*cost-sensitive*) is a deep MLP model that utilizes cost-sensitive learning to regularize the posterior probability distribution predicted for a given sample.

4.3 Evaluation Metric

We use the Specificity (*Spec*), Sensitivity (*Sens*), F_1-measure (F_1), Geometric-Mean (*G-Mean*), and the Area Under the receiver operating characteristic Curve (*AUC*) to evaluate the model performance, since they are widely used in previous imbalance learning research (Wang et al., 2018; Díaz-Vico et al., 2018; Last et al., 2017). The confusion matrix for multi-class classification is shown in Table 3. In the multi-class scenario, we report the model performance for each of the minority classes because: (1) the minority classes are usually more important than the majority class in most imbalance learning

Table 3: Confusion matrix for multi-class classification problem, where c denotes the class to evaluate.

	Predict Label $= c$	Predict Label $\neq c$
True Label $= c$	True Positive (TP)	False Negative (FN)
True Label $\neq c$	False Positive (FP)	True Negative (TN)

problems (He and Garcia, 2009; Chen and Shyu, 2011). (2) simply averaging model performance on different classes may cover model defects, especially when the class distribution is unbalanced.

(1) Class-specific performance measure:

- $Spec = \frac{TN}{TN+FP}$. Spec measures the model's capability to avoid false positive and finds all negative samples.
- $Sens = \frac{TP}{TP+FN}$. Sens measures the model's capability to avoid false negative and finds all positive samples.

(2) Overall performance measure:

- $F_1 = 2 \cdot \frac{\textit{precision} \cdot \textit{recall}}{\textit{precision} + \textit{recall}}$ is the harmonic mean of *precision* $= \frac{TP}{TP+FP}$ and *recall* $= \frac{TP}{TP+FN}$.
- *G-Mean* $= \sqrt{Spec \cdot Sens}$. G-Mean receives a higher value only when both Spec and Sens stay at a higher level. Thus, *G-Mean* can be considered as a trade-off between *Spec* and *Sens*.
- *AUC* computes the area under the ROC curve. It measures the model's capability to distinguish positive and negative classes.

In general, the model that gives higher values on these metrics is the one with better performance.

4.4 Experiment Setup

For all text datasets, we first pre-process each data instance via a pretrained Bert[6] (Devlin et al., 2019) model to produce a 1024-dimension feature vector, which is utilized for subsequent experiments. Note that this step does not lead to any ground-truth information leakage, because Bert is trained on Wikipedia corpus in an unsupervised manner.

Specifically, we choose the *BERT-Large, Cased (Whole Word Masking)*[7] model provided by Google Research team, and take the final hidden state of the special classification token [CLS] as the embedding for any input text sequence. This process is described in Figure 4.

[6]https://github.com/huggingface/transformers

[7]https://github.com/google-research/bert

```
'''Please install transformers via pip install transformers'''
from transformers import BertModel, BertTokenizer
import torch

# create model
tokenizer = BertTokenizer.from_pretrained('bert-large-cased')
model = BertModel.from_pretrained('bert-large-cased')

input = 'Here is some text to encode'
tokens = tokenizer.encode(input, add_special_tokens=True, max_length=512)
input_ids = torch.tensor([tokens])
with torch.no_grad():
    last_hidden_state = model(input_ids)[0]
# we take the final hidden state of the special classification token [CLS]
# as the embedding for input sentences
embedding = last_hidden_state[:, 0, :]
```

Figure 4: Implementation code used to extract sentence embedding via Bert.

After data pre-processing, we uniformly shuffle each dataset, and then divide it into development and test sets with the split ratio of 7:3. Thus, the class distribution in both development and test sets is same as that in the original dataset. To avoid any influence of random division, we repeat the experiments 10 times and report the average classification results.

