Supplementary Material: Learning Representations by Predicting Bags of Visual Words

Spyros Gidaris[1] Andrei Bursuc[1] Nikos Komodakis[2] Patrick Pérez[1] Matthieu Cord[1]

[1]Valeo.ai [2]University of Crete

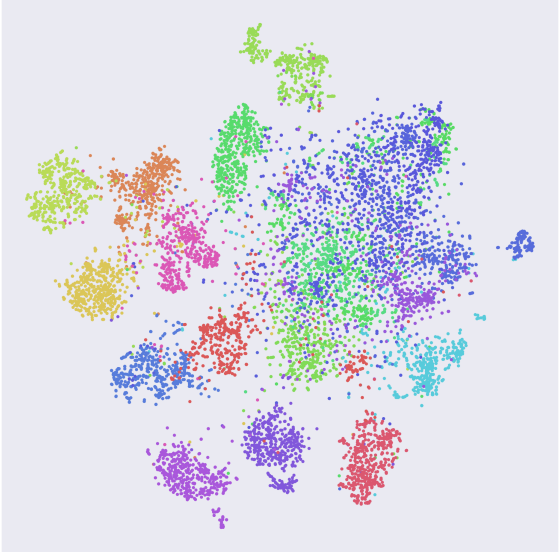

Figure 1: **t-SNE [6] scatter plot of the learnt self-supervised features on CIFAR-100.** Each data point in the t-SNE scatter plot corresponds to the self-supervised feature representation of an image from CIFAR-100 and is colored according to the class that this image belongs to. To reduce clutter, we visualize the features extracted from the images of 20 randomly selected classes of CIFAR-100.

Models	Linear
BoWNet $K = 512$	69.76
BoWNet $K = 1024$	70.43
BoWNet $K = 2048$	**71.01**
BoWNet $K = 4096$	70.99

Table 1: CIFAR-100 linear classifier results with WRN-28-10. Impact of vocabulary size. Here we used an initial version of our method implemented with less aggressive augmentation techniques.

Method	n =1	5	10	50	Linear
RotNet	58.3	74.8	78.3	81.9	60.3
BoWNet	69.1	86.3	89.2	92.4	71.5
Additional ablations					
BoWNet - predict $y()$	61.7	80.0	83.4	87.4	61.3
BoWNet - linear $\Omega(\cdot)$	70.4	85.6	88.3	90.7	63.3
BoWNet - binary	70.1	86.8	89.5	92.7	71.4

Table 2: CIFAR-100 linear classifier and few-shot results. For these results we use the WRN-28-10.

A. Visualizations

A.1. Visualizing the word clusters

In Figure 2 we illustrate visual words used for training our self-supervised method on ImageNet. Since we discover visual words using k-means, to visualize a visual word we depict the 16 patches with the smallest Euclidean distance to the visual word cluster centroid. As can be noticed, visual words encode mid-to-higher-level visual concepts.

A.2. t-SNE scatter plots of the learnt self-supervised features

In Figure 1 we visualize the t-SNE [6] scatter plot of the self-supervised features obtained when applying our method to the CIFAR-100 dataset. For visualizations purposes, we only plot the features corresponding to images that belong to 20 (randomly selected) classes out of the 100 classes of CIFAR-100. As can be clearly observed, the learnt features form class-specific clusters, which indicates that they capture semantic image information.

B. Additional experimental analysis

B.1. CIFAR-100 results

Here we provide an additional ablation analysis of our method on the CIFAR-100 dataset. As in section §4.1 of main paper, we use the WRN-28-10 [11] architecture.

Impact of vocabulary size. In Table 1 we report linear classification results for different vocabulary sizes K. We see that increasing the vocabulary size from $K = 256$ to $K = 2048$ in CIFAR-100, offers obvious performance im-

