CLE-SMOTE: Addressing Extreme Imbalanced Data Classification with Contrastive Learning-Enhanced SMOTE

Cara Lee, Faisal Nabulsi, Michael Xu, Christopher Kan, Andrew Kan, Rachel Yun, Bryan Jiang, Aiden Yun, Rana Suleimen, Talal Nabulsi, Isam Kharouf, Zaid Nabulsi

Abstract

Synthetic Minority Oversampling Technique (SMOTE) is a widely used oversampling method for addressing class imbalance by generating synthetic minority class examples. While effective, SMOTE occasionally introduces harmful examples into the dataset, hindering model performance. In this work, we introduce Contrastive Learning-Enhanced SMOTE (CLE-SMOTE), a method to identify and reduce the influence of these noisy SMOTE-generated examples. In our experiments on imbalanced datasets, CLE-SMOTE achieves promising results, substantially outperforming all baselines, including vanilla SMOTE, and approaching the performance of an equivalent network trained on a balanced dataset.

1 Introduction

In a wide array of machine learning tasks, the challenge of predictive modeling with imbalanced datasets – characterized by significant disparities in class frequencies – has been a persistent issue (He and Garcia, 2009; Yanminsun et al., 2011; Hoens and Chawla, 2013) impacting numerous areas of research (Nabulsi et al., 2021; Kazemzadeh et al., 2021; Chitturi and Nabulsi, 2021; Kiraly et al., 0; Sheehan et al., 2018; Kosaraju et al., 2019).

Synthetic Minority Oversampling Technique (SMOTE), a prominent oversampling algorithm, addresses class imbalance by generating synthetic minority class examples through interpolation between randomly chosen minority class neighbors (Chawla et al., 2002). While SMOTE generally performs well in practice, it can occasionally introduce noisy or unhelpful synthetic examples, thereby negatively impacting the learning process (Batista et al., 2004) (see Supplementary Figure 5). In this work, we present Contrastive Learning-Enhanced SMOTE (CLE-SMOTE), a method that supplements SMOTE by capping the influence of noisy, synthetically-generated examples on the network's training and allows for learning robust class representations despite severe class imbalance.

2 CLE-SMOTE

CLE-SMOTE comprises two stages: a pretraining stage and a supervised finetuning (SFT) stage. Before any training, we use SMOTE to generate synthetic examples of the minority class(es). More details are provided in Appendix B.2. A schematic of CLE-SMOTE is illustrated in Supplementary Figure 1.

2.1 Pretraining Stage

The goal of the pretraining stage is to enable the network to learn distinct feature spaces for each of the classes. This is beneficial in two ways: (1) it allows the network to distinguish between the different classes more effectively than in a traditional supervised learning fashion; and more importantly, (2) by learning distinct feature spaces for the minority class(es), our network can identify out-of-distribution, noisy, SMOTE-generated examples and cap their influence on parameter updates. To learn distinct feature spaces for each of our classes, we employ supervised contrastive learning and train on the entire SMOTE-augmented dataset (Khosla et al., 2021). More details are provided in Appendix A.1

2.2 Supervised Finetuning Stage

With the network pretrained using supervised contrastive learning, we add a single linear linear layer on top of the pretrained embedding network and finetune the entire network using cross-entropy loss. To cap the influence of potentially harmful SMOTE-generated examples, the embeddings (the output of the second-to-last layer of the network) for each non-SMOTE example are calculated at the start of each epoch. We compute the average embedding for each class and cap the loss for each smote example using the following formula:

$$L_{smote\_example} = min(ce\_loss, \frac{\beta}{cosine\_distance(\mu_i, smote\_embedding)}) \tag{1}$$

where ce_loss is the regular cross-entropy loss for the example, μ_i is the average embedding for the minority class the SMOTE example belongs to (computed at the start of each epoch), and β is the noise tolerance hyperparameter that regulates the extent to which noisier synthetic examples are capped. See Appendix A.2 for more details.