We implement our algorithm using Python 3.7.3 and PyTorch 1.2.0 library. All baseline methods are based on code released by corresponding authors. Hyper-parameters of these baselines were set based on values reported by the authors and fine-tuned via 10-fold cross-validation on the development set. In our approach, we set the output dimension of the SetConv layer $d_o = 128$, the size of support set $||S_{support}|| = N_1 + N_2 = 64$, the size of post-training subset $||S_{post}|| = 1000$, learning rate $r = 0.01$, $\beta_1 = 0.9$ (Adam), and $\beta_2 = 0.999$ (Adam). The input dimension d of the SetConv layer is set to be the same as the dimension of input data for each dataset. The sensitivity analysis of $||S_{post}||$ is shown in Section 4.6.

4.5 Result

4.5.1 Binary Classification

The binary classification performance of competing methods for incident detection and sentiment classification tasks are shown in Figure 5 and Figure 7 respectively. The results demonstrate that the proposed algorithm *significantly outperforms* the competing methods in most cases and achieves the *best* classification performance. Moreover, as shown in Figure 6, in contrast to baselines that are biased towards either the majority or minority class, the *high values of specificity and sensitivity* indicate that our algorithm performs almost *equally well* on

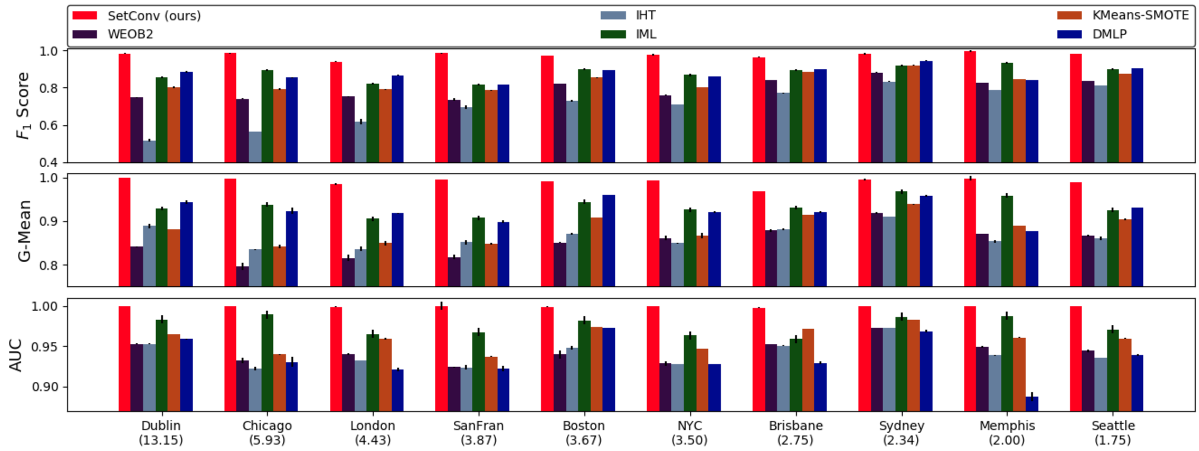

Figure 5: Binary classification (incident detection) performance of competing methods on the IRT dataset. The value in the bracket indicates the imbalance ratio (IR).

