Supplementary Material for "Granularity-aware Adaptation for Image Retrieval over Multiple Tasks"

Jon Almazán[1] Byungsoo Ko[2] Geonmo Gu[2]
Diane Larlus[1] Yannis Kalantidis[1]

[1]NAVER LABS Europe [2]NAVER Corp.

Appendix

We report further analysis that supports our choice to learn adaptors instead of fine-tuning the backbone (Section A), zero-shot performance of `Grappa` on datasets that are not in the Multiple Retrieval Tasks benchmark (Section B), and results on learning `Grappa`'s fusion layer using class labels (Section C). We also present PyTorch-style pseudo-code for our architecture (Section D).

A Learning adaptors vs fine-tuning the backbone

We validate the different configurations of adaptors presented in the main paper and compare them in a supervised setting with a typical baseline that involves fine-tuning all layers of a given pretrained model $\mathcal{M}$. In particular, we compare this baseline with fine-tuning just a single, randomly initialized adaptor ($\mathcal{M} + \mathcal{A}_1$), and with adding 8 random adaptors and fine-tuning them together with a fusion layer ($\mathcal{M}^*$).

For both settings, the underlying model $\mathcal{M}$ is *kept fixed*, and we use DINO [1] pretrained on ImageNet in all models. We use the train splits of the 6 datasets in MRT (Aircraft, Cars, CUB, Flowers, Food-101, and Products) and train 6 individual models per method, *using class labels* and a norm-softmax loss.

We report the performance on their respective test splits in the diagonal of Table I. We observe that only adding a single adaptor outperforms the other models in almost all datasets. This is probably due to the fact that these are relatively small datasets and a single adaptor provides enough capacity to specialize, yet too little to overfit. When comparing $\mathcal{M} + \mathcal{A}_1$ to $\mathcal{M}^*$, we already saw a similar behavior in Table 2 in the main paper (rows 2 and 3), suggesting that randomly initialized adaptors perform worse when they are simultaneously trained with a fusion layer, and that one way to address this is pretraining them for different objectives.

B Zero-shot performance

Supervised training separately on each dataset. We consider specialized models that have been trained in a supervised manner on each dataset separately,

Table I. **Supervised finetuning using class and task labels**, applied to the current task and to other retrieval tasks. Based on the ImageNet pre-trained DINO model $\mathcal{M}$, each model is finetuned on a single dataset in a supervised manner (row) and tested on the same and on the other datasets (column). We report MAP@R (mAP) and $\mathcal{R}$P. Note that those numbers constitute an **upper-bound** and they are not directly comparable with those from the main paper.

