Boosting Standard Classification Architectures Through a Ranking Regularizer

Supplementary material

Ahmed Taha[1] Yi-Ting Chen[2] Teruhisa Misu[2] Abhinav Shrivastava[1] Larry Davis[1]

[1]University of Maryland, College Park [2]Honda Research Institute, USA

1. Supplementary Material

The next subsections provide more details about our architecture and training procedure's technicalities. Further quantitative evaluations on fine-grained visual recognition (FGVR) are presented. Finally, we demonstrate the training procedure for the Honda Research Institute Driving Dataset.

1.1. Fine-Grained Visual Recognition

Figure 2 in the main paper presents our two-head architecture. The pre-logit layer x supports the softmax loss. Similarly, triplet loss utilizes h, where $x = pool(h)$. The network outputs, both logits and embedding, are formulated as follows.

$$\text{logits} = W_{\text{logits}} * \text{flat}(x) \tag{1}$$

$$\text{embedding} = W_{\text{emb}} * \text{flat}(h). \tag{2}$$

Orderless pooling, like averaging, reduces h dimensionality but loses spatial information. For example, in DenseNet161, $h \in R^{7\times7\times2208}$ while $x \in R^{1\times1\times2208}$. Thus, W_{emb} employs h, instead of x, to improve feature embedding. Figure S1 illustrates how h provides a finer control level while learning W_{emb}.

Figure S2 shows a t-SNE visualization for Flowers-102 embedding using 50 random classes, 20 samples per class. In the main paper, the inferior performance of triplet loss with hard-mining is associated with convergence to bad local minima, *i.e.*, a collapsed model (*i.e.* $f(x) = 0$) [7]. To examine such assumption, we train a DenseNet for 400K iterations on Stanford Dogs. This large number of iterations increases the chances of a model collapse. Figure S3 presents the performance on the test split after every 50K iterations. Triplet loss with hard-mining is evaluated with both soft and hard margin. Soft margin applies the softplus function $\ln(1+\exp(\bullet))$ while hard margin uses a fixed margin $m = 0.2$. The triplet loss with hard-mining deteriorates with soft margin when trained for a large number of iterations. Hard-mining with hard margin is more robust. We found similar behavior on other datasets like Stanford Cars and Aircrafts datasets.

Table 5 in the main paper presents a quantitative analysis for the feature embedding learned by the second head in our proposed architecture. Similarly, Table S1 presents feature embedding quantitative analysis using the architecture penultimate layer, *i.e.*, layer x (Figure 2 in the main paper). This layer is present in both our proposed two-head and single-head (vanilla softmax) architecture. Similar to Table 5, the triplet loss embedding is superior to the softmax embedding. Triplet loss with hard-mining achieves the best results on ResNet-50 but degrades on Inception-V4 trained for 80K iterations. Center loss achieves good results with DenseNet161 on NABirds but generally fluctuates and suf-

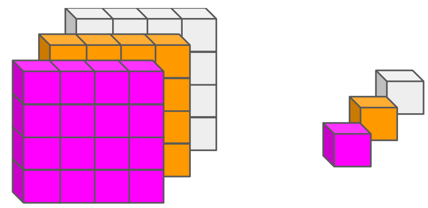

Figure S1: Orderless pooling reduces dimensionality but loses features spatial information.

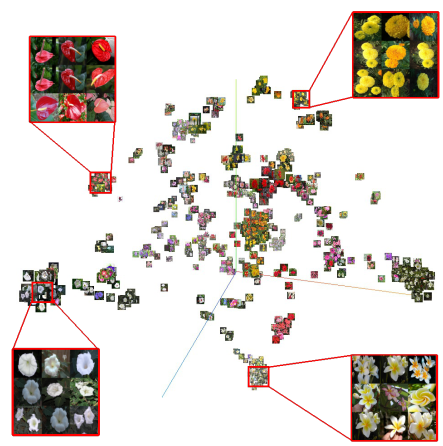

Figure S2: t-SNE visualization for Flowers-102 embedding using 50 random classes, 20 samples per class. Best viewed in color.