3 Experiments and Results

To benchmark our method, we ran classification experiments on the CIFAR-10 dataset (Krizhevsky, 2009). We curated two imbalanced subsets of the dataset: one binary (2-class, 100:1 ratio in the training set) and one multiclass (3-class, 200:20:1 ratio in the training set). See Appendix B.1 for more details. We compared our approach to the following baselines: No Method (nothing done to counter the class imbalance), Oversampling Minority Class, and Vanilla SMOTE. More details are provided in Appendix C.3. Results are shown in Table 1. Briefly, CLE-SMOTE outperformed all baselines and its performance approached that of a network trained on a balanced dataset. CLE-SMOTE's robust performance demonstrates its ability to learn despite the presence of noisy data. Our method can be extended to other scenarios where learning with noisy data/labels is required.

Method	Binary AUROC (Variance)	Multiclass AUROC (Variance)
No Method	0.824 (0.00145)	0.673 (0.000414)
Oversampling Minority Class	0.815 (0.00411)	0.727 (0.00125)
Vanilla SMOTE	0.829 (0.00230)	0.691 (0.00415)
CLE-SMOTE	0.876 (0.000220)	0.753 (0.00108)
No Class Imbalance	0.903 (0.00150)	0.8332 (0.0124)

Table 1: AUROC results from baseline and CLE-SMOTE experiments.

.

References

Abien Fred Agarap. Deep learning using rectified linear units (relu), 2019.

Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1):20–29, 2004.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, June 2002. ISSN 1076-9757. doi: 10.1613/jair.953. URL `http://dx.doi.org/10.1613/jair.953`.

Varun Chitturi and Zaid Nabulsi. Predicting poverty level from satellite imagery using deep neural networks. *CoRR*, abs/2112.00011, 2021. URL `https://arxiv.org/abs/2112.00011`.

Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009. doi: 10.1109/TKDE.2008.239.

T. Ryan Hoens and Nitesh V. Chawla. *Imbalanced Datasets: From Sampling to Classifiers*, chapter 3, pages 43–59. John Wiley Sons, Ltd, 2013. ISBN 9781118646106. doi: https://doi.org/10.1002/9781118646106.ch3. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118646106.ch3`.

Sahar Kazemzadeh, Jin Yu, Shahar Jamshy, Rory Pilgrim, Zaid Nabulsi, Christina Chen, Neeral Beladia, Charles Lau, Scott Mayer McKinney, Thad Hughes, Atilla Kiraly, Sreenivasa Raju Kalidindi, Monde Muyoyeta, Jameson Malemela, Ting Shih, Greg S. Corrado, Lily Peng, Katherine Chou, Po-Hsuan Cameron Chen, Yun Liu, Krish Eswaran, Daniel Tse, Shravya Shetty, and Shruthi Prabhakara. Deep learning for detecting pulmonary tuberculosis via chest radiography: an international study across 10 countries, 2021.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning, 2021.

Atilla P. Kiraly, Corbin A. Cunningham, Ryan Najafi, Zaid Nabulsi, Jie Yang, Charles Lau, Joseph R. Ledsam, Wenxing Ye, Diego Ardila, Scott M. McKinney, Rory Pilgrim, Yun Liu, Hiroaki Saito, Yasuteru Shimamura, Mozziyar Etemadi, David Melnick, Sunny Jansen, Greg S. Corrado, Lily Peng, Daniel Tse, Shravya Shetty, Shruthi

Prabhakara, David P. Naidich, Neeral Beladia, and Krish Eswaran. Assistive ai in lung cancer screening: A retrospective multinational study in the united states and japan. *Radiology: Artificial Intelligence*, 0(ja):e230079, 0. doi: 10.1148/ryai.230079. URL `https://doi.org/10.1148/ryai.230079`. PMID: 38477661.