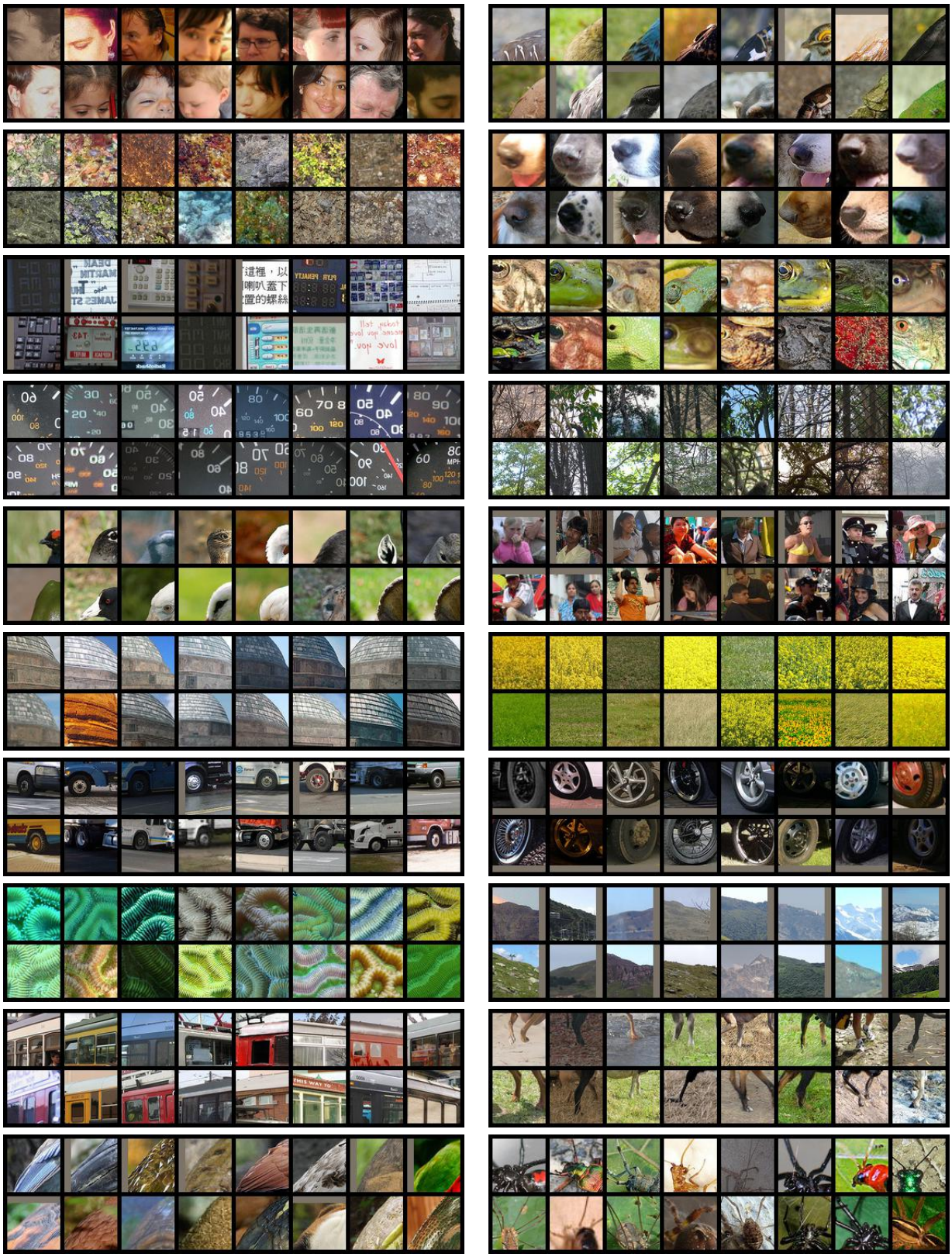

Figure 2: **Examples of visual word cluster members.** The clusters are created by applying k-means on the feature maps produced from the `conv4` layer of the ResNet50. For each visual word cluster we depict the 16 patch members with the smallest Euclidean distance to the visual word cluster centroid.

provements. Then, from $K = 2048$ to $K = 4096$, there is no additional improvement.

Reparametrized linear layer for $\Omega(\cdot)$. In section §2.3 of main paper, we described the linear-plus-softmax prediction layer $\Omega(\cdot)$ implemented with a reparametrized version of the standard linear layer. In this reparametrized version, instead of directly applying the weight vectors $W = [_1, \cdots, _K]$ to a feature vector $\Phi(\tilde{})$, we first L_2-normalize the weight vectors, and then apply a unique learnable magnitude γ for all the weight vectors (see equation (5) of main paper). The goal is to avoid always favoring the most frequently occurring words in the dataset by letting the linear layer of $\Omega(\cdot)$ learn a different magnitude for the weight vector of each word (which is what happens in the case of the standard linear layer). In terms of effect over the weight vector, this reparametrization is similar with the power-law normalization [8] used for mitigating the burstiness effect on BoW-like representations [4]. Here, we examine the impact of the chosen reparametrization by providing in Table 2 results for the case of implementing $\Omega(\cdot)$ with a standard linear layer (entry BoWNet - linear). We see that with the standard linear layer the performance of the BoWNet model deteriorates, especially on the linear classification metric, which validates our design of the $\Omega(\cdot)$ layer.

Predicting $y(\tilde{})$ instead of $y()$. In our work, given a perturbed image$\tilde{}$, we train a convnet to predict the BoW representation $y()$ of the original image . The purpose of predicting the BoW of the original image instead of the perturbed one$\tilde{}$, is to force the convnet to learn perturbation-invariant and context-aware features. We examine the impact of this choice by providing in Table 2 results for when the convnet is trained to predict the BoW of the perturbed image$\tilde{}$(entry BoWNet - predict $y(\tilde{})$). As expected, in this case there is a significant drop in the BoWNet performance.