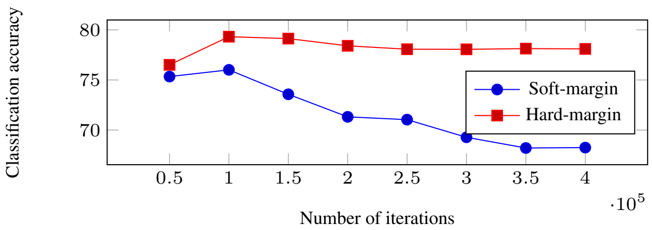

Figure S3: Model collapse study by training DenseNet161 for 400K iterations. Triplet loss with hard-mining evaluated with soft and hard margins.

		NMI	R@1	R@4	R@8	R@16
Cars - ResNet	Vanilla	0.791	77.88	91.17	94.65	96.9
	CNTR	0.756	77.98	91.12	94.58	96.78
	SEMI	0.823	81.41	92.79	95.91	97.74
	HARD	**0.853**	**85.31**	**94.30**	**96.82**	**98.07**
Flowers - ResNet	Vanilla	0.800	88.76	95.51	97.27	**98.49**
	CNTR	0.807	88.79	95.58	97.32	**98.49**
	SEMI	**0.818**	89.48	**95.82**	**97.37**	98.37
	HARD	0.742	**90.78**	95.56	96.93	97.98
Dogs - ResNet	Vanilla	0.587	51.62	74.22	83.02	89.76
	CNTR	0.526	48.74	71.90	80.92	87.81
	SEMI	0.621	54.18	76.39	84.50	91.10
	HARD	**0.684**	**60.37**	**80.34**	**87.33**	**92.26**
Aircrafts - ResNet	Vanilla	0.756	73.42	87.88	92.26	94.90
	CNTR	0.677	70.84	85.84	90.79	93.91
	SEMI	0.792	77.26	89.65	93.07	95.29
	HARD	**0.829**	**84.01**	**91.63**	**94.21**	**95.65**
NABirds - ResNet	Vanilla	0.669	50.70	71.20	79.48	85.80
	CNTR	0.623	47.40	68.18	76.56	83.33
	SEMI	0.657	50.05	70.83	78.84	85.52
	HARD	**0.723**	**55.85**	**75.81**	**83.26**	**88.67**
Cars - Inc-V4	Vanilla	0.660	72.47	86.77	90.55	93.55
	CNTR	0.496	61.55	79.09	85.09	89.69
	SEMI	**0.788**	**81.46**	**92.14**	**94.64**	**96.37**
	HARD	0.566	63.70	82.04	87.54	91.42
Flowers - Inc-V4	Vanilla	0.778	90.54	96.21	97.63	98.70
	CNTR	0.707	85.62	93.74	95.95	97.56
	SEMI	**0.801**	89.58	95.23	96.91	97.84
	HARD	0.731	**92.68**	**96.21**	**97.27**	**98.32**
Dogs - Inc-V4	Vanilla	0.421	41.11	62.97	72.59	81.13
	CNTR	0.453	**57.13**	68.32	72.35	76.90
	SEMI	**0.609**	55.03	**76.50**	**84.44**	**90.23**
	HARD	0.330	33.89	54.28	65.06	74.98
Aircrafts - Inc-V4	Vanilla	0.680	69.79	85.18	89.23	91.93
	CNTR	0.546	61.60	79.75	85.33	89.53
	SEMI	0.751	78.13	89.20	91.78	94.27
	HARD	**0.831**	**86.26**	**91.87**	**93.49**	**94.72**
NABirds - Inc-V4	Vanilla	0.546	41.03	60.11	68.88	76.71
	CNTR	0.438	24.30	40.43	49.38	58.78
	SEMI	**0.638**	**52.42**	**72.38**	**79.57**	**85.60**
	HARD	0.433	23.68	38.95	47.48	57.10
Cars - Dense	Vanilla	0.813	85.08	94.49	96.84	98.22
	CNTR	0.787	87.39	93.17	94.64	95.97
	SEMI	0.875	88.57	96.08	97.66	98.71
	HARD	**0.892**	**89.44**	**96.38**	**97.86**	**98.76**
Flowers - Dense	Vanilla	0.838	95.28	98.23	98.94	99.38
	CNTR	0.812	95.87	98.16	98.75	99.22
	SEMI	0.864	95.40	98.39	99.09	99.46
	HARD	**0.865**	**95.79**	**98.50**	**99.14**	**99.50**
Dogs - Dense	Vanilla	0.544	57.06	78.72	85.98	91.84
	CNTR	0.720	**70.96**	84.00	88.19	91.96
	SEMI	0.728	68.55	87.04	92.18	95.83
	HARD	**0.756**	70.63	**87.80**	**92.95**	**96.22**
Aircrafts - Dense	Vanilla	0.768	79.06	91.66	94.66	96.49
	CNTR	0.792	86.20	91.63	93.16	94.48
	SEMI	0.853	84.49	**94.15**	95.68	**96.97**
	HARD	**0.856**	**85.51**	93.70	**95.83**	96.94
NABirds - Dense	Vanilla	0.606	53.91	73.08	80.70	86.44
	CNTR	**0.818**	**75.28**	**86.88**	**90.85**	**93.69**
	SEMI	0.677	61.82	80.70	87.07	91.62
	HARD	0.674	61.64	80.21	86.77	91.37

