Invariant Information Clustering for Unsupervised Image Classification and Segmentation: Supplementary material

Xu Ji
University of Oxford
xuji@robots.ox.ac.uk

João F. Henriques
University of Oxford
joao@robots.ox.ac.uk

Andrea Vedaldi
University of Oxford
vedaldi@robots.ox.ac.uk

1. Release

We implemented IIC in PyTorch [6]. The code, datasets and trained models have been released online.

2. Further experimental details

We used three generic CNN bases b across our experiments: A (ResNet34 [4]), B (4 convolutional layers) and C (6 convolutional layers). For details see table 1. See table 2 for per-experiment details including b, batch size, input channels, input size, and number of clusters used in overclustering denoted by k. Recall the latter refers to the sole output head for semi-supervised overclustering but to the auxiliary head for unsupervised IIC, where the main head produces output with dimensionality k_{gt}. For segmentation, bilinear resampling is used to resize the network output back to input size for implementational simplicity. Since there is one pooling layer in network C which halves spatial size, this is by a factor of 2.

A	B	C
1×Conv@64	1× Conv@64	1× Conv@64
3×BasicBlock@64	1× MaxPool	1× Conv@128
4×BasicBlock@128	1× Conv@128	1× MaxPool
6×BasicBlock@256	1× MaxPool	2× Conv@256
3×BasicBlock@512	1× Conv@256	2× Conv@512
1×AvgPool	1× MaxPool	
	1× Conv@512	

Table 1: Architecture bases b, showing layer type and output channels. Pooling layers do not change channel size. Convolutional layers have filter size 3 or 5 and stride 1 or 2. The models used are standard ResNet and VGG-style networks. Implementations are given in the code.

3. Semi-supervised overclustering study

Paper fig. 6 contains accuracies normalised by dividing by the maximum accuracy for each series. The absolute accuracies are given in table 3 and table 4.

		b	n	h	r	k_{in}	k_{gt}	k	crop size(s)	input size
IIC	STL10	A	700	5	5	2	10	70	64	64
	CIFAR10	A	660	5	3	2	10	70	20	32
	CIFAR100-20	A	1000	5	5	2	20	140	20	32
	MNIST	B	700	5	5	1	10	50	16, 20, 24	24
	COCO-Stuff-3	C	120	1	1	5	3	15	128	128
	COCO-Stuff	C	60	1	1	5	15	45	128	128
	Potsdam-3	C	75	1	1	4	3	24	200	200
	Potsdam	C	60	1	1	4	6	36	200	200
IIC*	STL10	A	1400	5	5	2	10	140	64	64
	CIFAR10	A	1320	5	3	2	10	140	20	32
	CIFAR100-20	B	2800	5	5	5	20	280	20	24
	MNIST	B	350	5	5	1	10	25	16, 20, 24	24
	COCO-Stuff-3	C	180	1	1	5	3	15	128	128
	COCO-Stuff	C	90	1	1	5	15	45	128	128
	Potsdam-3	C	75	1	1	4	3	9	200	200
	Potsdam	C	60	1	1	4	6	24	200	200

Table 2: IIC denotes unsupervised clustering, IIC* denotes semi-supervised overclustering. n denotes batch size, h and r denote number of sub-heads and sample repeats (see paper section 4.1), k_{in} denotes input channels (1 for greyscale, 2 for Sobel filtered, 4 for RGBIR, 5 for Sobel filtered with RGB), k_{gt} denotes number of ground truth clusters, k denotes number of output channels for overclustering. COCO-Stuff and COCO-Stuff-3 are scaled by 0.33 prior to cropping; cropped images are scaled to final input size with bilinear resampling.

