Multi-task Self-Supervised Visual Learning
(2017)

Carl Doersch, Andrew Zisserman
Notes

Contributions

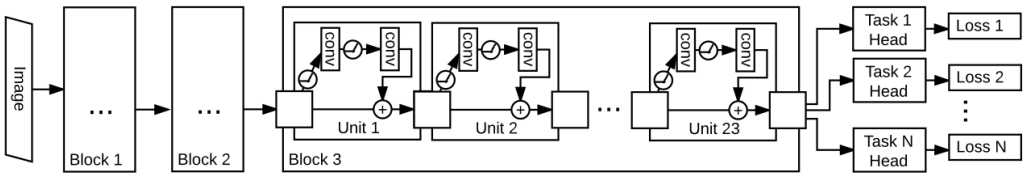
The authors investigates the possibility of using multi-task learning in a self supervised setting, i.e. combining multiple self supervised tasks. More specifically, the authors propose to use the following tasks:

- *Relative position:* in which the network predicts the position of patch with respect to the center patch. The samples are sampled randomly from a given image, and the network output a probability over 8 position of the second patch relative to the first one, in such task, color dropping (randomly dropping 2 out of 3 channels) is necessary to avoid chromatic aberration.
- *Colorization*: prediction the `ab` colors (from the Lab color space) from gray scale input image. The problem is formulated as classification task, outputting a probability over 313 possible values of `ab` colors.
- *Exemplar:* First we create a set of pseudo classes, each class is a given patch from and image and its augmenting using different data augmentations (color shifts, scaling, rotation, translation). Instead of training the network in a classification setting, the authors propose to use a triplet loss: with two positive patches from the same pseudo class x_1 and x_2, and a negative patch x_3.
- *Motion segmentation:* Given a single frame, the objective is to predict which pixels will move in the subsequent frames. The ground truth is generated using a dense tracking algorithm, and the network is trained using per-pixel cross entropy with two classes.

And then train a single network for unsupervised representation learning. They introduce some regularization techniques to force the network to learn factorized representations. This multi-task network is compared to other stand alone self-supervised tasks, this time using the same backbone (ResNet 101) and a unified same implementation.

Method

Model All the task share the same backbone, a ResNet 101, using only three of the four ResNet blocks to reduce computations. Extra head are added on top of the shared encoder, each one is specific to a given task and its specific loss. For exemplar task, the head is a Siamese network. At each training iteration only one head is active but the gradients are averaged across different iterations where different heads are active.

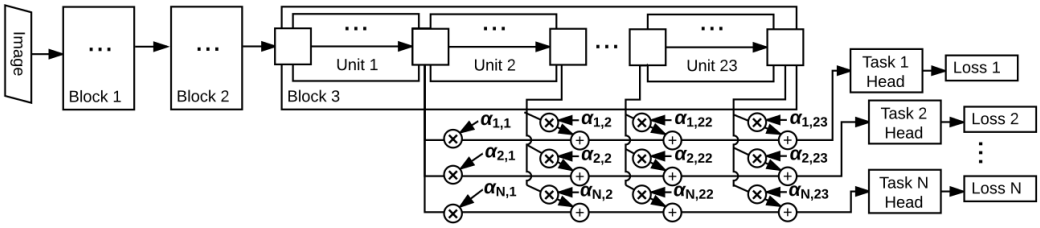


Conflicts In a multi-task learning setting, some challenges might rise: different tasks learn at different rates, and the possible conflicts between the tasks, such as *input channel* conflict (colorization: the inputs are gray scale, exemplar: the inputs are colored images), *learning tasks* conflict (one task

generalizes across instances of a class, the other should not), and given that different tasks require different features, for either self-supervised tasks or evaluation tasks. Fined grained information is important for classification / colorization but not for object detection / relative positioning.

Solving conflicts

- The authors propose to separate the features to be used for each tasks j using a set of weights $\alpha_{i,j}$ where $i \in [1, 23]$ and j in the number of tasks. For each features of layer i of the resnet block 3. These features are weighted by $\alpha_{i,j}$ and summed together: $\sum_{m=1}^{M} \alpha_{n,m} * Unit_m$, and the results are then fed as inputs for the head of task j. The set of weights are learned and are also subject to two constraints: sparsity using an $L1$ regularization and controlling the output variance $\sum_{m=1}^{M} \alpha_{n,m}^2 = 1$.



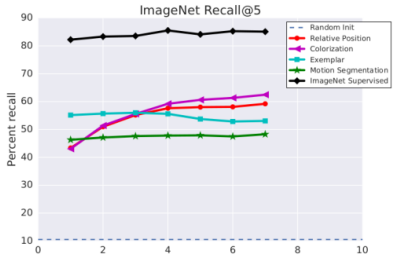
- To solve the problem of *input channel* conflict, the data-preprocessing is harmonized so that the inputs are adjusted for all tasks , for example in relative positioning, color jiterring is essential, instead of randomly dropping two color channels, the ab channels are dropped and L channel in duplicated, which is the same required input for colorization.