Vineet Kosaraju, Yap Dian Ang, and Zaid Nabulsi. Faster transformers for document summarization. *Vineet Kosaraju*, 2019.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Yassine Marrakchi, Osama Makansi, and Thomas Brox. Fighting class imbalance with contrastive learning. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2021. URL `https://api.semanticscholar.org/CorpusID:236513238`.

Zaid Nabulsi, Andrew Sellergren, Shahar Jamshy, Charles Lau, Edward Santos, Atilla P. Kiraly, Wenxing Ye, Jie Yang, Rory Pilgrim, Sahar Kazemzadeh, Jin Yu, Sreenivasa Raju Kalidindi, Mozziyar Etemadi, Florencia Garcia-Vicente, David Melnick, Greg S. Corrado, Lily Peng, Krish Eswaran, Daniel Tse, Neeral Beladia, Yun Liu, Po-Hsuan Cameron Chen, and Shravya Shetty. Deep learning for distinguishing normal versus abnormal chest radiographs and generalization to two unseen diseases tuberculosis and covid-19. *Scientific Reports*, 11(1), September 2021. ISSN 2045-2322. doi: 10.1038/s41598-021-93967-2. URL `http://dx.doi.org/10.1038/s41598-021-93967-2`.

Evan Sheehan, Zaid Nabulsi, and Chenlin Meng. Utilizing latent embeddings of wikipedia articles to predict poverty. *Stanford University*, 2018.

Mayuri S Shelke, Prashant R Deshmukh, and Vijaya K Shandilya. A review on imbalanced data handling using undersampling and oversampling technique. *Int. J. Recent Trends Eng. Res*, 3(4):444–449, 2017.

Yanminsun, Andrew Wong, and Mohamed S. Kamel. Classification of imbalanced data: a review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23, 11 2011. doi: 10.1142/S0218001409007326.

Appendix A. CLE-SMOTE Details

A.1 Pretraining

In the pretraining stage, we employ representation learning to allow the network to learn feature spaces for the different classes. Representation learning is designed to unravel the complexity of the feature space so that the derived embeddings align effectively with various classes. However, when working with datasets that exhibit a significant imbalance, networks are typically biased towards majority classes because they predominantly influence the loss. To address this challenge, our approach involves using supervised contrastive learning (Khosla et al., 2021) to distinguish between minority and majority classes within the feature space.

In the pretraining stage, our goal is twofold. Firstly, we aim to allow the network to learn distinct feature spaces for the classes despite class imbalance, a task more effectively accomplished by contrastive learning than standard supervised learning (Marrakchi et al., 2021). Secondly, because our aim is to cap the influence of noisy, synthetically-generated examples on training during the second stage, learning feature spaces for each class will aid in the detection of these outliers.

Specifically, for each example in a batch from our training set (referred to as the anchor), we randomly choose a positive example (an example belonging to the same class as the anchor) and a negative example (an example belonging to a different class than the anchor). We compute the Euclidean distance between the anchor and both the positive and negative examples, and use the following loss:

$$L = max(0, EUC(A, P) - EUC(A, N)) \tag{2}$$

where EUC is the euclidean distance, A is the embedding of the anchor, P is the embedding of the positive example, and N is the embedding of the negative example. We confine the selection of positive and negative examples to solely originate from non-SMOTE examples.

At the end pf the pretraining stage, our network will have learned separate feature spaces for each of our classes which will be useful in identifying and capping potentially harmful SMOTE-generated data points.