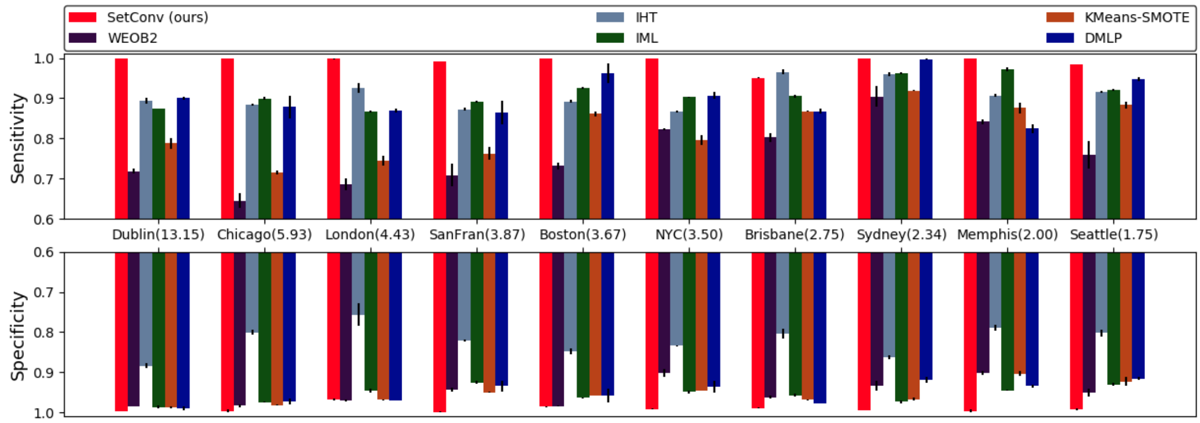

Figure 6: The performance diagnosis of competing methods for binary classification. The value in the bracket indicates the imbalance ratio (IR). In contrast to baselines that are biased towards either the majority or minority class, SetConv performs almost equally well on both classes.

both classes. That is, it not only makes few false positive predictions, but also produces few false negative predictions. It is also observed that our method is insensitive to geographical variations.

Our approach performs better because (1) compared to resampling based approaches, e.g., IHT and WEOB2, it makes full utilization of data via episodic training and set convolution operation, which avoids removing lot of samples from the majority class and losing important information. (2) compared to IML, SetConv enhances the feature extraction process by learning to extract discriminative features from a set of samples and compressing it into a single representation. It helps model to ignore sample-specific noisy information and focuses only on the latent concept common to different samples. (3) Compared to cost-sensitive approaches, e.g., CS-DMLP, episodic training assigns equal weights to both the majority and minority classes and eliminates the overhead of finding suitable cost values for different datasets. The model is forced to address class imbalance by learning to extract discriminative features during training.

4.5.2 Multi-Class Classification

To verify the effectiveness of the proposed algorithm in the multi-class classification scenario, we first compare it with competing methods on the IRT dataset (incident detection) and report their performance on the three minority classes, i.e., “Fire”, “Shooting”, and “Crash”. Due to space limitation, we only show the results of New York City (NYC) in Table 4, although similar results have been observed for other cities. We observe that our approach significantly outperforms baseline methods by providing much higher F_1, G-Mean and AUC metrics. Moreover, in contrast to baseline methods, it performs almost equally well on all the three

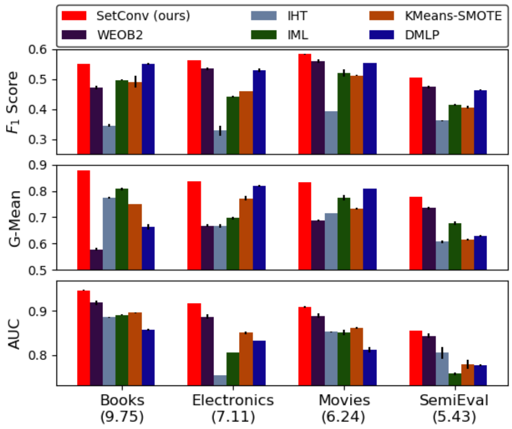

Figure 7: Binary sentiment classification performance of competing methods on the Amazon Review and SemiEval datasets. The value in the bracket indicates the imbalance ratio (IR).

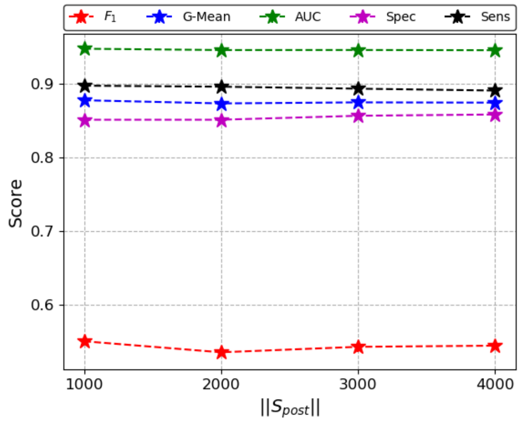

Figure 8: Effect of post-training subset size ($||S_{post}||$) on classification performance.

