Appendices

A Sequence to Sequence model

A family of our models are built on sequence to sequence models. A sequence to sequence (Seq2seq) model is an encoder-decoder model. The encoder encodes the input sequence to an internal representation called the 'context vector' which is used by the decoder to generate the output sequence. Usually, each cell in the Seq2seq model is a Long Short-Term Memory (LSTM) cell [Hochreiter and Schmidhuber1997] or a Gated Recurrent Unit (GRU) cell [Cho *et al.*2014].

Given input sequence $x_1, ..., x_T$, in order to predict output $Y_1^P, ..., Y_K^P$ (where the superscript P denotes 'predicted'), the Seq2seq model estimates conditional probability $P(Y_1^P, ..., Y_t^P | x_1, ..., x_T)$ for $1 \leq t \leq K$. At each time step t, the encoder updates the hidden state h_t^e, which can also include the cell state, by $h_t^e = f_h^e(x_t, h_{t-1}^e)$, where $1 \leq t \leq T$. The decoder updates the hidden state h_t^d by $h_t^d = f_h^d(Y_{t-1}^P, h_{t-1}^d, h_T^e)$, where $1 \leq t \leq K$. The decoder generates output y_t by

$$y_t = g(Y_{t-1}^P, h_t^d, h_T^e), \tag{1}$$

and $Y_t^P = q(y_t)$ with q usually being softmax function.

The model calculates the conditional probability of output $Y_1^P, ..., Y_K^P$ by $\Pr(Y_1^P, ..., Y_K^P | x_1, ..., x_T) = \prod_{t=1}^K \Pr(Y_t^P | Y_1^P, ..., Y_{t-1}^P)$.

B End to End Memory Networks

The other family of our models are built on an end-to-end memory network (MemN2N). This model takes $x_1, ..., x_n$ as the external memory, a 'query' x, a ground truth Y^{GT} and predicts an answer Y^P. It first embeds memory vectors $x_1, ..., x_n$ and query x into continuous space. They are then processed through multiple layers to generate the output label Y^P.

MemN2N has K layers. In the t^{th} layer, where $1 \leq t \leq K$, the external memory is converted into embedded memory vectors $m_1^t, ..., m_n^t$ by an embedding matrix A^t. The query x is also embedded as u^t by an embedding matrix B^t. The attention scores between embedded query u^t and memory vectors $(m_i^t)_{i=1,2,...,n}$ are calculated by $p^t = softmax((u^t)^T m_1^t, (u^t)^T m_2^t, ..., (u^t)^T m_n^t)$.

Each x_i is also embedded to an output representation c_i^t by another embedding matrix C^t. The output vector from the external memory is defined as $o^t = \sum_{i=1}^{n} p_i^t c_i^t$. By a linear mapping H, the input to the next layer is calculated by $u^{t+1} = Hu^t + o^t$.

In the last layer, by another embedding matrix W, MemN2N generates a label for the query x by $Y^P = softmax(W(Hu^K + o^K))$.

ALGORITHM 1: Out-of-core framework

```
for epoch = 1,...,T do
    for training sample x do
        Let X^T = ∅;
        for r = 1 to R do
            Randomly draw B samples from training set;
            U = nearest K samples to x among the B samples;
            Let X^T be the nearest K samples to x among U ∪ X^T;
        end
        Update parameters by a gradient iteration:
        Θ^R = Θ^R − α∇L̃(x, Y^GT, X^T, Y^T, Θ^R);
    end
end
```

C Out-of-core Algorithm

D Comparison with Other Related Benchmark Models

We first compare V2VSLS with the popular ranking-based model LambdaRank [Burges *et al.*2006] with lightGBM as the classifier. LambdaRank ranks the feature vectors in the training set for a given query by similarity. In inference, for a provided feature vector query, LambdaRank ranks the feature vectors in the training set and obtains the nearest $K = 5$ samples. Finally, the label of the query is obtained by majority voting among the corresponding labels of those nearest samples in the training set. The results are provided in Table 1. The performance improvement of approximately 40% is observed in the experiment, which shows the significant superiority of V2VSLS over the conventional ranking-based method.

To compare our work with another similar work which utilizes the nearest neighbors to make predictions, we implemented the kNN-Augmented Networks from [Wang *et al.*2017]. The comparison is shown in Table 1. In the implementation of kNN-Augmented Networks, we have fine-tuned the hyper-parameters: $K = 5$, $I = 8$, learning rate=0.001 and the Adam optimizer. There is a substantial gap between our models and the kNN-Augmented Networks, that we were unable to close despite a significant effort to fine tune the hyper-parameters.

E Synthetic Samples Visualization

Figure 1 shows a t-SNE [van der Maaten and Hinton2008] visualization of the original set and the oversampled set, using SensIT dataset, projected onto 2-D space. Although SMOTE and ADASYN overall perform well, their class boundaries are not as clean as those obtained by V2VSLS.

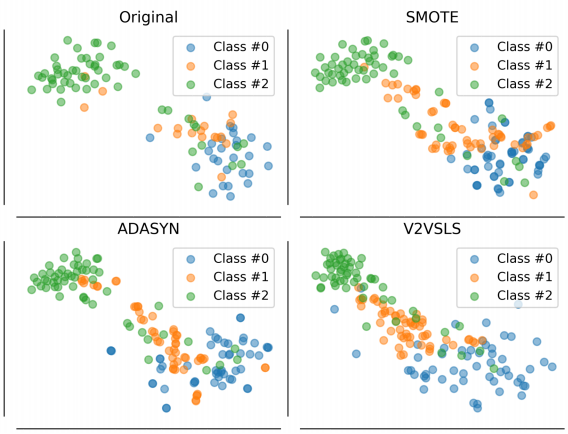

Figure 1: Visualization of oversampling methods.