Federated Learning with Metric Loss

Hyunsin Park [*1] **Hossein Hosseini** [*2] **Sungrack Yun** [*1]

Abstract

We consider federated learning of an embedding-based classifier with a metric loss, where each client has access to the data of only one class and cannot share embeddings with the server or other clients. In this setting, each client may not optimize the negative loss term which discriminates the embeddings between different clients. Training only with positive loss causes all embeddings to collapse into a single point. To address this problem, we propose Federated Metric Learning (FedMetric), a framework in which clients update their local models with a metric learning loss to minimize intra-class variance and maximize inter-class variance using proxy centers that are shared by other clients in a randomized way. Our initial experiments show the effectiveness of FedMetric on MNIST and CIFAR-10 datasets.

1. Introduction

The problem of training embedding networks has been widely studied due to its applicability to various tasks such as identification, verification, retrieval, and clustering (Snyder et al., 2017; Yun et al., 2019; Wang et al., 2018a; Cao & Jain, 2018; Nguyen et al., 2017; Schroff et al., 2015; Mikolov et al., 2013). Such networks are usually trained with a loss function that simultaneously minimizes the distance of instance embeddings of the same class and maximizes the distance of instance embeddings from different classes. In recent years, deep neural networks trained with large data have been used to obtain nonlinear embeddings (Zhong & Deng, 2019; Guan et al., 2020; Ge & Moh, 2017; Chung et al., 2018). Collecting large and high-quality data to train deep networks is, however, generally expensive for real-world applications (Zhao et al., 2017; Zhu et al., 2017; Zhang et al., 2020).

One approach to address the data collection problem is to train the model in the federated setup, in which a global model is iteratively updated by aggregating local models without the need to have direct access to local data (McMahan et al., 2017; Bonawitz et al., 2019; Konečnỳ et al., 2016; Sattler et al., 2019; Karimireddy et al., 2020). We consider a setting where each client has access to the data of only one target class and cannot share the embeddings with the server or other clients. In this setting, the clients can train the model using only their own embedding vector and, thus, cannot compute the negative loss term that depends on the embedding vectors of other users. Training with only the positive loss function, however, cause all embeddings to collapse into a single point.

The problem of training embedding networks in federated setup has been recently explored in different settings. Federated Averaging with Spreadout (FedAwS) (Yu et al., 2020) learns an embedding network for multi-class classification in the federated setup, where each client has access to only positive labels. In this method, the client embeddings are shared with the server, where a regularization terms is applied to increase the pairwise distances between embeddings. Another recent approach is Federated User Verification (FedUV) (Hosseini et al., 2021), which proposed to use predefined codewords with maximum pairwise distance as embeddings and, thus, does not require sharing client embeddings with the server. FedUV, however, does not take into account the similarity of clients' data in training embeddings which are obtained based on the predefined codewords.

We consider a setting, where clients update their local model with a metric learning loss that minimizes intra-class variance and maximizes inter-class variance. For training local models, clients generate hyperspheres within which all embedding vectors are located. They then train the model to minimize the overlap between the local hyperspheres. This approach requires hypersphere centers to be shared with other users and thus exposes security-sensitive information. We instead propose a method, called Federated Metric Learning (FedMetric), in which clients share the center of a proxy hypersphere that includes the true hypersphere. Here, the proxy hypersphere is set to be bigger than the true

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc. [1]Qualcomm Korea YH [2]Qualcomm Technologies, Inc.. Correspondence to: Hyunsin Park <hyunsinp@qti.qualcomm.com>.

This work was presented at the International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML 2021 (FL-ICML'21). This workshop does not have official proceedings and this paper is non-archival.

hypersphere such that the expose of security-sensitive information is minimized. We show our approach reduces the exposure of the sensitive embeddings to other users, while maintaining the performance of the model. We show the effectiveness of our method on MNIST and CIFAR-10 datasets.

