Supplementary Material for

Unsupervised Representation Learning for Binary Networks by Joint Classifier Training

Dahyun Kim[1,2] Jonghyun Choi[2,3,†]
[1]Upstage AI Research [2]NAVER AI Lab [3]Yonsei University
kdahyun@upstage.ai jc@yonsei.ac.kr

Note: We use blue color to refer to figures, tables, section numbers and citations **in the main paper** (*e.g.*, [17]). We use red or green colors to refer to figures, tables, section numbers and citations in this supplementary material.

1. Details on Downstream Task Configurations and Implementation Details (Section 4)

We present detailed explanations for each downstream task and the implementation. For the experiments, we strictly follow the experimental protocols from relevant prior work [6, 7, 17, 20, 46] whenever possible.

Object Detection. Following [20], we use the Faster R-CNN object detection framework. The Faster R-CNN framework is implemented using detectron2. We use Pascal VOC 2007 and Pascal VOC 2012 as the training dataset and evaluate on the Pascal VOC 2007 test set. We use the pretrained weights as the initial weights and fine-tune the entire detection framework. We use the exact same configuration file from [20].

Linear Evaluation. Following [17], we attach a linear classifier (a single fully-connected layer followed by a softmax) on top of the frozen backbone network and train only the classifier for 100 epochs using SGD. The initial learning rate is set to 30 and multiplied by 0.1 at epoch 60 and 80. The momentum is set to 0.9 with no weight decay. The classifier is trained on the target datasets *i.e.*, ImageNet and ImageNet100 [40].

Semi-Supervised Fine-Tuning. Following [6, 7, 46], we attach a linear classifier (a single fully-connected layer followed by a softmax) on top of the backbone network and fine-tune the backbone as well as the linear classifier using SGD for 20 epochs. Different initial learning rates are used for the backbone and the linear classifier. We select one from {0.1, 0.01, 0.001} for the backbone's initial learning rate. We multiply either {1, 10, 100} to the backbone's initial learning rate for the linear classifier initial learning rate. We found the performance for different pretraining methods to vary considerably for the different learning rate configurations and hence we sweep all the 9 combinations of initial learning rates described above and use the best configuration for each method for fine-tuning.

The momentum is set to 0.9 with a weight decay of 0.0005 and the learning rates for the backbone and the classifier are multiplied by 0.2 at epochs 12 and 16. For fine-tuning, only 1% or 10% of the labeled training images that are randomly sampled from the target datasets are used. The entire validation set is used for evaluation.

SVM Image Classification. Following [17], we first extract features from the backbone network and apply average pooling to match the feature vector dimension to be 4,096. Note that [17] uses ResNet50 as a backbone and extracts features after each residual block to report the best accuracy. In contrast, we are based on ReActNet [30] and extract features from the last layer as we found that to perform the best.

The feature vector dimension is 4,096 instead of 8,192 as in [17] because of the backbone architecture difference. With the extracted features, we use the LIBLINEAR [2] package to train linear SVMs. For the 'full-shot' classification, we use

[†] indicates corresponding author.
This work was done while DK and JC were an intern and an AI technical advisor at NAVER AI Lab.

the ‘trainval’ split of VOC07 dataset for training and evaluate on the ‘test’ split of VOC07 dataset. We report the mAP for the full-shot classification. For the few-shot classification, we use the ‘trainval’ split of VOC07 dataset in the few-shot setting for training and evaluate on the ‘test’ split of VOC07 dataset. The number of shots k (per class) varies from 1 to 96. We report average of mAP over five independent samples of the training data along with the standard deviation for the few-shot classification.

