# A APPENDIX

## A.1 TRAINING DETAILS

### A.1.1 HYPERPARAMETERS

All images are cropped to 224x224 and augmented by random resizing with a factor between $0.8$ and $1$ and random horizontal flips. All models are trained with a batch size of 128 using Adam **?** with a learning rate of $0.001$ with default betas, and evaluated with a batch size of 32. This is so that methods relying on optimizing a self-supervised criterion (TENT, TTT) will have approximately the same number of updates on the train set and the test set. For each model, we use early stopping on the validation set. We observed faster convergence on the validation set for non-hebbian models (within 20 epochs) than for hebbian models (within 50 epochs). Setting early stopping parameters accordingly, for non-hebbian models we train up to 25 epochs, with a patience of 5 epochs. For Hebbian models we train up to 50 epochs, with a patience of 10 epochs. For each method, we conducted hyperparameter searches on learning rates: $10^{-1}$, $10^{-2}$, $10^{-3}$ on seen and unseen combinations without continual shift. We used these hyperparameters for all continual shift runs. For TTT and TENT, we conducted a hyperparameter search on the test-time learning rate: $10^{-2}$, $10^{-3}$, $10^{-4}$ from ranges used in their papers on other datasets. We used these hyperparameters for all continual shift runs.

### A.1.2 TENT

We use Adam at test time, with a learning rate of $10^{-4}$ and default beta parameters.

### A.1.3 TTT

We use SGD at test time, with a learning rate of $10^{-4}$ and a momentum of $0.9$.

### A.1.4 HEBBIAN

At training time, in order to regularize the model, we reset the weights for each output hebbian neuron $\boldsymbol{W}_j^t$ by drawing randomly from the unit sphere every 32 examples. At test time, we reset the weights every 64 examples in the same way. During training, we perform BPTT over batches by detaching gradients for $W_{i,j}^t$ and $\Delta W_{i,j}^t$ every batch so memory use is constant per batch. We initialize the Hebbian update parameters from positive uniform distributions: $\boldsymbol{\eta}_{i,j} \sim U[0, 1e^{-3}]$ and $A_{i,j}, B_{i,j}, C_{i,j}, D_{i,j}, E_{i,j} \sim U[0, 1]$

### A.1.5 CLASS-SHIFT FINETUNED MODELS

The adaptive models fine-tuned with a class-shift ordering (ARM-CML(Cls:25x10), ARM-BN(Cls:25x10), LSTM(Cls:25x10)) are first initialized from the relevant model (ARM-CML, ARM-BN, LSTM, respectively) trained with i.i.d. sampling. We then freeze the backbone and fine-tune only the adaptive component (contextual network, batch-normalization parameters, LSTM, respectively) and the predictor heads on the training set (with a learning rate of $10^{-4}$), under a class-shift ordering. Results in Tab **??**. are from models finetuned on the training set with 25 tasks with 10 occurrences each. For the experiments sweeping over Occurrences (Figs 1-4), LSTM-tuned is finetuned on the training set with the same class shift ordering parameters ( of tasks,  of occurrences) that it is tested on.

### A.1.6 TRAINING COST

Each model was trained on a single Nvidia RTX 2080ti, taking between 3 (for the static baseline) and 16 hours (for the Hebbian method).

## A.2 IMAGE CORRUPTIONS

We evaluate our methods on Cifar-10 Corrupted (with the highest level of corruption - level 5) as in Sun et al. (2020). LSTM is competitive with UDA-SS on elastic; outperforming on glass, zoom, pixelate, and jpeg. LSTM is competitive with TTT on zoom, elastic, and pixelate; outperforming on

| Method | orig | gauss | shot | impul | defoc | glass | motn | zoom | snow | frost | fog | brit | contr | elast | pixel | jpeg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UDA-SS | 91.00 | 71.80 | 73.50 | 79.20 | 84.40 | 56.30 | 75.50 | 76.20 | 75.00 | 75.10 | 82.80 | 87.30 | 88.40 | 77.90 | 79.70 | 77.40 |
| TTT | 91.80 | 74.20 | 77.40 | 69.40 | 59.40 | 65.60 | 81.7 | 82.90 | 80.00 | 82.00 | 83.10 | 88.80 | 84.40 | 78.40 | 81.90 | 78.80 |
| LSTM | 84.15 | 66.13 | 67.70 | 42.66 | 78.57 | 69.07 | 70.37 | 81.33 | 71.12 | 69.67 | 57.34 | 78.64 | 29.32 | 76.39 | 79.88 | 79.21 |
| Hebb | 81.08 | 59.26 | 61.96 | 40.51 | 77.26 | 65.50 | 69.64 | 79.06 | 67.51 | 66.65 | 54.77 | 74.31 | 27.82 | 73.37 | 76.76 | 76.83 |

Table 3:

defocus, glass, and jpeg. Hebb is competitive with UDA-SS on jpeg; outperforming on glass and zoom. Hebb is competitive with TTT on glass and zoom; outperforming on defocus. LSTM is more robust than Hebb across a set of corruptions. Both methods perform significantly worse on pixel-level noise (Gauss, shot, impulse) and contrast corruption, while performing well on smoother corruptions (glass, zoom, pixelate, defocus, jpeg, elastic.) We believe these results demonstrate:

1. Methods that learn to adapt (LSTM  Hebb) can perform well on certain kinds of image corruptions that are smoother in their effects on the image.
2. Improved performance on the image corruption task is not linked to improved performance on semantic shift, and vice versa. These tasks test different abilities of methods in adapting to new distributions.

### A.3 TAU NORM CLASSIFIER ABLATION

:

| | No Semantic Shift (Seen Combinations) | | | Static Shift (Unseen Combinations) | | |
|---|---|---|---|---|---|---|
| $\tau$ | Att | Cls | Att+Cls | Att | Cls | Att+Cls |
| 0.1 | 28.02 | 34.50 | 14.70 | 13.22 | 16.77 | 1.67 |
| 0.2 | 28.05 | 35.54 | 14.69 | 13.09 | 16.75 | 1.73 |
| 0.3 | 28.10 | 34.45 | 14.65 | 13.00 | 16.67 | 1.76 |
| 0.4 | 27.95 | 34.41 | 14.51 | 12.94 | 16.57 | 1.78 |
| 0.5 | 27.99 | 34.31 | 14.57 | 12.98 | 16.49 | 1.83 |
| 0.6 | 27.89 | 34.32 | 14.54 | 12.96 | 16.42 | 1.88 |
| 0.7 | 27.63 | 34.25 | 14.34 | 12.95 | 16.33 | 1.90 |
| 0.8 | 27.64 | 34.18 | 14.27 | 12.94 | 16.30 | 1.96 |
| 0.9 | 27.43 | 34.16 | 14.20 | 12.92 | 16.22 | 1.99 |
| 1.0 | 27.47 | 34.18 | 14.26 | 12.78 | 16.18 | 1.99 |