2. Background

2.1. Federated learning

Federated learning (FL) is a method to train a model across distributed edge (client) devices without sharing local data information. During each round of learning process, the server broadcasts current global model to selected clients. After the clients update their local models from the shared global model using local data, local models are uploaded to to server. Finally, the server aggregates the local models to update the global model. The most commonly-used algorithm is Federated Averaging (FedAvg) (McMahan et al., 2017). Many federated learning methods consider classification losses. However, if the class is closely related with the client, sharing the parameters of the output layer for classification with other client is not appropriate since the output parameter contains client-specific information. This paper considers federated learning of an embedding networks using a metric-learning based loss.

2.2. Embedding network learning

Let $x \in \mathcal{X}$ be an input data. An embedding network $g_\theta(\cdot) : \mathcal{X} \to \mathbb{R}^d$ takes x as input and outputs an embedding vector $g_\theta(x)$. This embedding network can be learned with classification losses or metric learning losses. For the classification loss, softmax-based cross entropy loss and margin-based losses (Liu et al., 2017; Wang et al., 2018b; Deng et al., 2019) can be used. For the metric learning loss, contrastive loss (Chopra et al., 2005), triplet loss (Schroff et al., 2015), and prototypical loss (Snell et al., 2017) can be used.

2.3. Learning an embedding network with access to only positive data

DeepSVDD Deep Support Vector Data Description (DeepSVDD) (Ruff et al., 2018) was proposed for anomaly detection. It trains an embedding network by minimizing the volume of a hypersphere that encloses the instance embeddings of the data. Minimizing the volume of the hypersphere forces the network to extract the common factors of variation since the network needs to closely map the data points to the center of the hypersphere. To prevent a hypersphere collapse solution, they use neural networks without bias terms or bounded activation functions. The loss function of DeepSVDD is as follows:

$$d(g_\theta(x_i), C)^2, \tag{1}$$

where $d(\cdot,\cdot)$, x_i, and C are a distance measure, the i-th input, and the target center obtained by average of instance embeddings, respectively.

FedAwS FedAwS (Yu et al., 2020) was proposed for training an embedding network for multi-class classification in the federated setting, where each client has access to the only positive data. The embedding network can be shared among clients. However, class embeddings cannot be shared with other clients. They introduced a regularization to spread out class embeddings at server. The loss function of FedAwS is based on the following contrastive loss:

$$d(g_\theta(x_i), w_y)^2 + \lambda \sum_{c \neq y} (\max\{0, \nu - d(g_\theta(x_i), w_c)\})^2, \tag{2}$$

where y, $\{w_i\}_{i=1}^C$, and ν are the target label, class embeddings, and margin, respectively. The loss function cannot be optimized at each client since each client does not have access to client embeddings of other clients. The second term is replaced with $\sum_{c \in [C']} \sum_{c' \neq c} (\max\{0, \nu - d(w_c, w_{c'})\})^2$. Now this term can be optimized at the server because it is not a function of local data anymore. FedAwS still requires sharing class embedding with the server, which may leads privacy issues.

FedUV FedUV (Hosseini et al., 2021) was proposed to eliminate the requirement of sharing class embeddings with the server. Error correction codewords with known minimum distance are generated and used for class embeddings. In the paper, authors proved that each client can learn a local model using only the positive loss given the spread-out codewords. The loss function of FedUV is as follows:

$$\max\left(0, 1 - \frac{1}{c} v_y^T \sigma(W g_\theta(x_i))\right)^2, \tag{3}$$

where c, v_y, and W are scaling factor, codeword, and linear projection matrix, respectively. However, FedUV has limitation to model the similarity between clients in the embedding space since the codewords are predefined without consideration of local data.