Table 4: Multi-class classification performance of competing methods on the IRT-NYC dataset. 0.000 indicates a value less than 0.0005.

Fire					
	F_1	G-Mean	AUC	Spec	Sens
IHT	0.601±0.000	0.866±0.002	0.947±0.001	0.841±0.002	0.891±0.005
KMeans-SMOTE	0.831±0.005	0.894±0.003	0.967±0.001	0.978±0.001	0.818±0.005
IML	0.889±0.001	0.947±0.002	0.987±0.002	0.978±0.001	0.917±0.002
CS-DMLP	0.931±0.004	0.951±0.006	0.998±0.001	**0.993±0.004**	0.911±0.016
SetConv (ours)	**0.972±0.002**	**0.996±0.000**	**0.999±0.000**	0.992±0.001	**0.999±0.001**
Shooting					
	F_1	G-Mean	AUC	Spec	Sens
IHT	0.333±0.001	0.471±0.002	0.984±0.001	**0.997±0.001**	0.222±0.002
KMeans-SMOTE	0.895±0.002	0.969±0.003	0.962±0.001	0.996±0.002	0.944±0.003
IML	0.688±0.001	0.780±0.002	0.986±0.001	0.996±0.001	0.611±0.002
CS-DMLP	0.822±0.002	0.910±0.029	0.994±0.002	0.995±0.002	0.883±0.006
SetConv (ours)	**0.912±0.012**	**0.998±0.003**	**0.999±0.001**	0.995±0.001	**0.999±0.001**
Crash					
	F_1	G-Mean	AUC	Spec	Sens
IHT	0.306±0.023	0.762±0.019	0.865±0.011	0.755±0.020	0.769±0.019
KMeans-SMOTE	0.633±0.009	0.802±0.016	0.920±0.014	0.955±0.011	0.673±0.019
IML	0.662±0.002	0.937±0.003	0.959±0.001	0.932±0.001	0.942±0.003
CS-DMLP	0.702±0.054	0.917±0.002	0.969±0.013	0.951±0.017	0.885±0.019
SetConv (ours)	**0.931±0.013**	**0.977±0.001**	**0.997±0.001**	**0.992±0.002**	**0.962±0.001**

minority classes.

In most cases, the overall classification performance of our method is superior to that of competing methods in terms of F_1, G-Mean and AUC metrics. Although CS-DMLP may provide better overall performance than our method in few cases, it achieves that by making many false negative predictions and missing lots of minority class samples, which is undesired in practical applications.

4.6 Sensitivity Analysis

The main parameter in our algorithm is the size of post training subset, i.e., $||S_{post}||$. We vary $||S_{post}||$ from 1000 to 4000 to study its effect on the classification performance. As shown in Figure 8, our method performs stably with respect to different values of $||S_{post}||$. It demonstrates that the SetConv layer has learned to capture the class concepts that are common across different data samples. Thus, as long as $||S_{post}||$ is large enough, e.g., 1000, varying $||S_{post}||$ has little effect on model performance.

5 Conclusion

In this paper, we propose a novel permutation-invariant SetConv operation and a new training strategy named as episodic training for learning from imbalanced class distributions. The combined utilization of them enables extracting the most discriminative features from data and automatically balancing the class distribution for the subsequent classifier. Experiment results demonstrates the superiority of our approach when compared to SOTA methods. Moreover, the proposed method can be easily migrated and applied to data of other types (e.g., images) with few modifications.