Train \ Test	Aircraft		Cars		CUB		Flowers		Food-101		Products	
	$\mathcal{R}$P	mAP	$\mathcal{R}$P	mAP	$\mathcal{R}$P	mAP	$\mathcal{R}$P	mAP	$\mathcal{R}$P	mAP	$\mathcal{R}$P	mAP
DINO ($\mathcal{M}$)	17.5	9.2	9.0	3.5	33.6	22.9	62.5	57.0	27.0	16.1	34.1	31.5
Aircraft												
$\mathcal{M}$ (supervised finetuning)	**33.3**	**20.5**	7.7	2.7	17.2	8.6	56.2	49.5	14.3	5.5	33.9	31.3
$\mathcal{M}+\mathcal{A}_1$ (supervised)	**33.2**	**20.9**	10.5	4.4	23.2	13.4	59.4	53.1	18.3	8.6	30.0	27.5
$\mathcal{M}^*$ (random, supervised)	19.9	10.1	8.4	3.0	12.1	5.1	54.9	47.6	13.8	5.2	29.0	26.6
Cars												
$\mathcal{M}$ (supervised finetuning)	15.9	8.0	**35.2**	**24.9**	18.6	9.7	59.0	52.6	14.0	5.4	30.2	27.7
$\mathcal{M}+\mathcal{A}_1$ (supervised)	19.6	10.9	33.4	23.1	28.2	18.0	62.3	56.4	19.3	9.4	28.7	26.2
$\mathcal{M}^*$ (random, supervised)	12.2	5.1	8.3	2.7	10.8	4.2	49.1	41.1	11.7	3.9	26.4	24.2
CUB												
$\mathcal{M}$ (supervised finetuning)	14.1	6.4	7.5	2.7	39.1	28.2	50.0	42.7	16.0	6.7	34.8	32.1
$\mathcal{M}+\mathcal{A}_1$ (supervised)	16.4	8.4	9.2	3.7	**45.0**	**34.8**	58.7	52.9	23.2	12.6	36.5	33.7
$\mathcal{M}^*$ (random, supervised)	16.4	8.4	9.1	3.6	42.0	31.5	59.2	53.4	24.4	13.7	34.5	31.8
Flowers												
$\mathcal{M}$ (supervised finetuning)	14.7	6.7	7.3	2.5	21.1	11.6	68.7	63.5	15.9	6.7	34.5	31.8
$\mathcal{M}+\mathcal{A}_1$ (supervised)	17.3	8.9	9.1	3.5	29.9	19.3	**73.8**	**69.6**	23.0	12.5	36.1	33.4
$\mathcal{M}^*$ (random, supervised)	17.2	8.7	9.6	3.7	28.3	17.9	68.1	62.6	20.0	9.9	32.3	29.8
Food-101												
$\mathcal{M}$ (supervised finetuning)	8.4	2.8	5.0	1.4	8.7	2.9	44.0	35.8	30.9	18.8	25.0	22.8
$\mathcal{M}+\mathcal{A}_1$ (supervised)	16.7	8.6	8.5	3.1	35.1	24.3	56.0	49.4	**36.2**	**24.9**	35.2	32.5
$\mathcal{M}^*$ (random, supervised)	17.5	9.3	8.9	3.4	35.6	24.8	60.3	54.4	34.1	22.9	35.5	32.8
Products												
$\mathcal{M}$ (supervised finetuning)	7.2	2.3	4.9	1.6	5.6	1.5	36.6	27.9	8.8	2.2	42.4	39.7
$\mathcal{M}+\mathcal{A}_1$ (supervised)	12.1	5.2	7.0	2.4	20.8	11.2	53.0	46.0	15.0	6.1	**53.6**	**50.9**
$\mathcal{M}^*$ (random, supervised)	13.3	5.9	6.9	2.3	24.0	13.7	54.9	48.1	18.9	9.1	49.0	46.2

i.e. only on one of the six train splits that compose MRT. Our goal is to measure how much the performance of these specialized models drops when exposed to other retrieval tasks. For this, we evaluate each of these models on the remaining five datasets and report their performance in the off-diagonal part of Table I. We can see how *their performance drops significantly* compared to the pretrained DINO model, showing that they are overly suited to the retrieval task they have been trained on, and lose the generalization capability of the original pretrained model. It is also interesting to see that $\mathcal{M}+\mathcal{A}_1$ reports the lowest drops across all datasets, probably due to the fact that the underlying models are kept fixed and the adaptor's features are added in a residual fashion to the features of the backbone, retaining some of the generalization ability of the pretrained DINO model.

Supervised training jointly on all datasets. Finally, we measure how much the zero-shot ability of the underlying model is altered after adapting it with `Grappa` on multiple datasets (*e.g.* MRT) using label supervision. We compare with other adaptation strategies: supervised fine-tuning of all layers, supervised fine-tuning of a single adaptor, and supervised fine-tuning of 8 adaptors and a fusion layer. For this experiment, we use three new datasets that were not part of MRT: DTD [2], Eurosat [3], and Pets [4]. We report $\mathcal{R}$P and mAP in Table II.

Table II. **Performance of models trained on MRT, and tested on other retrieval tasks**. We measure the drop in generalization (zero-shot performance) after adapting the DINO model with Grappa versus alternative ways. Training is always performed on MRT, in a supervised manner for all methods (middle section), except for Grappa which is fully unsupervised (bottom section). We report MAP@R (mAP) and $\mathcal{R}$P.