3. Method

3.1. Algorithm

We propose Federated Metric Learning (FedMetric), a framework in which clients update local models with a metric learning loss to minimize intra-class variance and simultaneously maximize inter-class variance with respect to the centers of proxy hyperspheres generated by each of

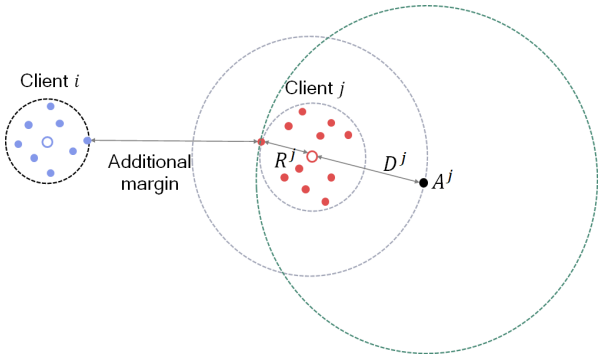

Figure 1. An illustration of the proposed FedMetric: the true hypersphere of client i (black dashed circle) and the proxy hypersphere of client j (green dashed circle) with the center A^j. The FedMetric tries to learn the network such that the distance between any point in client i and A^j should be larger than at least $R^j + D^j$.

the client. Instead of sharing information about true hyperspheres enclosing the instance embeddings of local data, we consider a proxy hypersphere which is bigger enough to include the true hypersphere and also lead to minimize the expose of security-sensitive information, i.e. the local data used to train a local embedding network such that the intra-class variation is minimized. With this proxy hypersphere, we try to maximize the inter-class variations between different clients.

Figure 1 shows an example of true hyperspheres and a proxy hypersphere for FedMetric. We want to minimize overlap between hyperspheres of different classes. However, sharing the true hypersphere leads privacy leakage. In FedMetric, instead of sharing the true hypersphere, a proxy hypersphere characterized by center A^j and radius $R^j + D^j$ including the true hypersphere is shared with client i. Here D^j is the distance between true center and the proxy center, and A^j is generated on the hypersphere with radius D^j (refer the details in Algorithm 1). Now, client i learns its embeddings to be located outside the shared proxy hypersphere.

The proposed FedMetric is based on the following loss function including a positive loss and a negative loss,

$$l(\theta, b) = l_{pos}(\theta, b) + \lambda \times l_{neg}(\theta, b), \tag{4}$$

where b represents a mini-batch of local data, and λ represents a scaling factor for the negative loss.

The positive loss function l_{pos} is generally optimized to minimize intra-class variation. The positive loss function is defined as follows:

$$l_{pos}(\theta, b) = \sum_{i \in b} d(g_\theta(x_i), C^k), \tag{5}$$

where d represents a distance between an embedding $g_\theta(x_i)$ and a C^k representing the center of the local hypersphere.

Algorithm 1 Federated Metric Learning

Initialize global model θ_0
for each global round t **do**
  select M clients
  for each client k of the selected clients **do**
    $\theta_t^k, A_t^k, M_t^k =$ Client Update(θ, $\{A^j\}$, $\{M^j\}$)
    Client Update(θ, $\{A^j\}$, $\{M^j\}$)
    Global mode update by averaging
    $\theta_t = \frac{1}{M} \sum_k \theta_t^k$
  end for
end for
function Client Update(θ, $\{A^j\}$, $\{M^j\}$)
  Calculate local center C^k by averaging the instance embeddings based on the θ
  for local batch b **do**
    $\theta \leftarrow \theta - \eta \nabla l(\theta, b)$
    $l(\theta, b) = l_{pos}(\theta, b) + \lambda \times l_{neg}(\theta, b)$
    $l_{pos}(\theta, b) = \sum_{i \in b} d(g_\theta(x_i), C^k)$
    $l_{neg}(\theta, b) = \sum_{i \in b} \sum_{j \neq k} \max(M^j - d(g_\theta(x_i), A^j), 0)$
  end for
  Calculate local radius: $R^k = \max_i(d(g_\theta(x_i), C^k)$
  Generate proxy center A^k
  $A^k = C^k + X$, where X a random sample from uniform distribution on surface of hyper-sphere characterized by zero center and D^k radius
  RETURN θ, A^k, $(R^k + D^k)$
end function

