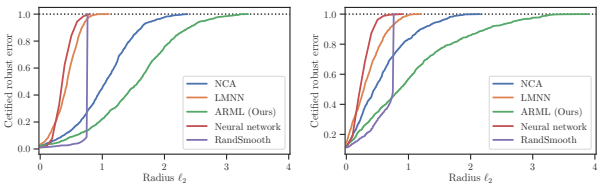
We appreciate all the reviewers for the constructive comments, and our responses are as below:

R1.1: concerns on compared methods and datasets. We fully understand the concern about our baselines since we are the first to improve certified robustness of metric learning. Therefore, as Reviewer 4 suggested, we add experiments comparing with neural networks certification methods, including ordinary neural networks certified by CROWN [48] and randomized-smoothing neural networks [11]. The results are shown in Figure i. Our method outperforms randomized-smoothing networks for a wide range of ℓ_2 perturbations, and we will add these baselines to the paper. As mentioned in the review, these two datasets are often used for evaluating robustness of neural networks, so surpassing a very recently proposed defensive network (RandSmooth) suggests the potential high impact of this work. Regarding the datasets, such UCI datasets are actually commonly used for performance evaluation in previous metric learning papers in various settings (Perrot and Habrard, NeurIPS'15; Zadeh *et al*., ICML'16; Chen *et al*., IJCAI'18).

R2.1: computational cost. In general, computational cost is not an issue for ARML. The average runtime (of 5 trials) of LMNN, ITML, NCA and ARML, are 66.4s, 95.2s, 480.9s and 146.1s respectively, on USPS for 100 iterations. To make the comparison fair, all of the methods are run on CPU (Xeon(R) E5-2620 v4 @ 2.10GHz). In fact, ARML is highly parallelizable and our implementation also supports GPU: when running on GPU (one Nvidia TITAN Xp), the average runtime of ARML is only 10.6s. We will add detailed discussions.



(a) MNIST. (b) Fashion-MNIST.

Figure i: Certified robust errors of Mahalanobis 1-NN compared with ordinary neural networks and randomized-smoothing neural networks (RandSmooth).

R2.2: classification error. We did report the classification error. The classification error is equivalent to the empirical robust error at radius 0 in both Table 1 and Table 2. (Note that the certified robust error in Table 1 is equivalent to the empirical robust error.) One of the major advantages of ARML over existing metric learning methods is that it could improve the classification error and robust error simultaneously. We will highlight them in these two tables.

R2.3 & R3.1: missing papers and existing metric learning methods against adversarial examples. Thanks for mentioning these related work. Both Chen *et al*. (2018) and Duan *et al*. (2018) introduced the adversarial framework as a mining strategy aiming to improve classification accuracy of metric learning, and Zheng *et al*. (2019) made an improvement upon them; Li *et al*. (2019) proposed an attack method for K-NN via a differentiable substitute; Mao *et al*. (2019) used a metric learning loss as a regularization to improve *empirical* robustness of deep learning models; A concurrent paper (Yang *et al*., 2020), made public after the NeurIPS deadline, proposed a similar training method to ours, but still lacks formal robustness verification like our Theorem 1 and Algorithm 2, and therefore did not *certify* robustness formally. In contrast, as we underscore in our paper, we propose the first *certified* defense based on metric learning. We will add a paragraph in the related work section to talk further about adversarial robustness of metric learning with additional references.

R3.2: loss function in Eq. (12). The loss functions in Eq. (11) and Eq. (12) are the same to the one in Eq. (5): a monotonically non-increasing function. In our implementation, we simply use the "negative" loss, i.e., $\ell(\epsilon) = -\epsilon$. We will make it clear in the paper.

R3.3: efficiency issue concerning sampling instances. Sampling from neighborhood instead of computing $k\text{th}\max$ and $k\text{th}\min$ is indeed a crucial technique of ARML to help improve training efficiency. We will clarify it.

R4.1: compare with robust deep learning. We have compared with robust neural networks (see R1.1).

R4.2: Eq. (6) and Eq. (13). It is noteworthy that Eq. (6) involves two layers of optimization, and it is correct that $\mathbb{I}$ and $\mathbb{J}$ are fixed for the inner optimization. The constraints assure that *at most* $k-1 = (K+1)/2 - 1$ "positive" training instances, i.e., instances in $\mathbb{I}$, are in the K-size neighborhood of $\boldsymbol{x}_{\text{test}} + \boldsymbol{\delta}_{(\mathbb{I},\mathbb{J})}$, by means of *not* constraining $(\boldsymbol{x}_{\text{test}} + \boldsymbol{\delta}_{(\mathbb{I},\mathbb{J})})$'s distances from these "positive" training instances; it is a *necessary* condition for a successful attack. Interestingly and not surprisingly, for Mahalanobis 1-NN — the case where we have $K = 1$ and $k = (K+1)/2 = 1$, and hence $\mathbb{I}$ is an empty set, and $\mathbb{J}$ has only one element — Eq. (6) is exactly reduced to Eq. (13).

R4.3: selection of $\boldsymbol{x}^+$ in Eq. (7) (or $\boldsymbol{x}_i$ in Eq. (10)). It is an insightful question. In theory, enumerating every $\{i : y_i = y_{\text{test}}\}$ in Eq. (10) will derive the *tightest* lower bound of the minimal adversarial perturbation. Nevertheless, in practice, only selecting a subset of $\{i : y_i = y_{\text{test}}\}$ of which $\boldsymbol{x}_i$ is "close" to $\boldsymbol{x}_{\text{test}}$ usually suffices to derive a satisfying lower bound. In the robustness evaluation phase, to derive certification bounds as tight as possible, we did not employ this strategy. We will add more discussions for Theorem 1.

R4.4: how the radius is determined in experiments. The radius is only used to show the experimental results, and is *not* a hyperparameter. In fact, we could also plot the certified robust error curve across "all" radii as in Figure i.