Transfer Learning. We perform linear evaluation on various target datasets. For object-centric datasets (*e.g.*, CIFAR10, CIFAR100, CUB-200-2011, Birdsnap), following [17], we attach a linear classifier (a single fully-connected layer followed by a softmax) on top of the frozen backbone network and train only the classifier for 100 epochs using SGD. The initial learning rate is set to 30 and multiplied by 0.1 at epoch 60 and 80. The momentum is set to 0.9 with no weight decay. The classifier is trained on the target datasets. For scene-centric datasets (*e.g.*, Places205), following [17], we attach a linear classifier on top of the frozen backbone network and train only the classifier. We train the classifier for 28 epochs using SGD. The learning rate is set to 0.01 initially and is multiplied by 0.1 at every 7 epochs. The momentum is set to 0.9 with weight decay of 0.00001. Note that different hyper-parameters are used from the object-centric datasets, following [17]. The classifier is trained on the target datasets.

Implementation Details. For our binary network backbone, we use authors’ implementation of the ReActNet-A [4] in all of our experiments. For pretraining, we use the LARS optimizer [5] with a batch size of $2,048$ for 200 epochs on ImageNet with the learning rate set as 0.3. For the FP network, we use a ResNet50 pretrained using MoCov2 [3] on ImageNet for 800 epochs. We use $\lambda_0 = 0.9$ and $\lambda_{T_{max}} = 0.7$ but other values performed similarly, given $\lambda_{T_{max}}$ was sufficiently smaller than λ_0. The implementation is largely based on an earlier version of [1].

2. Additional Comparisons to InfoMin and SimCLRv2 (Section 4)

We present additional comparisons to the InfoMin [41] and SimCLRv2 [7] with the ReActNet backbone to BURN (Ours) in Tables 1 and 2. Note that we use ImageNet100 [40] to pretrain the models instead of ImageNet for faster experiments. We use a ResNet50 trained for 600 epochs on ImageNet100 using BYOL as our pretrained FP network for BURN. We found that most methods perform poorly on the object detection and the transfer learning tasks, likely due to the the insufficient amount of pretraining data (*e.g.*, ImageNet100). Thus, we report results on linear evaluation, semi-supervised fine-tuning, and SVM image classification tasks but not on the object detection and transfer learning tasks.

Note that InfoMin and SimCLRv2 do not perform well in all the tested downstream tasks compared to BURN, with InfoMin scoring much lower across the board. We believe one reason for this is that the binary network lacks sufficient capacity to solve the instance discrimination task used in InfoMin and SimCLRv2; a similar problem was also found in lower capacity FP networks [1, 15].

Linear Evaluation. As shown in Table 1, BURN outperforms both InfoMin and SimCLRv2 by large margins, *i.e.*, over 30% for InfoMin and 14% for SimCLRv2 in the top-1 accuracy. Not only that, it even outperforms the supervised pretraining. InfoMin and SimCLRv2 perform poorly, with InfoMin scoring particularly low (see Section 2.1 for a detailed discussion).

Semi-Supervised Fine-tuning. As shown in Table 1, BURN outperforms InfoMin and SimCLRv2 by large margins (*e.g.*, over 40% for InfoMin and almost 20% for SimCLRv2 in top-1 accuracy in the 1% label setting). Additionally, BURN performs similar to the supervised pretraining whereas other compared SSL methods do not. Interestingly, SimCLRv2 performs much better than InfoMin, possibly because of the deeper projection layer, which was proposed in [7] to improve semi-supervised fine-tuning performance. In contrast, InfoMin performs poorly compared to other methods.

SVM Image Classification. We show the SVM image classification results for both the few-shot and full-shot (‘Full’) in Table 2. For the full-shot setting, BURN outperforms both InfoMin and SimCLRv2 by large margins, *i.e.*, over 21% for InfoMin and 7% for SimCLRv2 on mAP.

For the few-shot results, BURN outperforms both InfoMin and SimCLRv2 by large margins across all number of shots including the full-shot. Again, InfoMin performs worse than the other SSL methods as well, achieving mAPs that are well below the mAPs of BURN or SimCLRv2 in all numbers of shots (k) and the full-shot setting (please refer to Section 2.1 for discussion on the reason for InfoMin’s poor performance).