The negative loss function is generally optimized to maximize inter-class variation. The negative loss is defined as follows:

$$l_{neg}(\theta, b) = \sum_{i \in b} \sum_{j \neq k} (\max\{0, M^j - d(g_\theta(x_i), A^j)\})^2, \tag{6}$$

where $M^j = R^j + D^j$ represents the radius of a proxy hypersphere for the j-th client. It is used as the margin for the negative loss, and there is no loss when an embedding vector is located outside of a proxy hypersphere of the j-th client.

The proposed learning procedure is described in Algorithm 1 based on FedAvg (McMahan et al., 2017). In FedMetric, clients share not only the updated local model but also proxy center and margin with servers and other clients. At the client side, the local center is iteratively updated using the mean of instance embeddings in the local batch. After local model update, a proxy center is generated by adding random sample on a hypersphere with radius D^k and zero center to the local center C^k.

3.2. Security analysis of the shared vector

Given a shared hypersphere with center A and radius $M = R + D$ including the true hypersphere with center C and radius R, we can calculate the probability of the risk exposing the security-sensitive information. This can be calculated by the ratio of true hypersphere volume to the proxy hypersphere volume. The d-dimensional hypersphere volume with R radius is $\frac{\pi^{(d/2)}}{\Gamma(d/2+1)}R^d$. Now, the ratio is $\frac{R^d}{(R+D)^d}$. For example, when $R = 0.1, D = 0.1, d = 128$, the probability is 2.9×10^{-39}. We can easily reduce this probability by increasing the embedding vector dimensionality.

4. Experiments

4.1. Experimental Setup

We evaluate our methods on MNIST (LeCun et al., 2010) and CIFAR-10 (Krizhevsky et al., 2009) datasets. Table 1 shows the statics of the used datasets. To evaluate the performance of the proposed methods, we use the area under the receiver operating characteristic (AUROC) measure.

We train the embedding networks with 1 local iteration and $10,000$ rounds for MNIST and $100,000$ rounds for CIFAR-10. At each round, all clients are participated for global model update. We use embedding networks based on LeNet used in (Ruff et al., 2018) for MNIST and ResNet-32 used in (Yu et al., 2020) for CIFAR-10. In the models, we use Group Normalization (GN) (Wu & He, 2018) instead of batch normalization (BN) (Ioffe & Szegedy, 2015) following the observations that BN does not work well in non-i.i.d. data setting of federated learning (Hsieh et al., 2019). The LeNet-based model generates 32-dimensional embedding vector, and the ResNet-32-based models generates 64-dimensional embedding vector. Local models are trained with a SGD optimizer with the learning rate of 0.1 and the mini-batch size of 16. We compared our method FedMetric with DeepSVDD, FedAwS and FedUV. Note that DeepSVDD is not FL method and was proposed for one-class classification.

4.2. Results

Table 2 shows average AUROCs in % per method for one-class experiments on MNIST and CIFAR-10. We compared FedMetric with DeepSVDD, FedAwS, and FedUV. Also, in FedMetric, we compared three loss functions, 1) pos: only positive loss, 2) pos+neg(proxy): positive and negative loss with the proxy centers, and 3) pos+neg(true): positive and negative loss with the true centers. The second loss is our main proposal, and the the performance using the third loss function is our upper-bound performance. In DeepSVDD, each client has different model while federated learning methods use a single global model for evaluation. The performances are taken from the paper (Yu et al., 2020). FedMetric with pos showed better performance than DeepSVDD due to the use of shared information from other clients. By using positive loss and negative loss with the proxy centers, we obtained better performance. As shown in the table, FedMetric using positive loss with negative loss with the true centers is showed better performance than our proposed method. However, note that there is privacy issue when sharing the true centers with other clients.

