

SVMAX: A FEATURE EMBEDDING REGULARIZER

APPENDIX

Anonymous authors

Paper under double-blind review

The following appendix sections A and B extend their corresponding sections in the main paper. For instance, the extended-approach appendix A extends the approach section in the main paper.

A APPENDIX: EXTENDED APPROACH

This section presents the lower and upper bounds of N-pair and angular losses. Then, analyze a *theoretical* corner case for SVMax.

A.1 N-PAIR AND ANGULAR LOSSES

Lower and Upper Bounds of Ranking Losses: The N-pair and angular losses are bounded when their feature embeddings are L2-normalized. These bounds depend on the number of negative samples inside the training mini-batch B . Each mini-batch contains a single positive pair of samples per class, $\{a, p\}$, with all remaining samples, n , representing negative samples of that class. a, p, n denote the anchor, positive and negative samples, respectively. This gives $|n| = b - 2$ as the number of negative samples w.r.t. a mini-batch of size b . Both losses also use an inner product to quantify similarity between feature embeddings, *i.e.*, $\in [-1, 1]$, allowing us to find our bounds. Equation A1 shows the N-pair loss (NPL) formulation:

$$\text{NPL} = -\log \frac{\exp([a][p])}{\exp([a][p]) + \sum_{n \in B} \exp([a][n])}, \quad (\text{A1})$$

where each n is a negative sample, a is the anchor, and p is the positive sample inside the mini-batch B . $[\bullet]$ is the embedding function (encoder). Thus, the lower and upper bounds for the L2-normalized N-pair loss are the following:

$$[L, U]_{\text{NPL}} = [\log(e^2 + |n|) - 2, \log(e^2|n| + 1)]. \quad (\text{A2})$$

Equations A3 and A4 show the angular loss (AL) formulation:

$$\text{AL} = \log \left[1 + \sum_{n \in B} \exp(f_{a,p,n}) \right] \quad (\text{A3})$$

$$\text{s.t. } f_{a,p,n} = 4 \tan^2 \alpha ([a] + [p])^T [n] - 2(1 + \tan^2 \alpha) [a]^T [p], \quad (\text{A4})$$

where, again, each n is a negative sample, a is the anchor, p is the positive sample inside the mini-batch B , and $[\bullet]$ is the embedding function (encoder). The parameter α is a hyperparameter chosen before training and is thus a fixed value. As such, the lower and upper bounds for L2-normalized $f_{a,p,n}$ are:

$$[L, U]_{f_{a,p,n}} = [-10 \tan^2 \alpha - 2, 6 \tan^2 \alpha - 2]. \quad (\text{A5})$$

We use $\alpha = 45^\circ$ in all our experiments. This gives $\tan \alpha = 1$, $[L, U]_{f_{a,p,n}} \in [-12, 4]$ and our angular loss bounds as:

$$[L, U]_{\text{AL}} = [\log(e^{-12}|n| + 1), \log(e^4|n| + 1)]. \quad (\text{A6})$$

A.2 SVMAX CORNER CASE

Theoretically, the mean singular value s_μ can reach its upper bound for a mini-batch, even if the feature embedding is not perfectly uniform. During training, each mini-batch contains p different

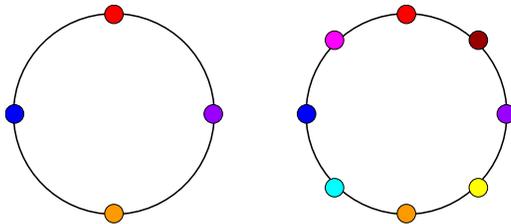


Figure A1: The feature embeddings of two independent mini-batches with $p = 4$ (Left) and $p = 8$ (Right) classes on the 2D unit circle. Different colors denote different classes. The mean singular value s_μ is maximum if (1) samples from the same class have zero standard deviation and (2) the p sampled classes are distributed perfectly in the embedding space.