Pretrain on	Method	Linear Eval.	Semi-Supervised Fine-tuning 1% Labels		Semi-Supervised Fine-tuning 10% Labels	
		Top-1 (%)	Top-1 (%)	Top-5 (%)	Top-1 (%)	Top-5 (%)
ImgNet100	Supervised Pre.	76.54	63.10	86.24	75.40	92.16
	InfoMin [41]	45.38	21.68	46.74	32.06	59.82
	SimCLRv2 [7]	61.40	43.78	72.28	60.06	84.9
	BURN (Ours)	**77.02**	**62.60**	**84.75**	**74.00**	**91.38**

Table 1. Linear evaluation (top-1) and semi-supervised fine-tuning (1% labels or 10% labels) on ImageNet100 after pretraining. BURN outperforms all other SSL methods by large margins across for both the linear evaluation and semi-supervised fine-tuning. The best result among SSL methods is shown in **bold**.

Pretrain on	Method	k = 1	k = 2	k = 4	k = 8	k = 16	k = 32	k = 64	k = 96	Full
ImgNet100	Supervised Pre.	22.18 $\pm$ 1.31	28.77 $\pm$ 1.97	36.59 $\pm$ 1.61	43.67 $\pm$ 0.93	50.61 $\pm$ 0.62	55.75 $\pm$ 0.43	59.39$\pm$ 0.20	60.88$\pm$ 0.41	64.77
	InfoMin	14.12$\pm$ 0.23	17.07$\pm$0.93	20.76$\pm$ 0.91	24.75 $\pm$ 0.27	29.9 $\pm$0.73	35.12$\pm$0.52	39.2 $\pm$0.31	41.90$\pm$ 0.22	47.32
	SimCLRv2	17.97 $\pm$ 0.56	22.87 $\pm$2.0	30.48 $\pm$ 1.02	34.98 $\pm$1.58	42.9 $\pm$1.03	48.81 $\pm$0.67	53.87 $\pm$0.48	56.21 $\pm$0.25	61.36
	BURN (Ours)	**25.47 $\pm$ 1.46**	**30.68 $\pm$ 1.96**	**39.01 $\pm$ 1.28**	**46.13 $\pm$ 1.54**	**52.90 $\pm$ 0.81**	**58.29 $\pm$0.40**	**63.21 $\pm$ 0.30**	**64.96 $\pm$ 0.09**	**69.56**

Table 2. SVM image classification using SVM (mAP) on the VOC07 dataset after pretraining on ImageNet100 is shown. The number of shots (k) is varied from 1 to 96. We report average mAP over 5 runs with the standard deviation. BURN outperforms all methods including the supervised pretaining. The best result among SSL methods for each shot is shown in **bold**.

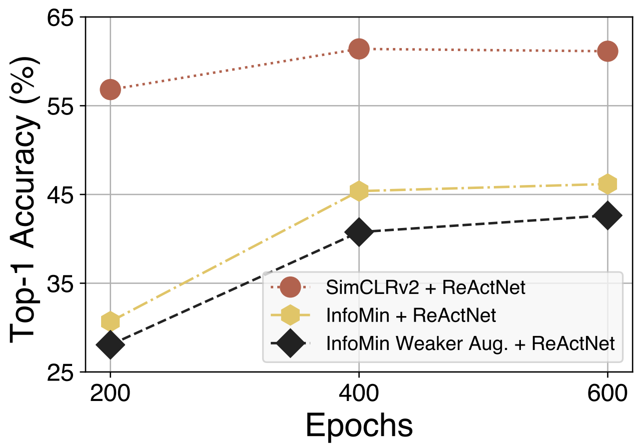


Figure 1. Comparison of linear evaluation performance on the ImageNet100 [40] dataset for InfoMin with a weaker augmentation strategy using ReActNet as the backbone. Weaker augmentation further degrades the performance of InfoMin.