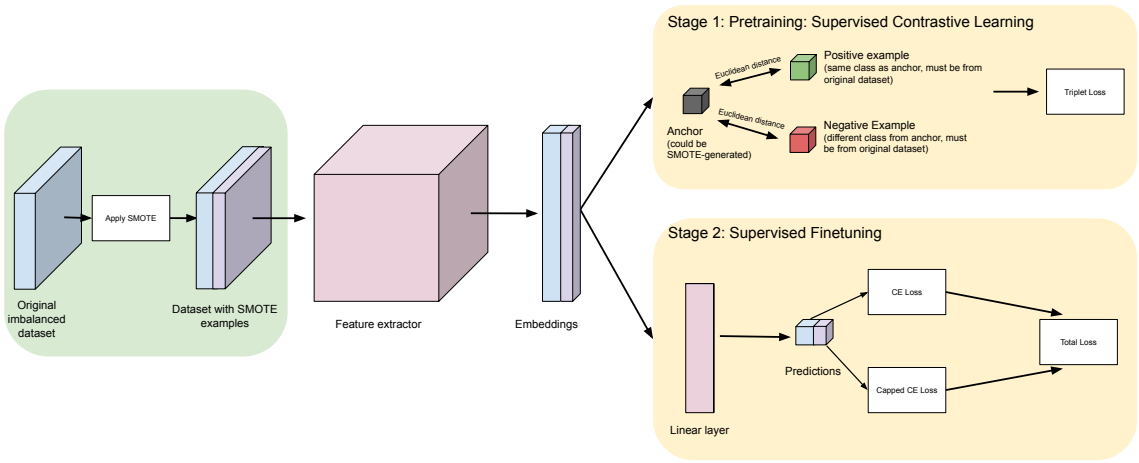
A.2 Supervised Finetuning

The supervised finetuning stage involves the addition of a single linear layer to the pretrained embedding network of the previous stage. We train the whole network on the entire dataset, including SMOTE-generated examples, using a customized cross-entropy loss (Equation 1). At the start of every epoch in this stage, we compute the embeddings for each non-SMOTE example and calculate the average non-SMOTE embedding for each class. We do this at the start of every epoch because the entire network's parameters are updated, meaning the learned feature spaces for each class shift throughout the training process.

We then implement a capping mechanism to address noisy and ineffective SMOTE-generated examples. We compute the cosine distance between each SMOTE example and the average embedding for its class. A higher cosine distance signifies a more substantial deviation from the feature space of its class, indicating that the example is dissimilar from

the rest of its class. Our loss function is designed to limit the influence of these out-of-distribution examples on parameter updates. For non-SMOTE examples, vanilla cross-entropy loss (no capping) is used.

The primary feature of the SMOTE loss function is *beta*, the noise tolerance hyperparameter which controls the extent to which we cap the SMOTE-generated examples. A higher *beta* signifies a higher noise tolerance and less aggressive capping.



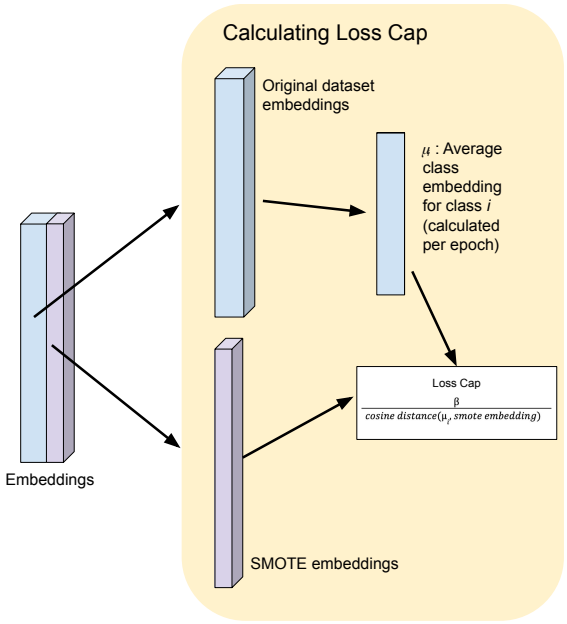
Supplementary Figure 1: CLE-SMOTE model architecture and training procedure. We first augment our dataset using SMOTE. Then, in the first stage, we pretrain the feature extractor using triplet loss. Once the network is pretrained, we add a linear layer on top of the embeddings generated from the first stage and train the entire network using cross-entropy loss for non-SMOTE examples and capped cross-entropy loss for SMOTE examples.