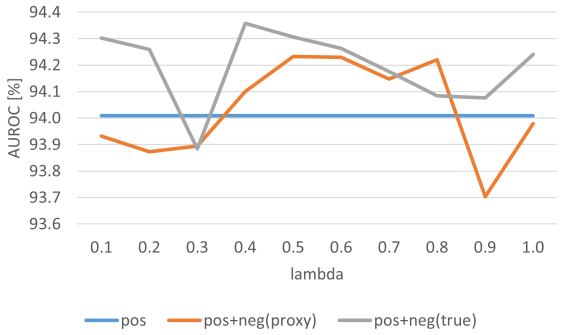

Figure 2. Ablation study of the negative loss weight on CIFAR-10.

Figure 2 shows AUROCs on the CIFAR-10 dataset by changing the negative loss weight λ. $\lambda = 0$ means the case using only positive loss. We tested FedMetric using lambda values from 0.1 to 1.0 with 0.1 interval. We compared the FedMetric of proxy center and true center with the FedMetric of only positive loss which already shows good performance. FedMetric with proxy center shows better performance than FedMetric with only positive loss when we choose lambda appropriately. As expected, FedMetric with true center generally shows better performance than FedMetric with proxy center.

Figure 3 shows AUROCs on the CIFAR-10 dataset by changing the distance D^k of the proxy center. Here, the distance was defined as $D^k = s \times R^k$ where s is a hyperparame-

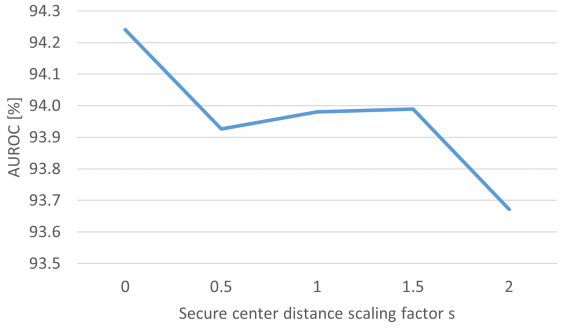

Figure 3. Ablation study of the proxy center distance scaling factor on CIFAR-10.

Table 1. Datasets used for experiments.

Name	# of Classes (Clients)	# of samples per class	input
MNIST	10	6000	28x28 gray-scale image
CIFAR-10	10	5000	32x32 colour image

Table 2. Average AUROCs in % per method for one-class classification on MNIST and CIFAR-10.

Dataset	DeepSVDD	FedMetric			FedAwS	FedUV
		pos	pos+neg(proxy)	pos+neg(true)		
MNIST	94.8	99.4	99.6	99.6	99.6	99.7
CIFAR-10	64.8	94.0	94.2	94.4	94.1	87.2

ter. When $s = 0$, the shared proxy centers are equal to the true centers. If the distance increases, the performance decreases but it helps to reduce the privacy concern. When $s < 1.0$, the shared center can be used to attack the client since the shared vector is located inside the hypersphere of the client. So we need to set $s > 1.0$. We can choose proper s by considering the trade-off between the performance and privacy.

As shown in Table 2, the proposed FedMetric shows a big performance improvement over DeepSVDD, especially when CIFAR-10 dataset was used. And, using both positive and negative loss terms is better than using only positive term, but the performance gap is not remarkable. We think the main reason is due to the small-sized dataset of easy classes in which the proposed FedMetric already obtained quite good performance with only positive loss term. With a bigger dataset and more confusable categories, adding the negative loss term would be more effective, and we will further explore and investigate on this in the future work.

5. Conclusion

We proposed FedMetric, a framework for learning embedding networks with a metric loss in the federated setting, where each client has access to the data of only a single target class. In FedMetric, each client learns a local model by minimizing positive and negative losses with local data and the shared proxy centers. We showed the effectiveness of the proposed method empirically. For the future work, we plan to apply our methods to large-scale datasets.