CUB-200-2011				Stanford CARS196				Stanford Online Products			
Query	Top 1	Top 2	Top 3	Query	Top 1	Top 2	Top 3	Query	Top 1	Top 2	Top 3

Figure A2: Qualitative retrieval evaluation using three datasets: CUB-200-2011, Stanford CARS196, and Stanford Online Products. For a given query image, the three nearest neighbors are depicted. Green and red outlines denote class match and mismatch, respectively

classes and l different samples per class, *i.e.*, the batch size $b = p \times l$. The sampled classes p in a mini-batch is smaller than the number of total classes C in the dataset, *i.e.*, $p < C$. If (1) the sampled p classes are perfectly distributed in the feature embedding and (2) the l samples for each class have zero standard deviation, the mean singular value equals the upper bound ($s_\mu = U$) even if the feature embedding for the whole dataset is not uniform. Figure A1 illustrates this scenario.

In practice, this will never happen during training because (1) the feature embedding dimension is large enough (*e.g.*, $d = 128$), (2) samples in the training mini-batches are randomly sampled, and (3) samples are independent across mini-batches. This corner case is omitted from the main paper because it undermines the paper’s flow and clarity.

B APPENDIX: EXTENDED EXPERIMENTS

This section provides further quantitative evaluations for SVMMax when $b \geq d$. Then, we evaluate the performance of SVMMax on small mini-batches, *i.e.*, $b < d$. Finally, we evaluate the computational complexity of SVMMax empirically. The supplementary *video* vividly shows how SVMMax speeds convergence on the MNIST dataset.

Evaluation Metrics: For quantitative evaluation, we leverage the **Recall@K** metric and **Normalized Mutual Info** (NMI) on the test split. The NMI score evaluates the quality of cluster alignments. $NMI = \frac{I(\Omega, C)}{\sqrt{H(\Omega)H(C)}}$, where $\Omega = \{\omega_1, \dots, \omega_n\}$, is the ground-truth clustering, while $C = \{c_1, \dots, c_n\}$ is a clustering assignment for the learned embedding. $I(\cdot, \cdot)$ and $H(\cdot)$ denote mutual information and entropy, respectively. We use K-means to compute C .

Quantitative Evaluation: In the main paper, SVMMax is evaluated quantitatively using GoogLeNet. Figures A3 and A4 present quantitative evaluation using ResNet50 on CUB-200-2011 and Stanford CARS196, respectively. Figure A2 presents a qualitative retrieval evaluation using the three datasets.

Our evaluation hyperparameters (*e.g.*, learning rate and batch size) do not favor a particular ranking loss. The ideal hyperparameters depend on the ranking loss and other factors such as the dataset

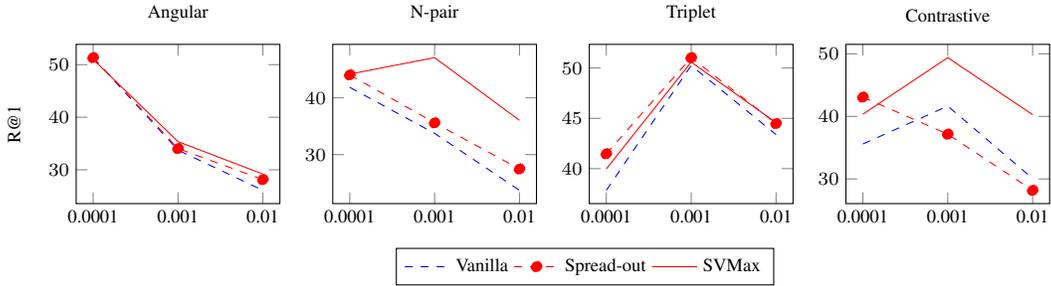


Figure A3: Quantitative evaluation on CUB-200-2011 using ResNet50. The X axis denotes the learning rate lr and the Y-axis denotes recall@1 performance.

