

Supplemental Material

S1 Training Details

Hyper-parameter settings for RODEO and the offline models for VOC and COCO are given in Table S1. Similarly, run time comparisons for the COCO dataset are in Table S2.

Table S1: Training parameter settings for RODEO and offline models.

PARAMETERS	VOC	COCO
Optimizer	SGD	SGD
Learning Rate	0.001	0.001
Momentum	0.9	0.9
Weight Decay	5e-4	5e-4
Offline Batch Size	2	2
Offline Epochs	25	10

Table S2: Training-time comparison of models.

METHOD	TIME(HOUR)
Fine-Tune	4.2
SLDA+Regress	2.0
RODEO	21.7
Offline	873.2

S2 Where to Quantize?

Our choices of layers to quantize are limited due to the architecture of the ResNet-50 backbone. ResNet-50 has four main major layers with each having (3,4,6,3) bottleneck blocks respectively. Since bottleneck blocks add a residual shortcut connection at the end, it is not possible to quantize from the middle of the block, leaving only four places to perform quantization. Quantizing earlier has some advantages since it leaves more trainable parameters for the incremental model, which could lead to better results [17]. But, it also requires twice the memory to store the same number of images as we move towards the earlier layers. For efficiency, we choose the last layer for feature quantization.

S3 Additional Results

We provide the individual mAP results for each increment of COCO in Table S3 and VOC in Table S4.

Table S3: Incremental mAP results for COCO evaluated after learning every 10 classes.

METHOD	1-40	50	60	70	80	MEAN	Ω_{mAP}
Fine-Tune	0.421	0.011	0.001	0.028	0.002	0.093	0.220
SLDA+Regress	0.351	0.300	0.260	0.233	0.233	0.275	0.655
RODEO $n = 4$ (recon, BAL)	0.380	0.347	0.306	0.302	0.313	0.330	0.784
RODEO $n = 4$ (recon, MIN)	0.380	0.355	0.353	0.325	0.329	0.348	0.829
RODEO $n = 12$ (recon, BAL)	0.380	0.311	0.296	0.283	0.289	0.312	0.741
RODEO $n = 12$ (recon, MIN)	0.380	0.317	0.320	0.293	0.289	0.320	0.760
RODEO $n = 4$ (recon, MAX)	0.380	0.015	0.060	0.064	0.074	0.119	0.282
RODEO $n = 4$ (recon, RANDOM)	0.380	0.275	0.241	0.203	0.158	0.251	0.598
RODEO $n = 4$ (real, BAL)	0.421	0.356	0.333	0.313	0.326	0.350	0.831
RODEO $n = 4$ (real, MIN)	0.421	0.372	0.359	0.339	0.339	0.366	0.870
RODEO $n = 12$ (real, BAL)	0.421	0.312	0.305	0.288	0.302	0.325	0.774
RODEO $n = 12$ (real, MIN)	0.421	0.328	0.330	0.318	0.312	0.342	0.812
RODEO $n = 4$ (real, NO-REPLACE)	0.421	0.406	0.380	0.362	0.383	0.390	0.928
Offline	0.421	-	-	-	0.420	-	1.000

Table S4: Incremental mAP results for the addition of each class in VOC dataset.

METHOD	BASE INIT.	+TABLE	+DOG	+HORSE	+MBIKE	+PERSN	+PLANT	+SHEEP	+SOFA	+TRAIN	+TV
Fine-Tune	0.709	0.270	0.277	0.263	0.253	0.24	0.228	0.220	0.208	0.201	0.196
ILwFOD	0.671	0.651	0.625	0.599	0.598	0.592	0.573	0.491	0.498	0.487	0.490
SLDA+Regress	0.665	0.603	0.574	0.540	0.524	0.490	0.457	0.443	0.430	0.418	0.409
RODEO $n = 4$ (recon)	0.614	0.596	0.582	0.602	0.650	0.667	0.635	0.581	0.635	0.620	0.617
RODEO $n = 12$ (recon)	0.613	0.599	0.656	0.655	0.694	0.705	0.679	0.658	0.644	0.656	0.667
RODEO $n = 12$ (real)	0.702	0.619	0.663	0.649	0.682	0.697	0.669	0.66	0.640	0.663	0.641
RODEO $n = 4$ (real)	0.702	0.619	0.629	0.665	0.678	0.701	0.671	0.649	0.640	0.661	0.646
Offline	0.711	0.726	0.730	0.737	0.740	0.746	0.716	0.716	0.721	0.716	0.715

S4 Additional SLDA+Stream-Regress Object Detection Details

An overview of the incremental training stage for the SLDA+Stream-Regress object detection model is given in Alg. 2. We use the Fast RCNN model to extract features from edge box proposals. Given a new input, we then make classification and regression predictions using the SLDA and Stream-Regress models, respectively. For both the SLDA model and the Stream-Regress models, we use shrinkage regularization with parameters of $1e-2$ and $1e-4$, respectively.