Although the performance of SetConv shows its advantage in classification, it may not be appropriate for high-dimensional sparse data. It is because the large amount of 0s in these data may lead to close-to-zero convolution kernels and limit the model's capacity for classification. Combining sparse deep learning techniques with SetConv is a potential solution to this issue. We leave it for future work.

Acknowledgement

This material is based upon work supported by NSF awards DMS-1737978, DGE-2039542, OAC-1828467, OAC-1931541, DGE-1906630, and an IBM faculty award (Research).

References

Sukarna Barua, Md. Monirul Islam, Xin Yao, and Kazuyuki Murase. 2014. Mwmote-majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Trans. Knowl. Data Eng.*, 26(2):405–425.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*. The Association for Computational Linguistics.

Cristiano Leite Castro and Antônio de Pádua Braga. 2013. Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data. *IEEE Trans. Neural Netw. Learning Syst.*, 24(6):888–899.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res.*, 16:321–357.

Chao Chen and Mei-Ling Shyu. 2011. Clustering-based binary-class classification for imbalanced data sets. In *Proceedings of the IEEE International Conference on Information Reuse and Integration, IRI 2011, 3-5 August 2011, Las Vegas, Nevada, USA*, pages 384–389. IEEE Systems, Man, and Cybernetics Society.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

David Díaz-Vico, Aníbal R. Figueiras-Vidal, and José R. Dorronsoro. 2018. Deep mlps for imbalanced classification. In *2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13, 2018*, pages 1–7. IEEE.

Alberto Fernández, Victoria López, Mikel Galar, María José del Jesús, and Francisco Herrera. 2013. Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowl.-Based Syst.*, 42:97–110.

Mikel Galar, Alberto Fernández, Edurne Barrenechea Tartas, Humberto Bustince Sola, and Francisco Herrera. 2012. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Trans. Systems, Man, and Cybernetics, Part C*, 42(4):463–484.

Haibo He and Edwardo A. Garcia. 2009. Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.*, 21(9):1263–1284.

Ruining He and Julian J. McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 507–517. ACM.

Kuan-Hao Huang and Hsuan-Tien Lin. 2017. Cost-sensitive label embedding for multi-label classification. *Machine Learning*, 106(9-10):1725–1746.

Zhouyuan Huo, Feiping Nie, and Heng Huang. 2016. Robust and effective metric learning using capped trace norm: Metric learning via capped trace norm. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1605–1614.

Bartosz Krawczyk. 2016. Learning from imbalanced data: open challenges and future directions. *Progress in AI*, 5(4):221–232.

Felix Last, Georgios Douzas, and Fernando Bação. 2017. Oversampling for imbalanced learning based on k-means and SMOTE. *CoRR*, abs/1711.00837.

Yi-Fan Li, Yang Gao, Gbadebo Ayoade, Hemeng Tao, Latifur Khan, and Bhavani M. Thuraisingham. 2019. Multistream classification for cyber threat data with heterogeneous feature space. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2992–2998. ACM.

Stephan Rabanser, Oleksandr Shchur, and Stephan Günnemann. 2017. Introduction to tensor decompositions and their applications in machine learning. *CoRR*, abs/1711.10781.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17, Vancouver, Canada. Association for Computational Linguistics.

Axel Schulz, Christian Guckelsberger, and Frederik Janssen. 2017. Semantic abstraction for generalization of tweet classification: An evaluation of incident-related tweets. *Semantic Web*, 8(3):353–372.

Michael R. Smith, Tony R. Martinez, and Christophe G. Giraud-Carrier. 2014. An instance level analysis of data complexity. *Machine Learning*, 95(2):225–256.

Parinaz Sobhani, Herna Viktor, and Stan Matwin. 2014. Learning from imbalanced data using ensemble methods and cluster-based undersampling. In *New*

Frontiers in Mining Complex Patterns - Third International Workshop, NFMCP 2014, Held in Conjunction with ECML-PKDD 2014, Nancy, France, September 19, 2014, Revised Selected Papers, volume 8983 of *Lecture Notes in Computer Science*, pages 69–83. Springer.