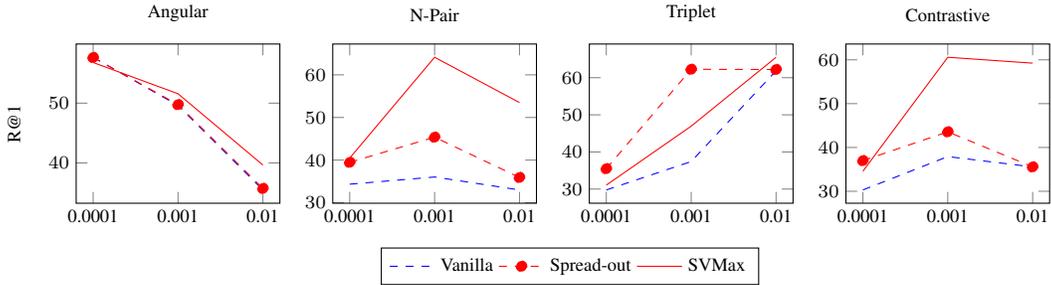


Figure A4: Quantitative evaluation on Stanford CARS196 using ResNet50.

size, batch size, and the network architecture. The hyperparameters in the main paper are inconsistent with the N-pair loss because this loss assumes an un-normalized embedding. In table A1, we evaluate SVMMax with the un-normalized embedding of three datasets. We leverage both the unbound SVMMax (UN SVMMax) in equation A7 and the bounded SVMMax (SVMMax) in equation A8. Surprisingly, the bounded SVMMax, which assumes an L2-normalized embedding, achieves competitive performance on the un-normalized embedding. It is important to note that while N-pair loss assumes an un-normalized embedding, N-pair loss *regularizes* the L2-norm of the embedding vectors to be small. Table A1 shows that the spread-out regularizer degenerates severely, while SVMMax remains resilient. The spread-out regularizer requires an L2-normalized embedding while SVMMax does not. In these experiments, the unbound SVMMax uses $\lambda = 0.01$ while the bounded SVMMax uses $\lambda = 1$.

$$L_{NN} = L_r - \lambda \frac{1}{d} \sum_{i=1}^d s_i = L_r - \lambda s_\mu, \tag{A7}$$

$$L_{NN} = L_r + \lambda \exp\left(\frac{U - s_\mu}{U - L}\right). \tag{A8}$$

Figures A5 and A6 present a quantitative evaluation with various embedding dimensions. We use batch sizes $b = \{288, 72\}$ and embedding dimensions $d = \{256, 64\}$. In this experiment, we employ a MobileNetV2 Sandler et al. (2018) to fit our neural network on a 24GB GPU. The N-pair and angular loss evaluations are dropped in Figure A6 because these losses assume a single pair of anchor-positive per class. The CARS196, with 98 training classes, is too small to provide $144 = \frac{288}{2}$ anchor-positive pairs.

In the main paper, we discuss two factors that contribute to model collapse in retrieval networks: learning rate and dataset intra-class variations. However, additional factors can also contribute. For instance, the likelihood of model collapse decreases as the mini-batch size b increases. In the early training stages, a large learning rate will induce a noisy gradient. A large training mini-batch mitigates this noisy gradient and learns a better feature embedding.

Table A1: Quantitative evaluation using N-pair loss without L2-normalization on three datasets and GoogLeNet with batch size $b = 144$, embedding dimension $d = 128$ and multiple learning rates $lr = \{0.01, 0.001, 0.0001\}$. $\Delta_{R@1}$ column indicates the R@1 improvement margin relative to the vanilla ranking loss.