Appendix B. Dataset Details

B.1 Imbalanced Dataset Curation

To assess the effectiveness of our method on learning from imbalanced datasets, we constructed two imbalanced training datasets derived from CIFAR-10 (Krizhevsky, 2009). The first dataset is binary, comprising only two classes with a training set ratio of 100:1. The second dataset consists of three classes with a ratio of 200:20:1. Additionally, we compiled 2 balanced training sets, which are identical to the imbalanced datasets but with increased data in the minority classes. These balanced datasets allow us to train an equivalent network without class imbalance as a benchmark.

We similarly curated 2 evaluation datasets, with the first comprising the same binary classes and the second consisting of the same three classes as the training sets. There is no class imbalance in the evaluation sets.



Supplementary Figure 2: CLE-SMOTE's method for calculating the loss cap for SMOTE examples. This is implemented in the capped loss function for SMOTE examples during supervised finetuning step. Every epoch, we compute the embeddings (the output of the second-to-last layer of the network) for every non-SMOTE examples, and take the average non-SMOTE embedding for each class. During training, for every SMOTE example, we cap the loss using the shown formula.

B.2 Applying SMOTE

Before training, we apply SMOTE (Chawla et al., 2002) to generate examples from the minority class(es) such that the dataset is balanced, leaving the majority class untouched. We set the parameter for nearest neighbors to k=5, determined empirically.

Appendix C. Experimental Details

C.1 Experimental Procedures

To assess CLE-SMOTE's effectiveness in capping the influence of noisy, synthetically-generated training examples, we train CLE-SMOTE on the imbalanced datasets described in Appendix B.1 and evaluate its performance on the balanced test sets. To ensure a more robust evaluation, we utilize both binary and mutliclass formulations.

C.2 Evaluation

Because of run-to-run variance in performance of the experiments, we run each experiment 10 times and take the mean and variance of the area under the receiver operating curve (AUROC) for each. For the 3-class experiments, we take the average one-vs-all AUROC for all three classes. For each experiment, we report a single AUROC and variance.

We train each experiment for 30 epochs and evaluate on the test set every 10 epochs, comparing the best AUROC.

C.3 Baselines

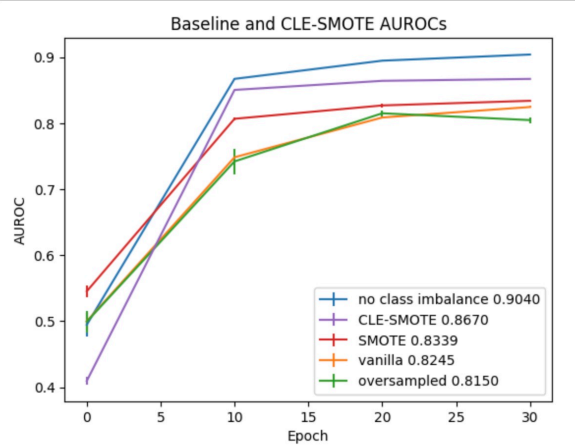
We benchmark CLE-SMOTE's performance against the following baselines:

- **Oversampling**: Overampling the minority class is a widely used method when training on imbalanced datasets (Shelke et al., 2017). We randomly oversample all minority classes to eliminate the class imbalance.
- **Vanilla SMOTE**: Here, we apply SMOTE with no adjustments, as described in the paper.
- **No Method**: Here, we apply no special methods when training the network on the imbalanced datasets.
- **No Class Imbalance**: Here, we apply no special methods and train the network on the balanced datasets (see Appendix B.1).

Results are shown in Table 1. In addition to the baseline methods discussed, we attempted to utilize undersampling as an alternative method; however, the experiments did not converge, and thus, the results are not reported.