2.1. Further Investigation on Performance Degradation of InfoMin (Follow-up of Section 2)

InfoMin with ReActNet as the backbone performs noticeably poorly as the results shown in Section 2. We first hypothesized in Section 2 that the performance degradation is due to the contrastive pretext task being too difficult for binary networks. However, that does not fully explain the additional performance discrepancy between InfoMin and Simclr2 as they both use the contrastive pretext task. Thus, we present further empirical analyses to investigate the performance degradation of InfoMin.

Note that the main differences between InfoMin and SimCLRv2 are 1) stronger augmentations and 2) the momentum encoder for supplying large amount of negative samples.

Effect of strong data augmentation. We first investigate the effects of using the 'stronger augmentations' used in InfoMin as they can distort the images excessively. The linear evaluation results (top-1 accuracy) are shown at epochs 200, 400 and 600 in Figure 1 for InfoMin with a weakened augmentation strategy, *e.g.*, only using random cropping, color distortion and Gaussian blur, following [6, 7, 46], instead of the stronger augmentation used in InfoMin. SimCLRv2 is also compared as

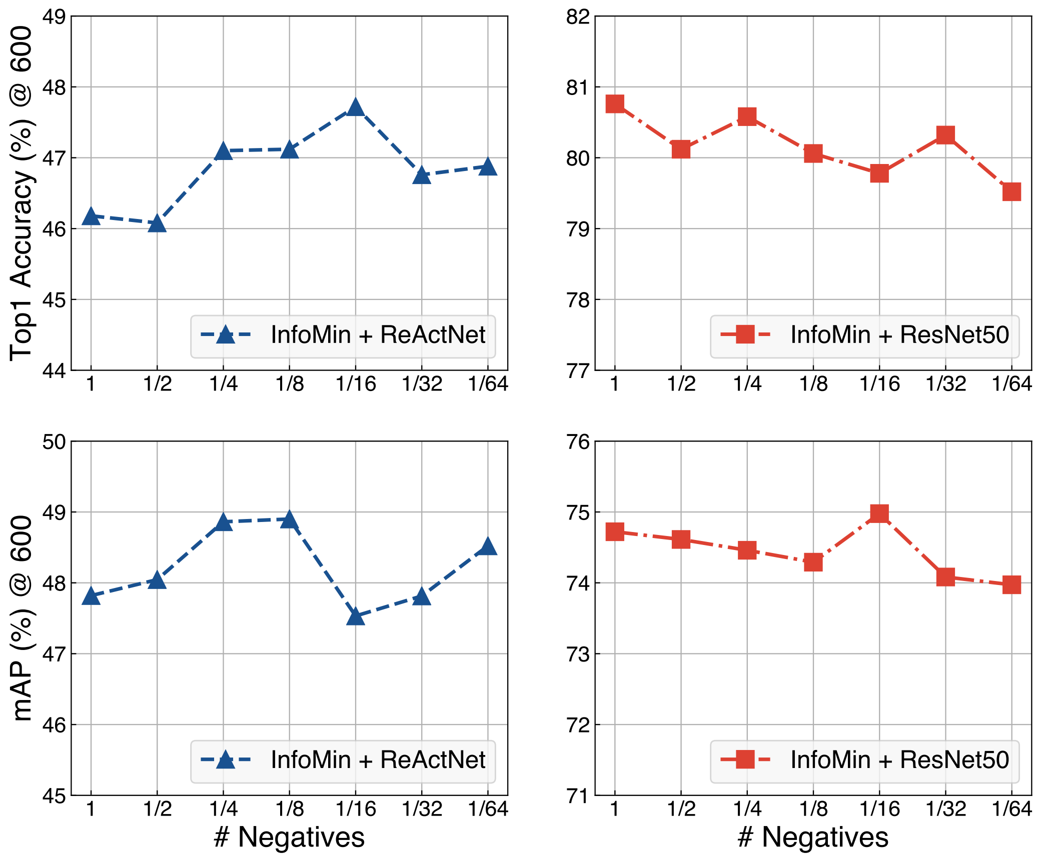
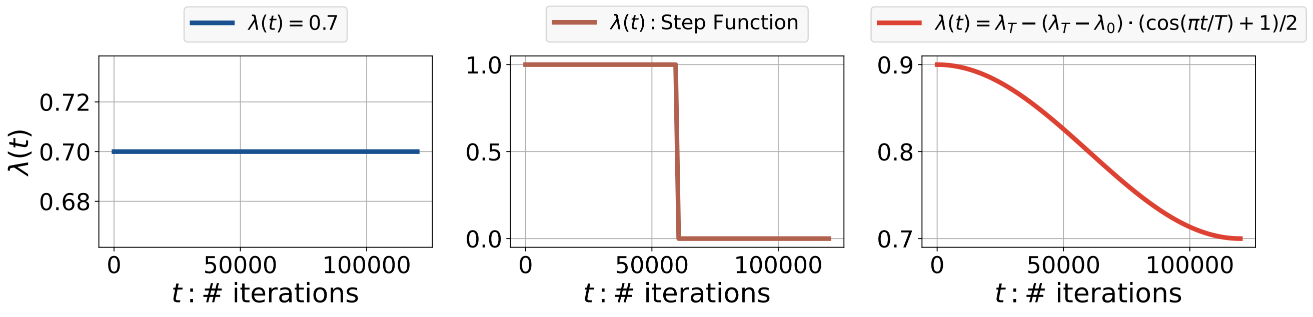