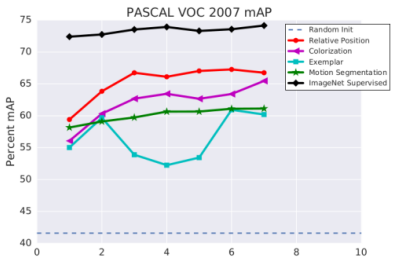
Results

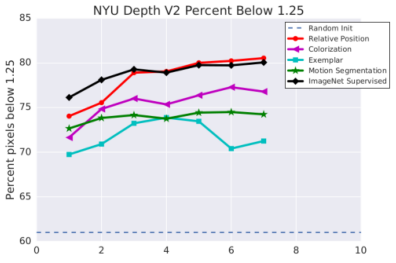
Evaluation Procedure

- Freezing the weights: the weights of the resnet (without any heads) are frozen, a classification layers is added on top and is trained on ImageNet
- Pascal-Voc detection: The backbone is used in a Faster RCNN network, and is fine-tuned for object detection.
- NYU V2 depth prediction: The output of the resnet is fed into an up-projection network for depth prediction.

Single Task







Pre-training	ImageNet top1		ImageNet top5	PASCAL		NYU
	Prev.	Ours	Ours	Prev.	Ours	Ours
Relative Position	31.7[46]	36.21	59.21	61.7 [7]	66.75	80.54
Color	32.6[46]	39.62	62.48	46.9[46]	65.47	76.79
Exemplar	-	31.51	53.08	-	60.94	69.57
Motion Segmentation	-	27.62	48.29	52.2[27]	61.13	74.24
INet Labels	51.0[46]	66.82	85.10	69.9[34]	74.17	80.06

Multi-Task: Naive combination

Pre-training	ImageNet	PASCAL	NYU
RP	59.21	66.75	**80.54**
RP+Col	66.64	68.75	79.87
RP+Ex	65.24	69.44	78.70
RP+MS	63.73	68.81	78.72
RP+Col+Ex	68.65	69.48	80.17
RP+Col+Ex+MS	**69.30**	**70.53**	79.25
INet Labels	85.10	74.17	80.06

Table 2. Comparison of various combinations of self-supervised tasks. Checkpoints were taken after 16.8K GPU hours, equivalent to checkpoint 7 in Figure 3. Abbreviation key: RP: Relative Position; Col: Colorization; Ex: Exemplar Nets; MS: Motion Segmentation. Metrics: ImageNet: Recall@5; PASCAL: mAP; NYU: % Pixels below 1.25.

Multi-Task: Harmonization & Lasso training

Pre-training	ImageNet	PASCAL	NYU
RP	59.21	66.75	80.23
RP / H	62.33	66.15	80.39
RP+Col	66.64	68.75	79.87
RP+Col / H	68.08	68.26	79.69

Table 3. Comparison of methods with and without harmonization, where relative position training is converted to grayscale to mimic the inputs to the colorization network. H denotes an experiment done with harmonization.

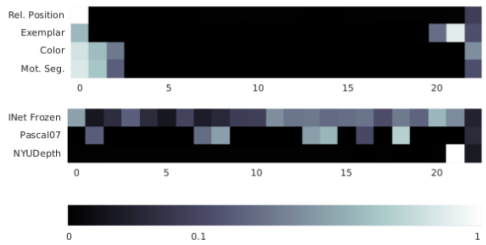


Figure 5. Weights learned via the lasso technique. Each row shows one task: self-supervised tasks on top, evaluation tasks on bottom. Each square shows $|\alpha|$ for one ResNet "Unit" (shallowest layers at the left). Whiter colors indicate higher $|\alpha|$, with a nonlinear scale to make smaller nonzero values easily visible.

Net structure	ImageNet	PASCAL	NYU
No Lasso	69.30	**70.53**	79.25
Eval Only Lasso	**70.18**	68.86	79.41
Pre-train Only Lasso	68.09	68.49	78.96
Pre-train & Eval Lasso	69.44	68.98	79.45

Table 4. Comparison of performance with and without the lasso technique for factorizing representations, for a network trained on all four self-supervised tasks for 16.8K GPU-hours. "No Lasso" is equivalent to table 2's RP+Col+Ex+MS. "Eval Only" uses the same pre-trained network, with lasso used only on the evaluation task, while "Pre-train Only" uses it only during pre-training. The final row uses lasso always.