Histogram BoW vs Binary BoW In section §2.2 of main paper, we described two ways for reducing the visual word description of an image to a BoW representation. Those are, **(1)** to count the number of times each word appears in the image (see equation (3)), called Histogram BoW, and **(2)** to just indicate for each word whether it appears in the image (see equation (4)) [9, 5], called Binary BoW. In Table 2 we provide evaluation results with both the histogram version (entry BoWNet) and the binary version (entry BoWNet - binary). We see that they achieve very similar linear classification and few-shot performance.

B.2. Small-scale experiments on ImageNet

Here we provide an additional experimental analysis of our method on the ImageNet dataset. Specifically, we study the impact of the feature block of the base convnet and the vocabulary size that are used for building the BoW representation. Due to the computationally intensive nature of ImageNet, we analyze those aspects of our method by performing "small-scale" ImageNet experiments. By "small-scale" we mean that we use the light-weight ResNet18 architecture and we train using only 20% of ImageNet training images, and for few epochs.

BoW from	linear cls	Vocabulary	linear cls
`conv3`	42.08	$K = 2048$	45.38
`conv4`	**45.38**	$K = 8192$	46.03
`conv5`	40.37	$K = 20000$	**46.45**
Vocabulary size $K = 2048$		**BoW from `conv4`**	

Table 3: ResNet18 small-scale experiments on ImageNet. Linear classifier results. The accuracy of the RotNet model used for building the BoW representations is 37.61. The left section explores the impact of the layer (of RotNet) that we use for building the BoW representations (with $K = 2048$). The right section explores the impact of the vocabulary size K (with BoW from the `conv4`).

Implementation details. We train the self-supervised models with SGD for 48 epochs. The learning rate is initialized at 0.1 and dropped by a factor of 10 after 15, 30, and 45 epochs. The batch size is 200 and weight decay $5e-4$.

Evaluation protocols. We evaluate the learned self-supervised representations by freezing them and then training on top of them 1000-way linear classifiers for the ImageNet classification task. The linear classifier is applied on top the feature map of the last residual block of ResNet18, resized to $512 \times 4 \times 4$ with adaptive average pooling. It is trained with SGD for 60 epochs using a learning rate of 0.1 that is dropped by a factor of 10 every 20 epochs. The weight decay is $1e-3$.

Results. We report results in Table 3. First, we study the impact on the quality of the learned representations of the RotNet feature block that is used for building the BoW representation. In the left section of Table 3 we report results for the cases of (a) `conv3` (2nd residual block), (b) `conv4` (3rd residual block), and `conv5` (4th residual block). We see that the best performance is for the `conv4`-based BoW. Furthermore, in the right section of Table 3 we examine the impact of the vocabulary size K on the quality of the learned representations. We see that increasing the vocabulary size from $K = 2048$ to $K = 20000$ leads to significant improvement for the linear classifier. In contrast, in Table 1 with results on CIFAR-100, we saw that increasing the vocabulary size after $K = 2048$ does not improve the quality of the learned representations. Therefore, it seems that the optimal vocabulary size depends on the complexity of the dataset to which we apply the BoW prediction task.

B.3. Full ImageNet and Places205 classification results

In Table 4 we provide the full experimental results of our method on the ImageNet and Places205 classification datasets.

C. Implementation details

C.1. Implementing RotNet

For the implementation of the rotation prediction network, RotNet, we follow the description and settings from Gidaris *et al*. [1]. RotNet is composed of a feature extractor $\hat{\Phi}(\cdot)$ and a rotation prediction module. The rotation prediction module gets as input the output feature maps of $\hat{\Phi}(\cdot)$ and is implemented as a convnet. It consists of a block of residual layers followed by global average pooling and a fully connected classification layer. In the CIFAR-100 experiments where $\hat{\Phi}(\cdot)$ is implemented with a WRN-28-10 architecture, the residual block of the rotation prediction module has 4 residual layers (similar to the last residual block of WRN-28-10), with 640 feature channels as input and output. In the MiniImageNet experiments where $\hat{\Phi}(\cdot)$ is implemented with a WRN-28-4 architecture, the residual block of the rotation prediction module has again 4 residual layers, but with 256 feature channels as input and output. Finally, in ImageNet experiments with ResNet50, the residual block of the rotation prediction module has 1 residual layer with 2048 feature channels as input and output.

During training for each image of a mini-batch, we generate its four rotated copies ($0°$, $90°$, $180°$, and $270°$ 2D rotations) and predict the rotation class of each copy. For supervision we use the cross-entropy loss over the four rotation classes.