Method	$lr = 0.01$				$lr = 0.001$				$lr = 0.0001$			
	NMI	R@1	R@8	$\Delta_{R@1}$	NMI	R@1	R@8	$\Delta_{R@1}$	NMI	R@1	R@8	$\Delta_{R@1}$
N-pair on CUB-200-2011												
Vanilla	0.560	46.08	79.09	-	0.553	44.24	78.12	-	0.555	45.32	78.71	-
Spread-out	degenerate (Trn loss = nan)			-	0.560	45.80	78.66	1.55	0.537	42.67	77.16	-2.65
UN SVMMax (Ours)	0.563	45.80	79.73	-0.29	0.553	44.90	78.58	0.66	0.555	44.97	78.80	-0.35
SVMMax (Ours)	0.561	46.27	79.86	0.19	0.563	46.40	79.69	2.16	0.563	45.70	79.27	0.37
N-pair on Stanford CARS196												
Vanilla	0.606	65.49	90.32	-	0.5983	64.92	90.00	-	0.480	44.13	79.77	-
Spread-out	degenerate (Trn loss = nan)			-	0.109	2.58	14.07	-62.34	0.393	31.53	67.85	-12.59
UN SVMMax (Ours)	0.606	65.45	90.87	-0.04	0.592	64.92	90.08	0	0.481	44.25	79.93	0.12
SVMMax (Ours)	0.611	69.72	92.15	4.23	0.601	67.31	91.33	2.39	0.492	48.73	83.18	4.60
N-pair on Stanford Online Products												
Vanilla	0.897	74.54	87.92	-	0.884	69.34	84.46	-	0.863	59.68	76.42	-
Spread-out	degenerate (Trn loss = nan)			-	0.877	64.23	80.34	-5.11	0.858	56.50	73.15	-3.18
UN SVMMax (Ours)	0.897	74.82	88.30	0.28	0.883	69.30	84.49	-0.05	0.864	59.69	76.43	0.01
SVMMax (Ours)	0.895	74.44	87.91	-0.10	0.882	69.52	84.62	0.17	0.864	59.93	76.73	0.26

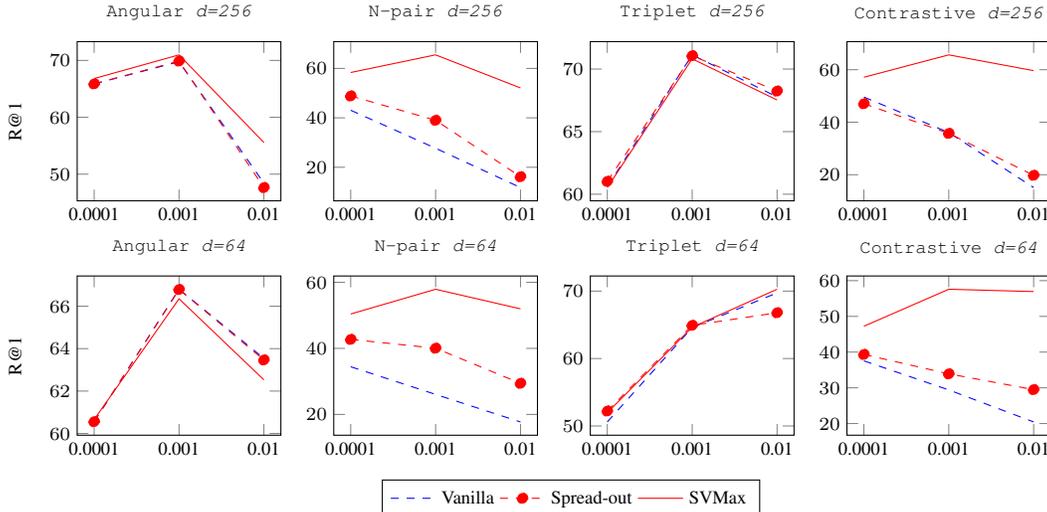


Figure A5: Quantitative evaluation on Stanford Online Products using various embedding dimensions $d = \{256, 64\}$ to demonstrate the stability of our hyperparameter.

B.1 SVMMax WITH SMALL BATCHES

In the main paper, we assumed $b \geq d$ to deliver a rigorous mathematical foundation for SVMMax. In this section, we present an empirical evidence to support SVMMax when $b < d$.