C.4 Architecture & Hyperparameters

For all of our experiments, we use the same architecture. We use a convolutional neural network with two convolutional layers (each followed by max pooling) and two fully



Supplementary Figure 3: Plot showing AUROC and Variance over the duration of training during for our experiments. CLE-SMOTE out-performs all other experiments on imbalanced data, approaching the performance of a network trained with no imbalance. CLE-SMOTE also trains faster than all baselines, converging sooner.

connected layers, with a ReLU activation (Agarap, 2019) block after each layer. During training, we employ dropout after the second convolutional layer. We update the weights using stochastic gradient descent.

In our experiment, we set β to 1.0. For each method, we ran a search to find the optimal learning rate. We then recorded the AUROC and variance at that learning rate.

C.5 Ablation Study

Before arriving at our current version of CLE-SMOTE, we conducted an ablation study to explore alternative methods of capping the loss of SMOTE-generated examples. Here are the formulations we tried:

- **Constant Cap**: We employ a constant cap for all SMOTE-generated examples on the loss. Empirically, we determined the optimal cap to be 5 and 1 for binary and multiclass formulations respectively.

- **Class Avg Cosine Distance Cap**: We utilize the same capping function as CLE-SMOTE (Equation 1), but we do not pretrain the network.

- **Batch Avg Cosine Distance Cap**: The same as Class Avg Cosine Distance Cap, but in lieu of the average class embedding in the entire dataset, we compute the average class embedding in the current batch.

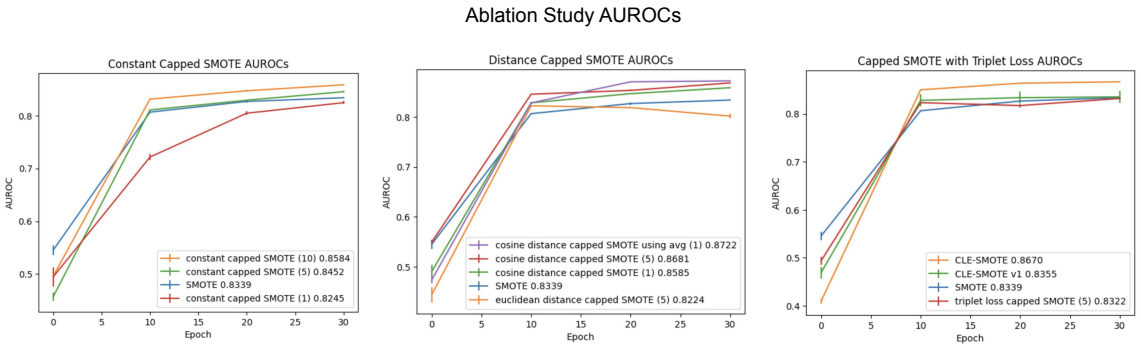
- **Euclidean Distance Cap**: The same as Class Avg Cosine Distance Cap, but in lieu of cosine distance, we use euclidean distance..

Method (with SMOTE)	Binary AUROC (Variance)	Multiclass AUROC (Variance)
Constant Cap	0.858 (0.00138)	0.727 (0.000580)
Class Avg Cosine Distance Cap	0.856 (0.00102)	0.744 (0.000493)
Batch Avg Cosine Distance Cap	0.868 (0.000354)	0.712 (0.00145)
Euclidean Distance Cap	0.822 (0.000923)	0.664 (0.000679)
CLE-SMOTE v1	0.835 (0.0126)	0.523 (0.00549)
CLE-SMOTE	0.876 (0.000220)	0.753 (0.00108)