After training we discard the rotation prediction module and consider only the feature extractor $\hat{\Phi}(\cdot)$ for the next stages, *i.e*. spatially dense descriptions and visual words.

C.2. Building BoW for self-supervised training

For the ImageNet experiments, given an image, we build its target BoW representation using visual words extracted from both the original and the horizontally flipped version of the image. Also, for faster training we pre-cache the BoW representations. Finally, in all experiments, when computing the target BoW representation we ignore the visual words that correspond to the feature vectors on the edge of the feature maps.

C.3. Few-shot protocol

The typical pipeline in few-shot learning is to first train a model on a set of *base* classes and then to evaluate it on a different set of *novel* classes (each set of classes is split into train and validation subsets). For MiniImageNet experiments we use this protocol, as this dataset has three different splits of classes: 64 training classes, 16 for validation, and 20 for test. For the few-shot experiments on CIFAR-100 we do not have such splits of classes and we adjust this protocol by selecting a subset of 20 classes and sample from the corresponding test images for evaluation. In this case, the feature extractor $\Phi(\cdot)$ is trained in a self-supervised manner on train images from all 100 classes of CIFAR-100.

Few-shot models are evaluated over a large number of few-shot tasks: we consider here $2,000$ tasks. The few-shot evaluation tasks are formed by first sampling t categories from the set of novel/evaluation classes and then selecting randomly n training samples and m test samples per category. The classification performance is measured on the $t \times m$ test images and is averaged over all sampled few-shot tasks. For few-shot experiments we use $t = 5$, $m = 15$, $n \in \{1, 5, 10, 50\}$.

References

[1] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *Int. Conf. on Learning Representations*, 2018. 4

[2] Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. In *Int. Conf. on Computer Vision*, 2019. 5

[3] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019. 5

[4] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. On the burstiness of visual elements. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2009. 3

[5] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Packing bag-of-features. In *Int. Conf. on Computer Vision*, 2009. 3

[6] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov), 2008. 1

[7] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. *arXiv preprint arXiv:1912.01991*, 2019. 5

[8] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *European Conf. on Computer Vision*, 2010. 3

[9] Josef Sivic and Andrew Zisserman. Video google: Efficient visual search of videos. In *Toward category-level object recognition*. Springer, 2006. 3

[10] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019. 5

[11] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conf.*, 2016. 1

[12] Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. Local aggregation for unsupervised learning of visual embeddings. In *Int. Conf. on Computer Vision*, 2019. 5

Method	ImageNet					Places205				
	conv2	conv3	conv4	conv5	pool5	conv2	conv3	conv4	conv5	pool5
Random [2]	13.7	12.0	8.0	5.6	-	16.6	15.5	11.6	9.0	-
Supervised methods										
ImageNet [2]	33.3	48.7	67.9	75.5	-	32.6	42.1	50.8	51.5	-
ImageNet*	32.8	47.0	67.2	76.0	76.2	35.2	42.6	50.9	52.8	52.0
Places205 [2]	31.7	46.0	58.2	51.7	-	32.3	43.2	54.7	62.3	-
Prior self-supervised methods										
RotNet*	30.1	42.0	52.5	46.2	40.6	32.9	40.1	45.0	42.0	39.4
Jigsaw [2]	28.0	39.9	45.7	34.2	-	28.8	36.8	41.2	34.4	-
Colorization [2]	24.1	31.4	39.6	35.2	-	28.4	30.2	31.3	30.4	-
LA† [12]	23.3	39.3	49.0	60.2	-	26.4	39.9	47.2	50.1	-
Concurrent work										
MoCo [3]	-	-	-	-	60.6	-	-	-	-	-
PIRL [7]	30.0	40.1	56.6	**63.6**	-	29.0	35.8	45.3	49.8	-
CMC‡ [10]	-	-	-	-	64.1	-	-	-	-	-
BowNet conv4	**34.4**	48.7	60.0	62.5	**62.1**	36.7	**44.7**	**50.5**	**50.9**	**51.1**
BowNet conv5	34.2	**49.1**	**60.5**	60.4	60.2	**36.9**	**44.7**	50.1	49.6	49.5

Table 4: top-1 center-crop linear classification accuracy on ImageNet and Places205. For the conv2–conv5 layers of , to evaluate linear classifiers, we resize their feature maps to around $9k$ dimensions (in the same way as in [2]). pool5 indicates the accuracy of the linear classifier trained on the 2048-dimensional feature vectors produced by the global average pooling layer after conv5. Entries in gray color are advantaged by either using 10-crops evaluation (method with †) or using two feature extractor networks (method with ‡). *: our implementation.