Train \ Test	Supervised?	DTD		Eurosat		Pets	
		$\mathcal{R}$P	mAP	$\mathcal{R}$P	mAP	$\mathcal{R}$P	mAP
$\mathcal{M}$ (DINO – zero-shot model)	-	41.5	30.6	76.3	52.5	76.0	70.9
$\mathcal{M}$ (supervised finetuning)	✓	23.1	12.7	65.8	41.9	23.4	10.5
$\mathcal{M} + \mathcal{A}_1$ (supervised)	✓	39.2	28.5	76.6	52.8	74.1	68.7
$\mathcal{M}^*$ (random, supervised)	✓	39.5	28.7	75.8	51.7	73.2	67.5
Grappa-N	✗	41.3	30.5	75.2	51.0	74.4	68.9
drop versus DINO		(↓ 0.2)	(↓ 0.1)	(↓ 1.1)	(↓ 1.5)	(↓ 1.6)	(↓ 2.0)
gains over best adapted		(↑ 1.8)	(↑ 1.8)	(↓ 1.4)	(↓ 1.8)	(↑ 0.3)	(↑ 0.2)

Table III. **Performance of a model trained on MRT, using supervised learning of the fusion layer using class labels (supervised fusion).** We report MAP@R (mAP) and $\mathcal{R}$P on the six tasks of the MRT benchmark. We compare with Grappa-N which is unsupervised.

	Aircraft		Cars		CUB		Flowers		Food-101		Products	
	$\mathcal{R}$P	mAP	$\mathcal{R}$P	mAP	$\mathcal{R}$P	mAP	$\mathcal{R}$P	mAP	$\mathcal{R}$P	mAP	$\mathcal{R}$P	mAP
Grappa-N	18.1	9.5	9.9	4.0	35.1	24.1	67.2	61.9	30.5	19.3	36.3	33.6
Supervised fusion	19.8	10.7	11.1	4.7	34.7	23.7	67.6	62.2	30.8	19.5	43.2	40.3

First, we can see how little the zero-shot performance of Grappa drops compared to DINO, performing almost on par with DINO on DTD, and with only slightly larger drops on the other two datasets. Second, we can also see that compared to other supervised adaptation methods, retains a similar generalization ability.

C Supervised learning for adaptor fusion

We report additional results where the fusion layer over pseudo-granularity adaptors is learned in a supervised way. More precisely, for this experiment, we freeze adaptors $\mathcal{P}_1$-$\mathcal{P}_8$ and we learn the fusion layer on the training set of MRT using class labels, a normsoftmax loss, and adam optimizer. We report results on Table III. As expected, we observe this fusion to significantly improve over Grappa-N in most of the datasets (we again stress that Grappa does not use any kind of labels). The difference is especially large on Products, mostly likely due to imbalance issues; this dataset accounts for 97% of the total number of classes.

D Pseudocode for a Grappa layer

In Algorithm 1 we show pseudocode for a layer of the Grappa model, consisting on a ViT block, followed by a set of adaptors, and a fusion layer combining the

Algorithm 1 PyTorch-style pseudocode for a `Grappa` layer.

```
# B: batch size
# T: number of tokens
# D: dimensionality of the embeddings
# N: number of adaptors
#
# mm: matrix-matrix multiplication
# pool: average pooling
# MSA: multi-headed self attention
# MLP: multi-layer perceptron
# Adaptors: list of N bottleneck layers
# Q: learnable "query" DxD projection
# K: learnable "key" DxD projection
#
# h_1: input of size BxTxD

# ViT layer
x = MSA(LayerNorm(h_1)) + h_1
y = MLP(LayerNorm(x))
h = y + x  # BxTxD

# Compute N adaptors
U = []
for Adaptor in Adaptors:
    U.append(Adaptor(h) + y)
U = stack(U).permute(1, 2, 0, 3)  # BxTxNxD

# Fusion
q = mm(Q.T, pool(h, dim=1))  # BxD
k = mm(K.T, pool(U, dim=1))  # BxNxD
# Dot product between "query" and "key" to get the raw attention score
attn = mm(k.T, q.unsqueeze(-2))  # BxN
# Normalize the attention scores to probabilities
attn = softmax(attn, dim=1)
# Fuse adaptors using attention probabilities
f = mm(U.T, attn.unsqueeze(-2))  # BxTxD

h = f + x  # BxTxD
```

output of these adaptors. Details of this layer are also shown in Figure 3 in the main paper.

References

1. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. Proc. ICCV (2021)
2. Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., Vedaldi, A.: Describing textures in the wild. In: Proc. CVPR (2014)
3. Helber, P., Bischke, B., Dengel, A., Borth, D.: Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. JSTAEORS (2019)
4. Parkhi, O.M., Vedaldi, A., Zisserman, A., Jawahar, C.V.: Cats and dogs. In: Proc. CVPR (2012)