References

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., McMahan, H. B., et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

Cao, K. and Jain, A. K. Automated latent fingerprint recognition. *IEEE transactions on pattern analysis and machine intelligence*, 41(4):788–800, 2018.

Chopra, S., Hadsell, R., and LeCun, Y. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pp. 539–546. IEEE, 2005.

Chung, J. S., Nagrani, A., and Zisserman, A. Voxceleb2: Deep speaker recognition. *arXiv preprint arXiv:1806.05622*, 2018.

Deng, J., Guo, J., Xue, N., and Zafeiriou, S. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4690–4699, 2019.

Ge, L. and Moh, T.-S. Improving text classification with word embedding. In *IEEE International Conference on Big Data*, pp. 1796–1805, 2017.

Guan, S., Tai, Y., Ni, B., Zhu, F., Huang, F., and Yang, X. Collaborative learning for faster stylegan embedding. *arXiv preprint arXiv:2007.01758*, 2020.

Hosseini, H., Park, H., Yun, S., Louizos, C., Soriaga, J., and Welling, M. Federated learning of user verification models without sharing embeddings. *arXiv preprint arXiv:2104.08776*, 2021.

Hsieh, K., Phanishayee, A., Mutlu, O., and Gibbons, P. B. The non-iid data quagmire of decentralized machine learning. *arXiv preprint arXiv:1910.00189*, 2019.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. Scaffold: Stochastic controlled

averaging for federated learning. In *ICML*, pp. 5132–5143, 2020.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., and Song, L. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 212–220, 2017.

McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. Learning differentially private recurrent language models. In *ICLR*, 2017.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

Nguyen, K., Fookes, C., Ross, A., and Sridharan, S. Iris recognition with off-the-shelf cnn features: A deep learning perspective. *IEEE Access*, 2017.

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., and Kloft, M. Deep one-class classification. In *International conference on machine learning*, pp. 4393–4402. PMLR, 2018.

Sattler, F., Wiedemann, S., Müller, K.-R., and Samek, W. Robust and communication-efficient federated learning from non-iid data. *IEEE Trans. neural networks and learning systems*, 31(9):3400–3413, 2019.

Schroff, F., Kalenichenko, D., and Philbin, J. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Snyder, D., Garcia-Romero, D., Povey, D., and Khudanpur, S. Deep neural network embeddings for text-independent speaker verification. 2017.

Wang, F., Cheng, J., Liu, W., and Liu, H. Additive margin softmax for face verification. 2018a.

Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., Li, Z., and Liu, W. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5265–5274, 2018b.

Wu, Y. and He, K. Group normalization. In *ECCV*, 2018.

Yu, F., Rawat, A. S., Menon, A., and Kumar, S. Federated learning with only positive labels. In *International Conference on Machine Learning*, pp. 10946–10956. PMLR, 2020.

Yun, S., Cho, J., Eum, J., Chang, W., and Hwang, K. An end-to-end text-independent speaker verification framework with a keyword adversarial network. 2019.

Zhang, S., Huang, Z., Zhou, H., and Zhou, Z. Sce: Scalable network embedding from sparsest cut. In *SIGKDD*, pp. 257–265, 2020.

Zhao, W., Guan, Z., Chen, L., He, X., Cai, D., Wang, B., and Wang, Q. Weakly-supervised deep embedding for product review sentiment analysis. *IEEE Trans. Knowledge and Data Engineering*, 30(1):185–197, 2017.

Zhong, Y. and Deng, W. Adversarial learning with margin-based triplet embedding regularization. In *ICCV*, pp. 6549–6558, 2019.

Zhu, J., Shan, Y., Mao, J., Yu, D., Rahmanian, H., and Zhang, Y. Deep embedding forest: Forest-based serving with deep embedding features. In *SIGKDD*, pp. 1703–1711, 2017.