When $b < d$, there will be b singular values, instead of d . The lower and upper bounds of SVMMax, per mini-batch, become $[L, U] = [\frac{\sqrt{b}}{b}, \sqrt{\frac{b \times d}{\max(b, d)} \frac{\sqrt{b}}{b}}]$. It is possible that SVMMax will utilize only b dimensions of the feature embedding space. We argue against this possibility using a toy example. Consider the following two mini-batches $(m_1, m_2) \in R^{3 \times d}$

$$m_1 = \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \end{bmatrix}, m_2 = \begin{bmatrix} m_{21} \\ m_{22} \\ m_{23} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \end{bmatrix},$$

where the mini-batch size $b = 3$. Each individual mini-batch utilizes only the first three dimensions, i.e., $\text{rank}(m_1) = \text{rank}(m_2) = 3$. While all other dimensions $[4, d]$ contain zeros, the maximum mean singular value is feasible with only the first three dimensions. However, due to the

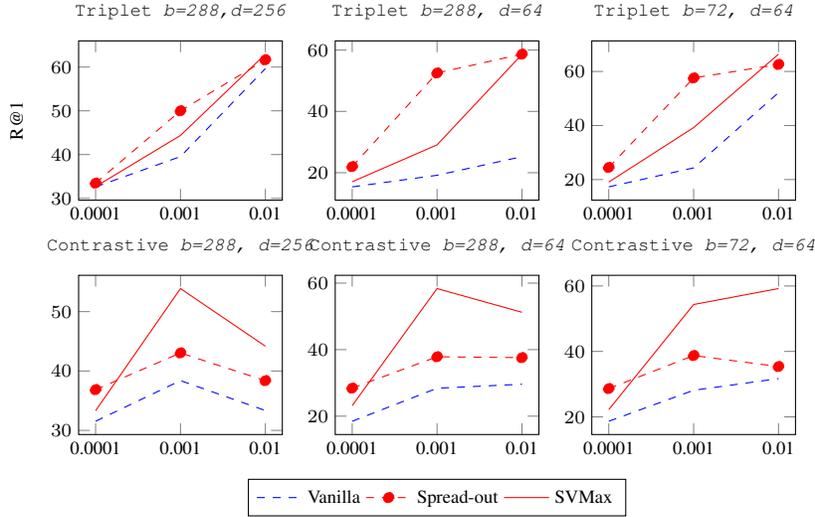


Figure A6: Quantitative evaluation on Stanford CARS196 using MobileNet, various embedding dimensions $d = \{256, 64\}$, and batch sizes $b = \{288, 72\}$ to demonstrate the stability of our hyperparameter. $\lambda = 1$ and 0.1 for contrastive and triplet loss, respectively.

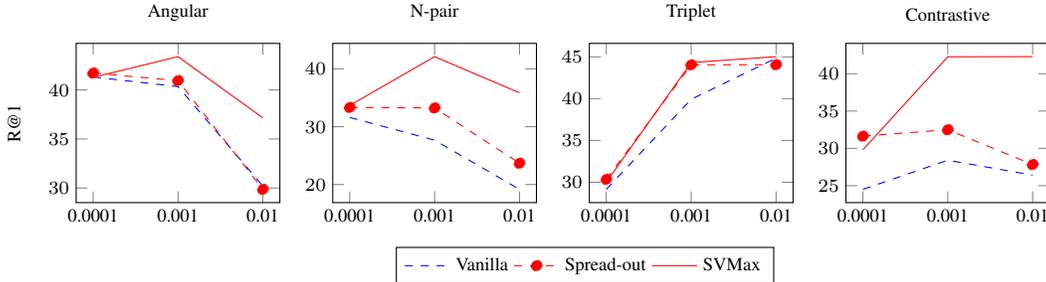


Figure A7: Quantitative evaluation on CUB-200-2011 using GoogLeNet with $b = 72$ and $d = 128$, *i.e.*, $b < d$.

random sampling procedure, a future mini-batch m_3 will contain elements from both m_1 and m_2 . For instance, $m_3 = [m_{11} \ m_{21} \ m_{22}]^T$ will have a $rank(m_3) = 2$. For the mini-batch m_3 , the mean singular value is not maximum. To maximize s_μ , one feasible solution is to keep utilizing only the first three dimensions. However, this solution is like tossing a coin N times and expecting N heads. It is a feasible solution but unlikely.