Yanmin Sun, Mohamed S. Kamel, Andrew K. C. Wong, and Yang Wang. 2007. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378.

Isaac Triguero, Mikel Galar, Sarah Vluymans, Chris Cornelis, Humberto Bustince, Francisco Herrera, and Yvan Saeys. 2015. Evolutionary undersampling for imbalanced big data classification. In *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015*, pages 715–722. IEEE.

Nan Wang, Xibin Zhao, Yu Jiang, and Yue Gao. 2018. Iterative metric learning for imbalance data classification. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 2805–2811. ijcai.org.

Shuo Wang, Leandro L. Minku, and Xin Yao. 2015. Resampling-based ensemble methods for online class imbalance learning. *IEEE Trans. Knowl. Data Eng.*, 27(5):1356–1368.

Fei Wu, Xiao-Yuan Jing, Shiguang Shan, Wangmeng Zuo, and Jing-Yu Yang. 2017. Multiset feature learning for highly imbalanced data classification. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 1583–1589. AAAI Press.

Dandan Yan, Youlong Yang, and Benchong Li. 2017. An improved fuzzy classifier for imbalanced data. *Journal of Intelligent and Fuzzy Systems*, 32(3):2315–2325.

Zhuoyuan Zheng, Yunpeng Cai, and Ye Li. 2015. Oversampling method for imbalanced classification. *Computing and Informatics*, 34(5):1017–1037.

Appendix

Hardware Configuration

All experiments are performed on a server with the following hardware configuration: (1) 1 Intel Core i9-7920X 2.90 GHZ CPU with a total of 24 physical CPU cores (2) 4 GeForce GTX 2080 TI GPU with 11 GB video memory (3) 126 GB RAM. (4) Ubuntu 16.04 and a 4.15.0-39-generic Linux kernel.

Proof of Permutation Invariant Property

An important property of the SetConv layer is permutation-invariant, i.e., it is insensitive to the order of input samples. As long as the input samples are same, no matter in which order they are sent to the model, the SetConv layer always produces the same feature representation.

To prove it, let's consider an arbitrary permutation matrix π. Our goal is to show that $SetConv(\pi X) = SetConv(X)$.

$$
\begin{aligned}
&SetConv(\pi X) \\
&= \frac{1}{N}\Big(Concat(\pi X) \cdot \big[g_1(Y - \pi X) \circledast g_2(W)\big]\Big) \\
&= \frac{1}{N} \cdot \\
&\Big(Concat(X)E(\pi) \cdot \big[(\pi \cdot g_1(Y - X)) \circledast g_2(W)\big]\Big) \\
&= \frac{1}{N}\Big(X'E(\pi) \cdot E[\pi]^T \big[g_1(Y - X) \circledast g_2(W)\big]\Big) \\
&= \frac{1}{N}\Big(X' \cdot I \cdot \big[g_1(Y - X) \circledast g_2(W)\big]\Big) \\
&= SetConv(X)
\end{aligned}
\tag{8}
$$

Here $Concat$ is the concatenation operation which transforms a N-by-d matrix into a Nd-dimensional row vector. $E(\pi)$ is the expansion operator for the permutation matrix π. For example, considering a 2-by-2 permutation matrix,

$$\pi = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$E(\pi)$ is given by:

$$E(\pi) = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

For a toy example, $Concat(\pi X)$ is computed as below:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a & a \\ b & b \end{bmatrix} = \begin{bmatrix} b & b \\ a & a \end{bmatrix} \rightarrow \begin{bmatrix} b & b & a & a \end{bmatrix}$$

On the other hand, $Concat(X)E(\pi)$ is given by

$$
\begin{aligned}
&Concat(X)E(\pi) \\
&= \begin{bmatrix} a & a & b & b \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} b & b & a & a \end{bmatrix} \\
&= Concat(\pi X)
\end{aligned}
$$