Figure 2. Analysis of the number of negatives (# Negatives) used in InfoMin with a binary network (ReActNet) or a FP network (ResNet50) on linear evaluation (top-1 (%) and SVM classification(mAP (%)) on the ImageNet100 [40] and VOC07 datasets at epoch 600. Decreasing the number of negatives is helpful for binary networks while not being beneficial for FP networks.



(a) Constant Function (b) Heaviside Step Function (c) Smooth Annealing Function

Figure 3. Plots for the various $\lambda(t)$ compared in Table 7 are shown.

a reference. As shown in the figure, weakening the augmentation strategy further degrades the performance of InfoMin at epochs 200, 400, and 600. This implies that the stronger augmentation strategy used by InfoMin may not be the main cause of InfoMin's sudden performance drop. Hence, the comparison motivates us to further investigate the use of the momentum encoder with binary networks.

Usage of momentum encoder. We then suspect that the use of momentum encoder to supply a large amount of negatives may not be beneficial in the binary case, as is also discussed in Sec. 4.2 when we investigated various choices for a moving target network. Thus, using a momentum encoder may make the features of the negative samples too different to be helpful when it is used with binary networks, especially when the number of negative samples is large (65,536). To empirically validate this, we vary the number of negative samples used in InfoMin to see the performance trend when either ReActNet (binary) or ResNet50 (FP) is used as the backbone in Fig. 2.

As shown in the figures, InfoMin with ReActNet performs the best with a substantially smaller # negatives ($1/16$) while with ResNet50 it performs worse with less # negatives. For further analysis of the effect made by the reduced number of negative samples, we evaluate the performance of SVM classification on the VOC07 dataset in the second row of Fig. 2. We observe similar trends in the SVM classification results. While the above comparisons do not fully explain the InfoMin's performance drop, it is still interesting to see that using a momentum encoder to supply a large number of negatives may not be as helpful for binary networks.

3. Plots of Various $\lambda(t)$ (Table 7 in Section 4.2)

In Fig. 3, we plot (1) the constant function, (2) the Heaviside step function that is shifted and horizontally reflected, *i.e.*, $H(-t + T\text{max}/2)$ with T_{max} being the maximum iterations, and (3) the smooth annealing function (Eq. 3) to visualize the various choices of $\lambda(t)$ compared in Table 7.

References

[1] MMSelfSup Contributors. MMSelfSup: Openmmlab self-supervised learning toolbox and benchmark. `https://github.com/open-mmlab/mmselfsup`, 2021. 2

[2] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 2008. 1

[3] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 2

[4] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *ECCV*, 2020. 2

[5] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017. 2