Figure A7 presents a quantitative evaluation using CUB-200 on GoogLeNet with $b = 72$ and $d = 128$. Similarly, Figure A8 presents a quantitative evaluation using Stanford Online Products. SVMMax consistently outperforms the vanilla and spread-out baselines even when $b < d$.

Finally, Figure A9 depicts the mean singular value on the test split of CUB-200. We train our network using (1) contrastive loss with and without SVMMax, and (2) different mini-batch sizes $b = \{72, 144\}$. We fix the embedding dimension $d = 128$ to study the batch size’s impact, *i.e.*, $b < d$ versus $b \geq d$. The test split of CUB-200 has 5924 test images. Thus, the upper bound of the mean singular value $U = \sqrt{\frac{b \times d}{\max(b, d)}} \frac{\sqrt{b}}{d} = 6.80$, where $b = 5924$ and $d = 128$ for the *whole* test split. After training our network, the actual mean singular value $s_\mu = 5.64$ with batch size $b = 72$, and $s_\mu = 5.81$ with $b = 144$. These mean singular values significantly outperform their vanilla contrastive loss counterparts ($s_\mu = 1.9$). Compared to $b = 144$, s_μ is smaller when using the mini-batch size $b = 72$. At a mini-batch level, SVMMax spreads the feature embedding across $d = 128$ dimensions when $b = 144$, while SVMMax spreads the feature embedding across $d = 72$ dimensions when $b = 72$. Yet, the comparable s_μ (5.64 versus 5.81) emphasizes that SVMMax supports $b < d$.

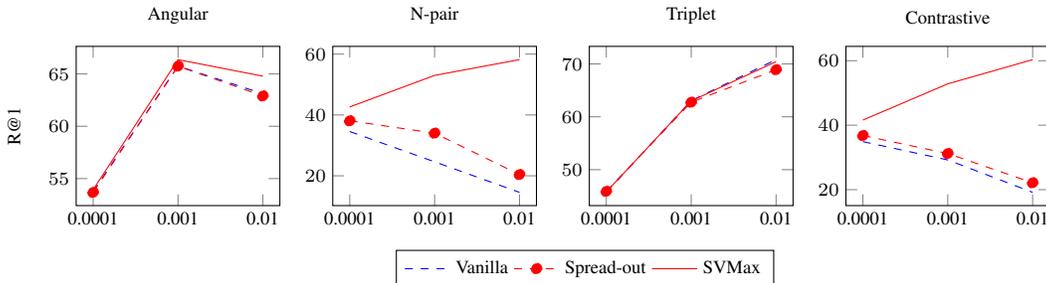


Figure A8: Quantitative evaluation on Stanford Online Products using GoogLeNet with $b = 72$ and $d = 128$, i.e., $b < d$.