4. Rendering predictions

To generate the visualisation in paper fig. 3, the entire MNIST dataset was run through each network snapshot. The prediction for each image x, say $z = \Phi(x) \in [0,1]^C$ for C classes (see paper section 3.1), was rendered as a point with coordinate position p:

$$p = \left[\sum_{c=1}^{C} z_c \cdot sin\left(\frac{2\pi c}{C}\right), \sum_{c=1}^{C} z_c \cdot cos\left(\frac{2\pi c}{C}\right)\right].$$

	STL10		CIFAR10		CIFAR100-20		CIFAR100		MNIST	
% of max k	k	ACC	k	ACC	k	ACC	k	ACC	k	ACC
100	140	**63.1**	140	**65.0**	280	**34.7**	1000	**20.3**	100	98.6
50	70	61.4	70	62.2	140	33.1	500	20.3	50	98.6
25	35	59.7	35	60.5	70	30.0	250	19.1	25	**98.7**
12.5	18	54.8	18	53.7	35	25.7	125	15.0	13	97.9

Table 3: Absolute accuracy for semi-supervised overclustering experiments in paper fig. 6-right.

STL10	1.0		0.5		0.25		0.1		0.01	
% of max k	n_a	ACC	n_a	ACC	n_a	ACC	n_a	ACC	n_a	ACC
100	5000	**63.1**	2500	**61.0**	1250	58.6	500	52.4	50	25.5
50	5000	61.4	2500	59.8	1250	**59.1**	500	**57.8**	50	30.7
25	5000	59.7	2500	59.2	1250	58.5	500	57.6	50	**44.1**
12.5	5000	54.8	2500	54.8	1250	54.1	500	50.6	50	41.3

	1.0		0.5		0.25		0.1		0.01	
	n_a	ACC	n_a	ACC	n_a	ACC	n_a	ACC	n_a	ACC
STL10	5000	**63.1**	2500	61.0	1250	58.6	500	52.4	50	25.5
CIFAR10	50000	**62.9**	25000	62.7	12500	62.6	5000	62.0	500	53.9
CIFAR100-20	50000	**34.5**	25000	34.0	12500	33.6	5000	31.9	500	20.1
CIFAR100	50000	**20.3**	25000	19.2	12500	17.9	5000	15.1	500	7.43
MNIST-25	60000	**98.9**	30000	98.9	15000	98.9	6000	98.9	600	98.9

Table 4: Absolute accuracy for semi-supervised overclustering experiments in paper fig. 6-left (top) and fig. 6-center (bottom). n_a denotes number of labels used to find mapping from output k to k_{gt} for evaluation.

5. Optional entropy coefficient

Consider inserting a coefficient, $\lambda \geq 1$, into the definition of mutual information (eq. 3, paper section 3.1):

$$I_\lambda(z, z') = \sum_{c=1}^{C} \sum_{c'=1}^{C} \mathbf{P}_{cc'} \cdot \ln \frac{\mathbf{P}_{cc'}}{\mathbf{P}_c^\lambda \cdot \mathbf{P}_{c'}^\lambda} \quad (1)$$

$$= I_1(z, z') + (\lambda - 1) \cdot (H(z) + H(z')). \quad (2)$$

For $\lambda = 1$, this reduces to the standard mutual information definition. However, inserting an exponent of $\lambda > 1$ into the denominator of (1) translates into prioritising the maximisation of prediction entropy (2).

6. Expectation over all shifts $t \in T$

Recall that IIC for segmentation involves maximising mutual information between a patch and all its neighbours within local box given by T (paper section 3.3). An alternative formulation of paper eq. (5) would involve bringing the expectation over T within the computation for information as follows:

$$\max_{\Phi} I(\mathbf{P}),$$

$$\mathbf{P} = \frac{1}{n|T||G||\Omega|} \sum_{i=1}^{n} \sum_{t \in T} \sum_{g \in G} \overbrace{\sum_{u \in \Omega} \Phi_u(\mathbf{x}_i) \cdot [g^{-1}\Phi(g\mathbf{x}_i)]_{u+t}^\top}^{\text{Convolution}}.$$

We found paper eq. (5) to work marginally but consistently better, for example by 0.1% for COCO-Stuff-3 and 0.02% for Potsdam-3. This is likely because closer neighbours are more informative than farther ones, and an external expectation avoids entangling the signal between close and far neighbours prior to computing mutual information.