We train the SLDA model as proposed in [15] with one slight modification. In [15], there was a single mean vector stored per class. However, in our work we allow SLDA to store two mean vectors per class, where one mean vector is representative of the actual class data and the second mean vector is representative of the background for that particular class. During test time, we thus obtain two scores for each class: the main class score and the background class score. We keep the main class score for each class and only keep the maximum score of all background scores.

Training the Stream-Regress model is similar to training the SLDA model. That is, we first initialize one mean vector $\mu_x \in \mathbb{R}^d$ to zeros, where d is the dimension of the data. We initialize another mean vector $\mu_y \in \mathbb{R}^m$ to zeros, where m is the number of regression targets,

```

Data: training set
Result: model fit to dataset
1 base initialization;
2 for image do
3   get edge box proposals;
4   get features, labels, and regression targets for edge box proposals;
5   get features and labels for ground truth;
6   freeze covariance matrix for SLDA model;
7   for box feat, label, regression targ in (edge box proposal features, edge box labels,
   edge box proposal regression targets) do
8     L2 normalize box feat;
9     if label is background then
10      | fit SLDA model on box feat and specific background label;
11    end
12    fit Stream-Regress model on box feat, label, and regression targ
13  end
14  unfreeze covariance matrix for SLDA model;
15  for box feat, label in (ground truth features, ground truth labels) do
16    L2 normalize box feat;
17    fit SLDA model on box feat and specific background label;
18  end
19 end

```

Algorithm 2: Incremental update procedure for SLDA+Stream-Regress.

and we have four regression coordinates per class including the background class. We also initialize two covariance matrices, $\Sigma_x \in \mathbb{R}^{d \times d}$ and $\Sigma_{xy} \in \mathbb{R}^{d \times m}$, and a total count of the number of updates, $N \in \mathbb{R}$.

Given a new sample $(\mathbf{x}_t, \mathbf{y}_t)$, where $\mathbf{y}_t \in \mathbb{R}^m$ is a one-hot encoding of the regression targets, we make the following updates to our model:

$$N = N + 1 \quad (1)$$

$$\mathbf{dx} = \mathbf{x}_t - \boldsymbol{\mu}_x \quad (2)$$

$$\mathbf{dy} = \mathbf{y}_t - \boldsymbol{\mu}_y \quad (3)$$

$$\Sigma_x = \Sigma_x + \frac{1}{N} \left(\frac{N-1}{N} \mathbf{dx}^T \mathbf{dx} - \Sigma_x \right) \quad (4)$$

$$\Sigma_{xy} = \Sigma_{xy} + \frac{1}{N} \left(\frac{N-1}{N} \mathbf{dx}^T \mathbf{dy} - \Sigma_{xy} \right) \quad (5)$$

$$\boldsymbol{\mu}_x = \boldsymbol{\mu}_x + \frac{\mathbf{dx}}{N} \quad (6)$$

$$\boldsymbol{\mu}_y = \boldsymbol{\mu}_y + \frac{\mathbf{dy}}{N} . \quad (7)$$

To make predictions, we first compute the precision matrix

$$\mathbf{\Lambda} = [(1 - \varepsilon)\mathbf{\Sigma}_x + \varepsilon\mathbf{I}]^{-1} , \quad (8)$$

with shrinkage parameter ε and identity matrix $\mathbf{I} \in \mathbb{R}^{d \times d}$. We then compute regression targets, $\hat{\mathbf{r}} \in \mathbb{R}^m$, for an input \mathbf{x}_t as:

$$\hat{\mathbf{r}} = \mathbf{x}_t \mathbf{A} + \mathbf{b} , \quad (9)$$

where $\mathbf{A} = \mathbf{\Lambda} \mathbf{\Sigma}_{xy}$ and $\mathbf{b} = \boldsymbol{\mu}_y - \boldsymbol{\mu}_x \mathbf{A}$.