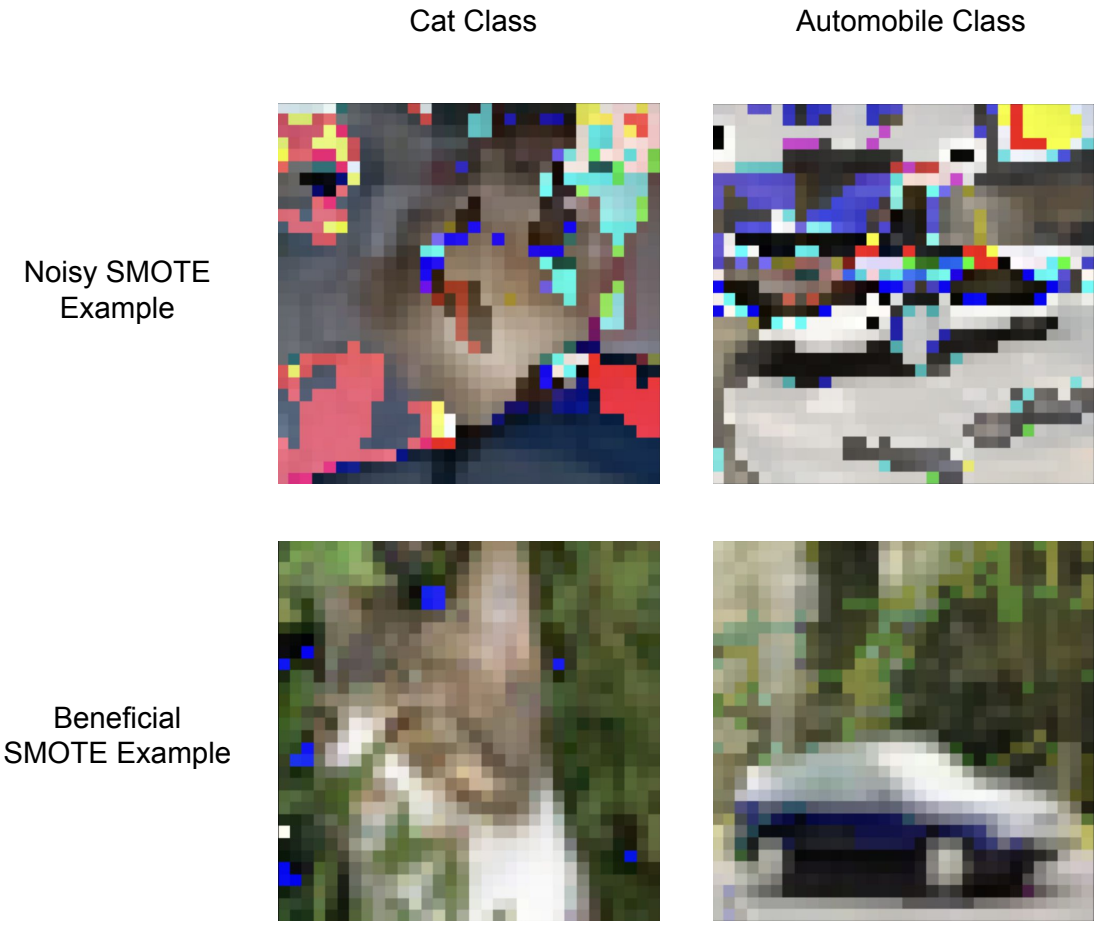
Supplementary Table 2: AUROC results from our ablation study.

- **CLE-SMOTE v1** : In the initial version of CLE-SMOTE, we excluded SMOTE-generated examples from being used as anchors during the pretraining stage.

Results for our ablation study are shown in Supplementary Table 2.



Supplementary Figure 4: Ablation study AUROCs and variances during the first 30 epochs of training. The leftmost graph shows results from constant capped SMOTE experiments. The center graph shows results from cosine and euclidean distance capped SMOTE experiments. The rightmost graph shows results from triplet loss capped SMOTE experiments and versions of CLE-SMOTE.



Supplementary Figure 5: Examples of beneficial and noisy SMOTE examples generated from the CIFAR-10 dataset.