7. Random transformations g

Horizontal flipping, random crops and random colour changes in hue, saturation and brightness constitute the g used in most of our experiments. We also tried random affine transforms but found our models performed better without them, as the presence of skew and scaling materially affected the network's ability to distill visual correspondences between pairs of images.

8. Dataset sizes

For the sizes of the training and testing sets used in our experiments, see table 5 and table 6.

	STL10		CIFAR10		CIFAR100-20		MNIST	
	Train	Test	Train	Test	Train	Test	Train	Test
IIC	113k	13k	60k	60k	60k	60k	70k	70k
Semi-supervised	105k	8k	50k	10k	50k	10k	60k	10k

Table 5: Datasets for image clustering.

	COCO-Stuff-15		COCO-Stuff-3		Potsdam-6		Potsdam-3	
	Train	Test	Train	Test	Train	Test	Train	Test
IIC	51804	51804	36660	36660	8550	5400	8550	5400
Semi-supervised	49629	2175	35228	1432	7695	855	7695	855

Table 6: Datasets for segmentation.

9. Baseline experiments

DeepCluster [1], also originally implemented in PyTorch, was adapted from the released image clustering code for both purely unsupervised image clustering and segmentation. Since this is not the intended task for the method, DeepCluster was used as a feature learner, with k-means performed on learned feature representations in order to obtain cluster assignments for evaluation. Data augmentation transforms are used as with IIC, the same b as IIC is used for each model's feature representation, and the number of output clusters is set to $10 \times k_{gt}$ as suggested by the paper. The feature descriptor lengths range from 4096 (image clustering) to 512 (segmentation). For image clustering, the k-means procedures at training and test time are both trained and evaluated on the full training and test sets respectively. For segmentation, since all descriptors for the training set cannot fit in RAM (needed not only for the implementation of k-means, but also for the PCA dimensionality reduction) it was necessary to use sampling for k-means both during computation of the pseudolabels for training, and evaluation. This was done with 10M and 50M samples for Potsdam* and COCO-Stuff* datasets respectively. Once the k-means centroids were obtained, training still occured over

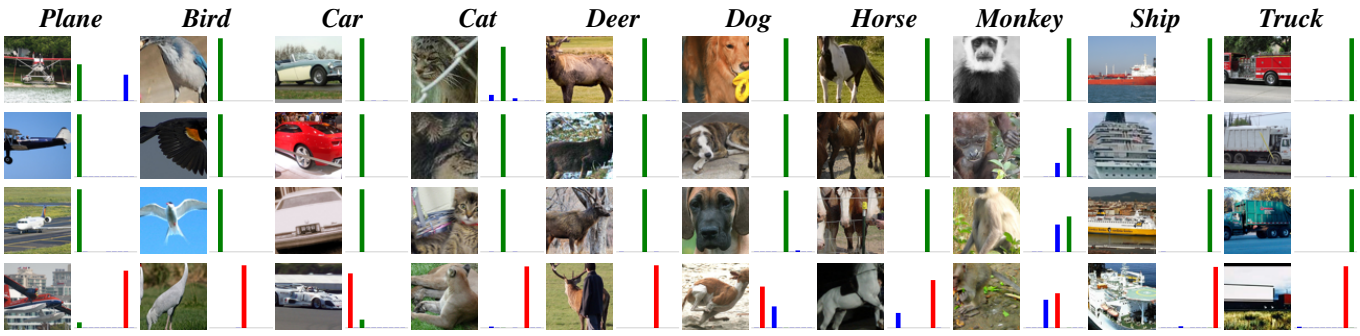

Figure 1: Additional unsupervised clustering (IIC) results on STL10. Predicted cluster probabilities shown as bars. Prediction corresponds to tallest, ground truth is green, incorrectly predicted classes are red, and all others are blue. The bottom row shows failure cases.