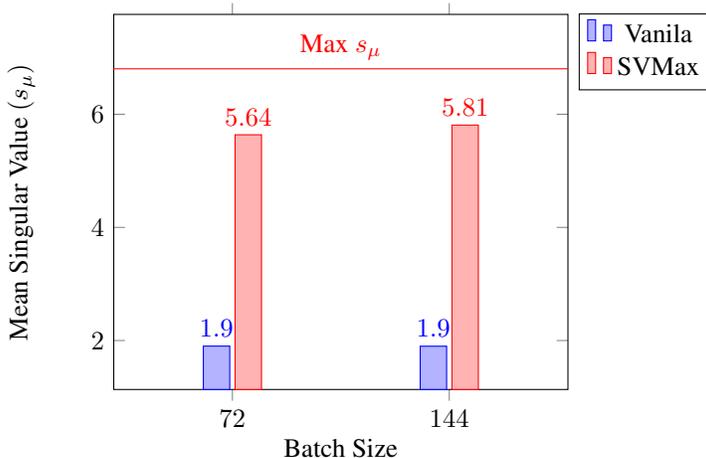


Figure A9: The mean singular values s_μ for networks trained with an embedding dimension $d = 128$. The X and Y-axes denote the mini-batch size b and the s_μ of the feature embedding of CUB-200’s test split. The feature embedding is learned using a contrastive loss with and without SVMMax. The horizontal red line denotes the upper bound on s_μ . Compared to $b = 144$, the mean singular value s_μ , with SVMMax and $b = 72$, decreases marginally. Thus, the SVMMax still promotes a uniform feature embedding even when $b < d$.

With respect to GANs, if the batch-size limitation is set aside, the following points are worth noting: (I) Image-synthesis GANs have bounded outputs $[0, 255]$; White images will not fool the discriminator. Thus, s_μ remains bounded but with different bounds from those presented in the approach section in the main manuscript. (II) Alain & Bengio (2016) (§3.4) address practical concerns when working with high dimensional features. (III) GANs have synthesized not only high quality images, but also feature embeddings Zhu et al. (2018).

B.2 SVMAX COMPUTATIONAL COMPLEXITY

We compute the singular values s_μ using TensorFlow (TF) `tf.linalg.svd`. This function runs on the GPU. We did not notice any computational overhead or numerical instability during training. Figure A10 (Left) provides a timing analysis of the TF function using square matrices. For a typical mini-batch size ($b < 256$), the function takes around 0.01 seconds. This speed depends on the GPU specification and recent GPUs would perform faster. Figure A10 (Right) provides a timing analysis for the mini-batch training time using MobileNet. The overhead added by SVMMax is minimal compared to the overhead of performing gradient descent on a deep network. We conclude that for a reasonable batch size b and embedding dimension d , SVMMax adds minimal computational complexity to the training process.

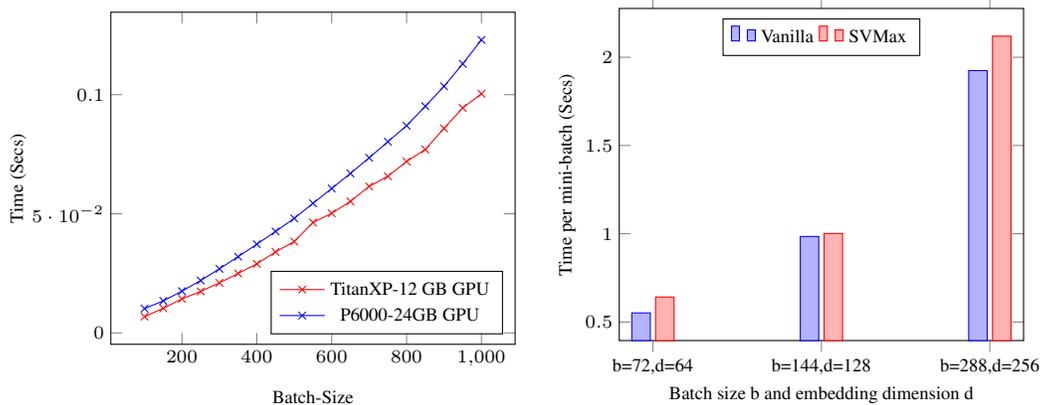


Figure A10: (Left) Timing analysis for the Tensorflow (TF) `tf.linalg.svd` function. The x-axis denotes the batch size b , and the y-axis denotes the running time in seconds. We time this TF function using two different GPUs on two different machines. (Right) Timing analysis for a mini-batch training time using MobileNet. The x-axis denotes both the batch size b and the embedding dimension d . The y-axis denotes the batch training time in seconds.

REFERENCES

- Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.
- Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.
- Yizhe Zhu, Mohamed Elhoseiny, Bingchen Liu, Xi Peng, and Ahmed Elgammal. A generative adversarial approach for zero-shot learning from noisy texts. In *CVPR*, 2018.