Table S1: Detailed feature embedding quantitative analysis across the five datasets using ResNet-50, Inception-V4 and DenseNet-161 architectures' penultimate layer x. Triplet with hard mining achieves superior embedding with ResNet-50 trained for 40K iterations. Semi-hard triplet is competitive and stable with Inception-V4 trained for 80K iterations. Center loss suffers a high instability.

fers with Inception-V4. Triplet loss with semi-hard margin achieves sub-optimal embedding but maintains the highest stability compared to center and hard-mining approaches.

Figure S4 graphically summarizes Table S1. It provides a comparative embedding evaluation between the single-head softmax verses the two-head with semi-hard triplet loss using recall@1 metric. Triplet loss improvements, over the softmax model, are reported as ($\triangle$). The Flowers-102 dataset has the smallest training split with 1020 images only. With this limited data, the head-two architecture achieves marginal improvement if any.

Table S2 compares our proposed two-head architecture, using DenseNet161, with state-of-the-art approaches on the five FGVR datasets. Our two-head architecture with the semi-hard triplet loss regularizer achieves competitive results.

1.2. Autonomous Car Driving

The Honda Research Institute Driving Dataset (HDD) contains 137 sessions S. Each session S_i represents a navigation task performed by a driver. S is divided into 93, 5, and 36 sessions for training, validation and testing splits respectively. Three sessions are removed for missing annotations. HDD has four annotation layers to study the drivers' actions: (1) Goal-oriented, (2) stimulus-driven, (3) cause and (4) attention. The **Goal-oriented** layer, utilized in our experiments , defines drivers' actions to reach their destinations, *e.g.*, *left-turn* and *intersection passing*. Ramanishka *et al*. [6] provides further details for the other three annotation layers.

Triplet loss mini-batches require both positive and negative samples. The FGVR datasets have uniform class distribution. Thus, training batches' construction is straightforward by sampling random classes and their corresponding images as outlined in the main paper. On the other hand, HDD suffers class imbalance. A different batch construction procedure is required.

Algorithm 1 outlines our training procedure. First, N_B mini-batches are constructed, each containing b random actions. The batches' embeddings are computed using N_B feed forward passes. The $2D$ matrix D_ϕ stores the pairwise distance between the $N_B \times b$ actions. All positive

Flowers-102

Method	Acc
Det.+Seg. [1]	80.66
Overfeat [8]	86.80
Softmax	92.56
Two-Head (Semi)	**93.65**

Aircrafts

Method	Acc
LRBP [2]	87.30
Liu *et al.* [4]	88.50
Softmax	89.13
Two-Head (Semi)	**89.64**

NABirds

Method	Acc
Branson *et al.* [9]	35.70
Van *et al.* [3]	75.00
Softmax	78.69
Two-Head (Semi)	**79.57**

Cars

Method	Acc
Liu *et al.* [5]	86.80
Liu *et al.* [4]	92.00
Softmax	91.64
Two-Head (Semi)	**92.36**

Dogs

Method	Acc
Zhang *et al.* [9]	80.43
Krause *et al.* [3]	80.60
Softmax	**81.58**
Two-Head (Semi)	80.89

Table S2: Quantitative evaluation on the five FGVR datasets using DenseNet161. Our two-head architecture with semi-hard triplet loss regularizer compares favorably with state-of-the-art results.