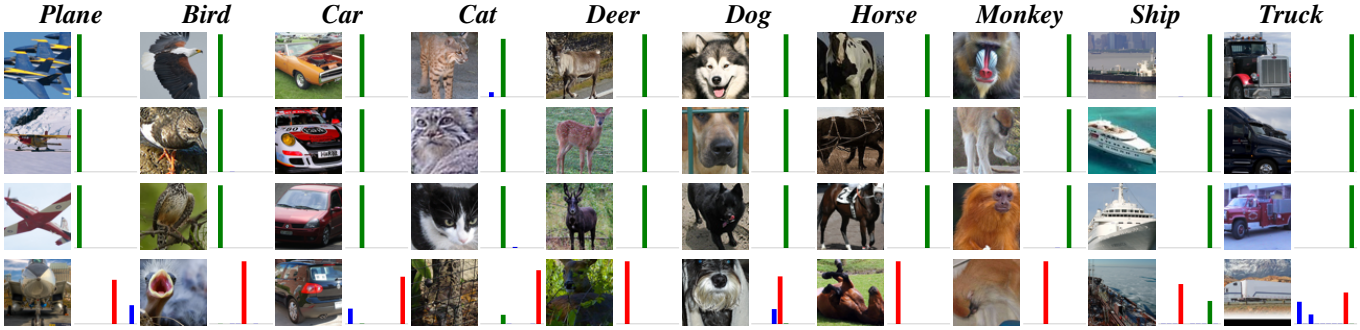

Figure 2: Semi-supervised overclustering results on STL10. Predicted cluster probabilities shown as bars. Prediction corresponds to tallest, ground truth is green, incorrectly predicted classes are red, and all others are blue. The bottom row shows failure cases.

Figure 3: Additional unsupervised segmentation (IIC) results on COCO-Stuff-3 (non-stuff pixels in black). Left to right for each triplet: image, prediction, ground truth.

Figure 4: Additional semi-supervised clustering for segmentation results on COCO-Stuff-3 (non-stuff pixels in black). Left to right for each triplet: image, prediction, ground truth.

the entire training set with accuracy computed over the entire test set. For the semi-supervised experiment, finetuning of the learned representation was used, as with IIC.

ADC [3], originally implemented in TensorFlow, was adapted from the released code for image clustering only. For the fully unsupervised CIFAR100-20 experiment (paper table 1), since ADC was already implemented for CIFAR100, we adopted the existing architecture and training settings for CIFAR100 when training CIFAR100-20. Similarly, we adopted the existing architecture and settings included for STL10 for the semi-supervised experiment, training an SVM on top of fixed features as this is the semi-supervised implementation provided in their code.

Triplets [7] was implemented as a representation learner by setting the positive example for each image to be its random transform, using the same transformations as the IIC experiments for fairness. The negative example for each image was set to a randomly selected image. K-means was run

Figure 5: Additional segmentation results for unsupervised IIC and semi-supervised overclustering on Potsdam-3. Left to right for each quadruplet: image, IIC prediction, semi-supervised overclustering prediction, ground truth.

on the learned embeddings to obtain cluster assignments.

For segmentation baselines Isola [5] and Doersch [2], which are unsupervised feature learning methods without segmentation code, we use our own implementation. Since both operate by predicting the spatial relationship between pairs of patches (spatial proximity and exact relative position respectively), we adapted them to segmentation by randomly sampling pairs from the dense features produced by b (which are either close or far for Isola, for example), using additional linear layers to predict the spatial relationship, minimising the distance between this prediction and known ground truth, and backpropagating gradients end-to-end.

10. Additional examples

For more examples of image clustering and segmentation results for both unsupervised IIC and semi-supervised overclustering, see fig. 1, fig. 2, fig. 3, fig. 4 and fig. 5.

References

[1] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. *arXiv preprint arXiv:1807.05520*, 2018. 2

[2] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015. 4

[3] P. Haeusser, J. Plapp, V. Golkov, E. Aljalbout, and D. Cremers. Associative deep clustering - training a classification network with no labels. In *Proc. of the German Conference on Pattern Recognition (GCPR)*, October 2018. 3

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[5] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson. Learning visual groups from co-occurrences in space and time. *arXiv preprint arXiv:1511.06811*, 2015. 4

[6] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 1

[7] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *Advances in neural information processing systems*, pages 41–48, 2004. 3