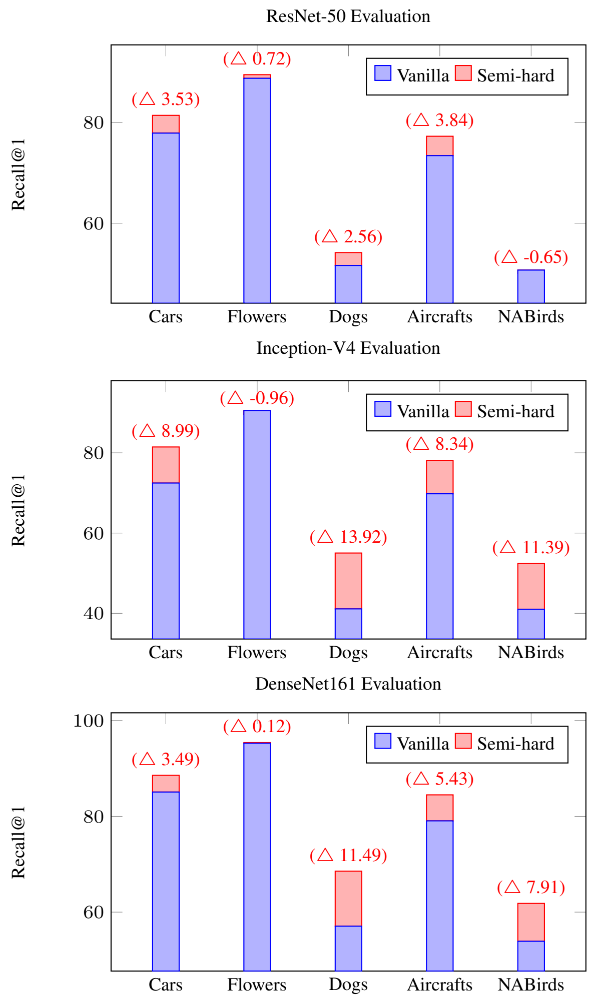

Figure S4: Comparative embedding evaluation between single-head softmax and two-head with semi-hard triplet loss using the penultimate layer in ResNet-50, Inception-V4 and DenseNet161 respectively. Triplet loss semi-hard improvements over the softmax model are reported as (△).

pairs (a, p) and their corresponding semi-hard negatives n are identified. For a fair comparison with vanilla softmax approach, only $(b//3)$ random triplets (a, p, n) are utilized for back-propagation. This process repeats for N training iterations.

Algorithm 1 HDD training procedure. In our experiments, $b = \{33, 36\}$ is the mini-batch size, $N_B = 3$ is the number of mini-batches, and $N = 10K$ is number of training iterations.

for all iteration i in N **do**
  $S_\phi = \Phi$
  for all j in N_B **do**
    Add a random batch, of size b, to S_ϕ
  end for
  Compute action embeddings E_ϕ for S_ϕ
  Compute pairwise distance matrix D_ϕ using E_ϕ
  $T_{tri} = \Phi$
  Construct all positive pairs (a, p)
  for all (a, p) in positive pairs **do**
    Find nearest semi-hard negative n using D_ϕ
    append (a, p, n) to T_{tri}
  end for
  if $len(T_{tri}) > b//3$ **then**
    T_{tri} =shuffle$(T_{tri})[0 : b//3]$
  end if
  // T_{tri} contains b actions
  Feed-forward T_{tri}
  compute softmax + triplet losses and back-propagate.
end for

References

[1] A. Angelova and S. Zhu. Efficient object detection and segmentation for fine-grained recognition. In *CVPR*, 2013. 3

[2] S. Kong and C. Fowlkes. Low-rank bilinear pooling for fine-grained classification. In *CVPR*. IEEE, 2017. 3

[3] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition. In *ECCV*. Springer, 2016. 3

[4] T.-Y. Lin and S. Maji. Improved bilinear pooling with cnns. *arXiv preprint arXiv:1707.06772*, 2017. 3

[5] M. Liu, C. Yu, H. Ling, and J. Lei. Hierarchical joint cnn-based models for fine-grained cars recognition. In *International Conference on Cloud Computing and Security*. Springer, 2016. 3

[6] V. Ramanishka, Y.-T. Chen, T. Misu, and K. Saenko. Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning. In *CVPR*, 2018. 2

[7] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 1

[8] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014. 3

[9] Y. Zhang, X.-S. Wei, J. Wu, J. Cai, J. Lu, V.-A. Nguyen, and M. N. Do. Weakly supervised fine-grained categorization with part-based image representation. *IEEE Transactions on Image Processing*, 2016. 3